

Old Dominion University

ODU Digital Commons

Engineering Management & Systems
Engineering Theses & Dissertations

Engineering Management & Systems
Engineering

Spring 2007

A Structured Systemic Framework for Software Development

Kevin MacGregor Adams
Old Dominion University

Follow this and additional works at: https://digitalcommons.odu.edu/emse_etds



Part of the [Software Engineering Commons](#), and the [Systems Engineering Commons](#)

Recommended Citation

Adams, Kevin M.. "A Structured Systemic Framework for Software Development" (2007). Doctor of Philosophy (PhD), Dissertation, Engineering Management & Systems Engineering, Old Dominion University, DOI: 10.25777/znb3-j517
https://digitalcommons.odu.edu/emse_etds/43

This Dissertation is brought to you for free and open access by the Engineering Management & Systems Engineering at ODU Digital Commons. It has been accepted for inclusion in Engineering Management & Systems Engineering Theses & Dissertations by an authorized administrator of ODU Digital Commons. For more information, please contact digitalcommons@odu.edu.

A STRUCTURED SYSTEMIC FRAMEWORK FOR SOFTWARE DEVELOPMENT

By

Kevin MacGregor Adams

B.S. Ceramic Engineering, May 1981, Rutgers University

M.S. Naval Arch & Marine Engineering, May 1986, Massachusetts Institute of Technology

M.S. Materials Engineering, May 1986, Massachusetts Institute of Technology

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

ENGINEERING MANAGEMENT

OLD DOMINION UNIVERSITY

May 2007

Approved by:

Chuck Keating (Director)

Rafael Landaeta (Member)

C. Ariel Pinto (Member)

Hasan Sayani (Member)

ABSTRACT

A STRUCTURED SYSTEMIC FRAMEWORK FOR SOFTWARE DEVELOPMENT

Kevin MacGregor Adams
Old Dominion University, 2007
Committee Director: Dr. Charles B. Keating

The purpose of this research was to develop and apply a systems-based framework for the analysis of software development project performance. Software development project performance is measured at the project level; that is, cost, schedule, and product quality that affect the overall project. To date, most performance improvement efforts have been focused on individual processes within the overall software development system. Making improvements to sub-elements, processes, or sub-systems without regard for the overall project is a classic misbehavior entered into by practitioners who fail to use a holistic, systemic approach. Attempts to improve sub-system behavior are at odds with *The Principle of Sub-optimization*. (van Gigch, 1974) The traditional method of predicting software development project performance, in terms of sub-system performance is too restrictive. A new holistic, systemic view based on systems principles offers a more robust way to look at performance.

This research addressed this gap in the systems and software body of knowledge by developing a generalizable and transportable framework for software project performance that is based on systems principles. A rigorous mixed-method research methodology, employing both inductive and case study methods, was used to develop and validate the framework. Two research questions were identified as integral to increasing the understanding of a systems-based framework.

- ✓ How does systems theory apply to the analysis of software development project performance?
- ✓ What results from the application of a systems-based analysis framework for analyzing performance on a software development project?

Using *Discoverers' Induction* (Whewell, 1858), a systems-based framework for the analysis of software development project performance was constructed, adding to the systems and software body of knowledge and substantiating a comprehensive and unambiguous theoretical construct for software development. Then, the framework was applied to two completed software development projects to support validation.

The structured systemic framework shows significant promise for contribution to software practitioners by indicating future software development project performance.

The research also made a contribution in the area of research methodologies by resurrecting William Whewell's *Discoverers' Induction* (1858) and furthering the use of the case study method in the engineering management and systems engineering domain, areas where their application has been very limited.

This dissertation is dedicated to my family; my parents, my wife, and my children, from whom I receive the strength and daily motivation to live life.

ACKNOWLEDGMENTS

I would like to thank my dissertation advisor, Dr. Chuck Keating, for his insight, counsel, and guidance throughout my doctoral endeavor, from enrollment to graduation. Dr. Keating is responsible for planting the systems seed that has grown into a passionate quest. To the members of my committee, my anonymous expert reviewer and panelists, for their critical evaluation and advice; their willingness to give up hours of their valuable time to review and critique my research played a key role in ensuring a quality product in the end. I owe a particular debt of gratitude to Dr. Laura Snyder of St. John's University. Dr. Snyder is an expert in the philosophy of science and the 19th century philosopher William Whewell. Her in-depth review and critical comments on my use of Whewell's *Discoverers' Induction* played a critical role in the decision to use an inductive method in engineering research.

To my students at the University of Maryland and Virginia Wesleyan College I extend a warm thank you for the unique opportunity to both teach you and learn from you. To my colleagues, Dr. John Richardson and Dr. Hasan Sayani in the Graduate School at the University of Maryland University College, a special thank you for the kindness and support you have shown me over the last seven years.

A special debt of gratitude goes out to the software and systems professionals who donated some of their valuable time to complete the development project questionnaires. Their support and in-depth knowledge of the systems analyzed in the case studies added validity to the data in this study.

To my mentor Dr. Robert N. Charette for his encouragement and support in all I have attempted in the field of systems and software engineering, I thank you and look forward to working with you on a number of exciting endeavors.

Finally, I would like to thank my family for their support. The many hours I spend helping students, reading, learning, and conducting research are all hours that you have freely given me as part of your love. To my children Jennifer, Brian, and Kristen thank you for sharing your busy lives with me, you are a source of inspiration and pride. To my parents, thank you for the solid education and support over the last 50 years. To my wife Sherry, thank you for your love, support, encouragement, and constant compassion. Last of all, to my sailboat Annabelle Lee, I promise, there is sailing after a dissertation.

TABLE OF CONTENTS

	Page
ABSTRACT	ii
ACKNOWLEDGMENTS	vi
TABLE OF CONTENTS	viii
LIST OF TABLES	xi
LIST OF FIGURES	xiv
GLOSSARY	xvi
PREFACE	1
CHAPTER I – INTRODUCTION	4
PURPOSE OF THE STUDY	5
RESEARCH OBJECTIVES	7
RESEARCH QUESTIONS	8
THEORETICAL FOUNDATION FOR THE RESEARCH	10
RESEARCH SIGNIFICANCE	12
STUDY LIMITATIONS	13
STUDY DELIMITATIONS	15
SUMMARY	16
CHAPTER II – LITERATURE REVIEW	18
RATIONALE AND APPROACH UNDERLYING THE REVIEW	18
LITERATURE SEARCH SCHEMA	21
BREADTH OF REVIEW	21
SYNTHESIS OF THE LITERATURE ON SYSTEMS PRINCIPLES	24
SYNTHESIS OF THE LITERATURE ON SYSTEMIC IMPROVEMENT	
FRAMEWORKS FOR SOFTWARE DEVELOPMENT	57
SYNTHESIS OF THE LITERATURE ON THE APPLICATION OF SYSTEMS	
PRINCIPLES TO SOFTWARE DEVELOPMENT	72
SYNTHESIS OF THE LITERATURE ON SOFTWARE DEVELOPMENT PROJECT	
PERFORMANCE	75

SYNTHESIS OF THE LITERATURE – THE RELATIONSHIP OF RESEARCH TO THEORY AND PRACTICE.....	79
CRITIQUE OF FINDINGS	82
SUMMARY	85
CHAPTER III – RESEARCH METHODOLOGY.....	87
EMPIRICAL SCIENCE AND METHODOLOGY	87
RESEARCH AND DISSERTATION CONCEPT	88
THE RESEARCH PERSPECTIVE	93
THE RESEARCHER’S VIEW	94
THE PROBLEM UNDER STUDY	101
RATIONALE FOR SELECTION OF THE MIXED METHOD DESIGN.....	106
CHALLENGES TO THE INDUCTIVE METHOD.....	110
CHALLENGES TO THE CASE STUDY METHOD	116
COMPLIANCE WITH THE CANONS OF SCIENCE	122
SUMMARY	127
CHAPTER IV: RESEARCH DESIGN, METHODS, AND PROCEDURE.....	129
THE RESEARCH DESIGN	129
METHOD FOR THE THEORETICAL FRAMEWORK DEVELOPMENT	134
METHOD FOR THE FRAMEWORK VALIDATION	145
THE DETAILED RESEARCH PROCEDURE.....	150
DATA ANALYSIS SOFTWARE	213
SUMMARY	215
CHAPTER V: RESEARCH RESULTS	216
INDUCTIVE DEVELOPMENT OF THE FRAMEWORK.....	216
VALIDATION OF THE FRAMEWORK	263
SUMMARY	269
CHAPTER VI: DISCUSSION OF RESULTS	271
RESEARCH OBJECTIVES AND QUESTIONS	271
INDUCTIVE METHOD IN ENGINEERING RESEARCH.....	273
SOFTWARE SYSTEM ENGINEERING	273
FRAMEWORK EMERGENCE	274

FRAMEWORK AREAS AND MEASURES.....	275
CROSS-CASE ANALYSIS.....	298
SUMMARY	304
CHAPTER VII: CONCLUSION	305
LIMITATIONS OF THE STUDY.....	305
IMPLICATIONS OF THE RESULTS	306
FUTURE RESEARCH RECOMMENDATIONS.....	309
SUMMARY	312
REFERENCES.....	314
APPENDIX A: ODU IRB RESEARCH EXEMPTION.....	339
APPENDIX B: GUIDELINES FOR THE OUTSIDE EXPERT.....	340
APPENDIX C: GUIDELINES FOR THE PANEL OF EXPERTS.....	351
APPENDIX D: LITERATURE VERIFICATION COMMENTS FROM THE OUTSIDE EXPERT.....	359
APPENDIX E: FRAMEWORK VERIFICATION COMMENTS FROM THE PANEL OF EXPERTS	361
APPENDIX F: DIA BHS CASE STUDY	366
APPENDIX G: FBI VCF SYSTEM CASE STUDY	428
VITA	478

LIST OF TABLES

<i>Table</i>	<i>Page</i>
Table 1: Scholarly Journals in Literature Review	23
Table 2: Ackoff's Machine-Age and Systems-Age Characteristics	32
Table 3: Evolution of the Systems Engineering Definition	40
Table 4: Summary of the Literature on Systems Principles	57
Table 5: Comparison of CMMI® Model Representations	61
Table 6: Software Process Improvements and CMM® Level.....	69
Table 7: Summary of the Literature on Systemic Improvement Frameworks for Software Development.....	71
Table 8: Systems-Based Methods for Software Development.....	73
Table 9: Summary of the Literature on the Application of Systems Principles to Software Development.....	75
Table 10: Summary of the Literature on Software Development Project Performance	78
Table 11: Literature Relationship to Research Purpose.....	79
Table 12: Distinguishing Characteristics of Qualitative and Quantitative Approaches	102
Table 13: Information Systems Research Methods	106
Table 14: Measures of Case Study Design Quality	109
Table 15: Scientific Basis for Generalization.....	117
Table 16: Case Study Criticisms and Mitigation Strategies	121
Table 17: Canons of Science and Design Quality Concepts.....	123
Table 18: Elements of the Research Design.....	133
Table 19: Forms of Theory and Characteristics	137
Table 20: Bourgeois' Theory-Building Format	139
Table 21: Strategies for Theory Construction and Features	139
Table 22: Terminologies used for Research Phases in Case Study Methods.....	147
Table 23: Strengths of Case Study Approach	148
Table 24: Structure for the Qualitative Element	152
Table 25: Qualifications for Expert Reviewer	156
Table 26: Characteristics of Open Coding	160
Table 27: Characteristics of Axial Coding	163
Table 28: Whewell's Methods for the Construction of Conception.....	166
Table 29: Characteristics of Selective Coding	166
Table 30: Properties for Relating Concepts within a Theory	167
Table 31: Construct Shaping Procedure	168
Table 32: Framework Verification Criteria.....	172
Table 33: Interaction Categories.....	176
Table 34: Levels of Efficiency of a Law of Interaction.....	177
Table 35: Characteristics used in Evaluating Utility.....	178
Table 36: Pragmatic Factors Affecting Frameworks	179
Table 37: Post-Development Validity Checks for Measurement Instruments.....	180
Table 38: Qualifications for Expert Panelists.....	181
Table 39: Structure for the Quantitative and Reporting Elements.....	184
Table 40: General Selection Criteria for Case Studies.....	185

Table 41: Software Development Project Selection Criteria.....	186
Table 42: Characteristics Missing from NCTP Diamond Model	189
Table 43: NCTPO Pentagon Model Characteristics and Measures.....	190
Table 44: Evidence and Data Collection Methods for Case Studies	193
Table 45: Levels of Questions in a Case Study Protocol	196
Table 46: Guidelines for Questionnaire Development.....	202
Table 47: Interpretive Strategy for the Case Study	206
Table 48: Needs of the Case Study Audience	209
Table 49: Characteristics of an Exemplary Case Study	211
Table 50: Sources for the Theoretical Concepts	223
Table 51: Sample Ordinal Scale	230
Table 52: Functions Elements Criteria and Source	234
Table 53: Process Measurement Criteria.....	235
Table 54: JR Measurement Criteria.....	236
Table 55: EA Measurement Criteria.....	236
Table 56: TOOL Measurement Criteria	237
Table 57: SITE Measurement Criteria.....	237
Table 58: ACAP/PCAP Measurement Criteria.....	240
Table 59: PCON Measurement Criteria	240
Table 60: APEX/PLEX/LTEX Measurement Criteria	240
Table 61: CTRL Measurement Criteria.....	241
Table 62: INT Measurement Criteria	242
Table 63: POL Measurement Criteria	243
Table 64: CC Measurement Criteria.....	243
Table 65: ATT Measurement Criteria	244
Table 66: RELY Measurement Criteria	245
Table 67: DATA Measurement Criteria.....	245
Table 68: RUSE Measurement Criteria.....	245
Table 69: DOCU Measurement Criteria.....	246
Table 70: TIME Measurement Criteria	246
Table 71: STOR Measurement Criteria.....	247
Table 72: PVOL Measurement Criteria	247
Table 73: COMP Measurement Criteria.....	249
Table 74: TECH Measurement Criteria	249
Table 75: LAW Measurement Criteria.....	251
Table 76: STD Measurement Criteria	252
Table 77: MAN Measurement Criteria.....	253
Table 78: MAT Measurement Criteria	253
Table 79: CAP Measurement Criteria	254
Table 80: PACE Measurement Criteria.....	254
Table 81: METH Measurement Criteria.....	254
Table 82: COMM Measurement Criteria	255
Table 83: OWN Measurement Criteria	256
Table 84: MGT Measurement Criteria	256
Table 85: CUST Measurement Criteria.....	257
Table 86: SUP Measurement Criteria.....	257

Table 87: USER Measurement Criteria.....	259
Table 88: POLT Measurement Criteria.....	263
Table 89: DIA BHS Function Element Scores.....	265
Table 90: DIA BHS Structure Element Scores	266
Table 91: DIA BHS Environment Element Scores.....	267
Table 92: FBI VCF Function Element Scores	268
Table 93: FBI VCF Structure Element Scores	269
Table 94: FBI VCF Environment Element Scores.....	270
Table 95: High-Level FSE Framework Result for DIA BHS.....	276
Table 96: High-Level FSE Framework Result for FBI VCF.....	288

LIST OF FIGURES

<i>Figure</i>	<i>Page</i>
Figure 1: Structure for the Inquiry	7
Figure 2: Systems Science and Software Engineering Project Management.....	12
Figure 3: Schema for Literature Review	22
Figure 4: Systems Principles.....	24
Figure 5: Systems Approaches	25
Figure 6: Simplified View of Entropy.....	45
Figure 7: Systemic Improvement Frameworks for Software Development.....	57
Figure 8: Evolution of Major Software and Systems Standards.....	58
Figure 9: Application of Systems Principles to Software Development	72
Figure 10: Software Development Project Performance	75
Figure 11: Literature Threads	81
Figure 12: Research and Dissertation Concept	91
Figure 13: Elements of the Research Design	94
Figure 14: The Researcher's Perspective	95
Figure 15: The Ontological Continuum.....	96
Figure 16: Epistemological Schools of Thought.....	97
Figure 17: Continuum of Ontological Assumptions and Epistemological Stances	97
Figure 18: Approaches to Qualitative Analysis	108
Figure 19: Lee & Baskerville's Generalizability Framework.....	118
Figure 20: Making Inferences: Two Levels	119
Figure 21: High-Level Research Design and Study Phases.....	130
Figure 22: Framework Representation Continuum.....	135
Figure 23: Model-based Understanding of Theories	136
Figure 24: Flow Diagram of Observable to Inference	159
Figure 25: Schema for the Decomposition of Facts.....	161
Figure 26: General Classification Schema	165
Figure 27: Steps in the Augmented <i>Discoverers' Induction</i> Procedure	173
Figure 28: The Theory Building Triangle	183
Figure 29: Software Development Project Grid.....	187
Figure 30: NCTPO <i>Pentagon</i> Model.....	191
Figure 31: The Case Study Chain of Evidence	204
Figure 32: Normative Theory Triangle	212
Figure 33: The Transition from Descriptive Theory to Normative Theory	213
Figure 34: Subcategories.....	219
Figure 35: Tree Node Hierarchy.....	221
Figure 36: How Categories and Concepts Influenced the Framework.....	222
Figure 37: Theoretical Concepts Underlying the Framework	224
Figure 38: High-Level Framework and Construct Elements.....	226
Figure 39: The Four Types of Scales.....	229
Figure 40: Functions Construct of the Framework	233
Figure 41: Structure Construct of the Framework	239
Figure 42: Environment Construct of the Framework	251

Figure 43: Grid of Organizational Behavior	262
Figure 44: Decision Tree used Evaluate Empirical Data using the FSE Framework	264
Figure 45: Domain of Software Development Projects in Research	275
Figure 46: DIA BHS Project Characteristics on NCTPO Pentagon Model	276
Figure 47: DIA BHS on the FSE Framework Grid	277
Figure 48: Diagram for DIA BHS Development Area	278
Figure 49: Diagram for DIA BHS Improvement & Training Area	279
Figure 50: Diagram for DIA BHS Life Cycle Support Area	279
Figure 51: Diagram for DIA BHS Management Area	280
Figure 52: Diagram for DIA BHS Social System Area	281
Figure 53: Diagram for DIA BHS Cybernetic Controls Area	282
Figure 54: Diagram for DIA BHS Technical System Area	283
Figure 55: Diagram for DIA BHS Resources Area	284
Figure 56: Diagram for DIA BHS Stakeholders Area	285
Figure 57: Diagram for DIA BHS External Controls and Infrastructure Areas	286
Figure 58: FBI VCF Project Characteristics on NCTPO Pentagon Model	287
Figure 59: FBI VCF on the FSE Framework Grid	288
Figure 60: Diagram for FBI VCF Development Area	289
Figure 61: Diagram for FBI VCF Improvement & Training Area	290
Figure 62: Diagram for FBI VCF Life Cycle Support Area	291
Figure 63: Diagram for FBI VCF Management Area	291
Figure 64: Diagram for FBI VCF Social System Area	292
Figure 65: Diagram for FBI VCF Cybernetic Controls Area	293
Figure 66: Diagram for FBI VCF Technical System Area	294
Figure 67: Diagram for FBI VCF Resources Area	295
Figure 68: Diagram for FBI VCF Stakeholders Area	296
Figure 69: Diagram for FBI VCF External Controls and Infrastructure Areas	297
Figure 70: Cross-Case Diagram for Project Characteristics	299
Figure 71: Cross-Case Diagram for Improvement & Training Area	299
Figure 72: Cross-Case Diagram for Life Cycle Support Area	300
Figure 73: Cross-Case Diagram for Management Area	301
Figure 74: Cross-Case Diagram for Cybernetic Functions Area	302
Figure 75: Cross Case Diagram for Resources Area	302
Figure 76: Cross-Case Diagram for Stakeholders Area	303

GLOSSARY

ACM	Association for Computing Machinery.
Axial Coding	The process of relating categories to their subcategories, termed <i>axial</i> because coding occurs around the axis of a category, linking categories at the level of properties and dimensions (Strauss & Corbin, 1998, p. 123).
Axiology	Ethics, the responsibility of a researcher for the consequences of his/her research approach and its results (Iivari, Hirschheim, and Klein, 1998, p. 175).
BHS	Baggage Handling System
Canons of Science	The universal scientific standard for all research. The canons include: (1) Significance or Truth Value, this canon addresses the credibility of the research findings; (2) Applicability, this canon addresses how transferable and applicable the findings of the research are to other setting and contexts. This is often referred to as generalizability; (3) Consistency, this canon addresses the ability of the findings to be replicated by other researchers. This is often referred to as replicability; and (4) Neutrality, this canon addresses the finding of the research and ensures that they are a direct result of the inquiry and not a function of the prejudice and/or bias of other the researcher or the particular research design (Lincoln & Guba, 1985, pp. 294-301).
Categories	Concepts that stand for phenomena (Strauss & Corbin, 1998, p. 101).
CMM[®]	Capability Maturity Model for Software. (SEI, 2002)
CMMI[®]	Capability Maturity Model Integration. (SEI, 2002)
COCOMO II	Constructive Cost Model Version II. An objective cost model for planning and executing software projects. (Boehm et al, 2000, p. xxvii.)
Codes	Tags or labels for assigning units of meaning to the descriptive or inferential information compiled during a study. Codes usually are attached to <i>chunks</i> of varying size – words, phrases,

	sentences, or whole paragraphs, connected or unconnected to a specific setting (Miles & Huberman, 1994, p. 56).
Coding	The part of analysis that involves how to differentiate and combine the data retrieved and the reflections made about the information (Miles & Huberman, 1994, p. 56).
Coherence	Deals with the adequacy of a hypothesis. A hypothesis should be sufficient to explain some view of the subject which is consistent with all the observed facts (Whewell, 1858, p. 85).
Colligation	The mental operation of bringing together a number of empirical facts by superinducing upon them some idea or conception that unites the facts and renders them capable of being expressed by a general law (Snyder, 1997a, p. 585).
Concept	An abstraction formed by generalization from particulars (Kerlinger & Lee, 2000, p. 40). The building blocks of theory (Strauss & Corbin, 1998, p. 101).
Consilience	A hypothesis' ability to explain and predict cases of a different kind from those which were contemplated in the initial formation of the hypothesis (Ducasse, 1951b, pp. 229-230).
Construct	A concept; with the added meaning of having been deliberately and consciously invented or adopted for a special scientific purpose (Kerlinger & Lee, 2000, p. 40). Something that the scientist puts together from his own imagination, something that does not exist as an isolated, observable dimension of behavior (Nunnally, 1967, p. 85).
Complementarity	The apparently incompatible sorts of information about the behavior of the object under examination which we get by different experimental arrangements can clearly not be brought into connection with each other in the usual way, but may, as equally essential for an exhaustive account of all experience, be regarded as <i>complementary</i> to each other (Bohr, 1937, p. 291).
Conception	The special modification of ideas which are exemplified in particular facts. Conceptions; as a circle, a square number, an accelerating force, a neutral combination of elements, a genus (Whewell, 1858, p. 31).
Customer	The company, organization, or person who is paying for the software system to be developed (Pfleeger, 1998, p.14).
DIA	Denver International Airport

Dimensions	The range along which general properties of a category vary, giving specification to a category and variation to the theory (Strauss & Corbin, 1998, p. 101).
Element	The highest level of the research design. The distinction between elements is made based on the methodological differences (i.e. qualitative, quantitative, and reporting).
Epistemology	The nature of knowledge and the proper methods of inquiry (Iivari, Hirschheim, and Klein, 1998, p. 174).
Explication of Concepts	The clear development from Fundamental Ideas in the discoverer's mind, as well as their precise expression in the form of Definitions or Axioms when that can be done (Whewell, 1858, p. 49).
Face Validation	The extent to which an instrument <i>looks like</i> it measures what it is intended to measure. It concerns judgments about an instrument <i>after</i> it is constructed (Nunnally, 1967, p. 99).
FBI	Federal Bureau of Investigation
Framework	A type of model; a conceptual model that can be applied to carry out some specific purpose, function or task.
Induction	The <i>process</i> of a true Colligation of Facts by means of an exact and appropriate conception. An <i>Induction</i> is also employed to denote the <i>proposition</i> which results from this process (Whewell, 1858, p. 70) [<i>Author's Note: Although this is far from being a univocal term, it accurately represents Whewell's view and its use throughout the dissertation.</i>].
IEEE	Institute of Electrical and Electronics Engineers.
IRB	Institutional Review Board.
IS	Information Systems.
ISD	Information Systems Development.
ISO/IEC	International Organization for Standardization and International Electro-technical Commission.
Measure	An observed score gathered through self-report, interview, observation, or some other means (Edwards & Bagozzi, 2000, p.156).

Milestone	A point in time during the research, independent of tier (element, phase or step), when a major product is produced or a decision is made.
Model	An interpretive description of a phenomenon (object or process) that facilitates perceptual as well as intellectual access to that phenomenon. 'Description' is intended as a term wide enough to admit various forms of external representations, propositional or non-propositional. A model is not, however, a description in the trivial sense of a mere phenomenological description of a phenomenon. It gives a description that is an interpretation in that the description goes beyond what 'meets the eye', e.g. by exploiting a theoretical background that is relevant to interpreting the phenomenon (Bailer-Jones, 2003, p. 61).
NCTPO Pentagon	Project characterization model that uses novelty, complexity, technology, pace and organizational maturity as criteria.
NVivo7	A computer-assisted qualitative data analysis software package used to help qualitative researchers with open, axial, and selective coding. (Gibbs, 2002)
OQE	Objective Quality Evidence. Any statement of fact pertaining to the quality of a product or service based on observations, measurements, or tests which can be verified. Verifiable evidence includes traceable records and other documents that support assertions that deliberate steps were taken to comply or perform against established requirements or criteria. (NAVSEA, 2007)
Ontology	The structure and properties of what is assumed to exist (Iivari, Hirschheim, and Klein, 1998, p. 172).
Open Coding	The analytic process through which concepts are identified and their properties and dimensions are discovered in data (Strauss & Corbin, 1998, p. 101).
Ordinal Scale	A scale in which (1) a set of objects is ordered from most to least with respect to an attribute, (2) there is no indication of how much in an absolute sense any of the objects possess the attribute, and (3) there is no indication of how far apart the objects are with respect to the attribute (Nunnally, 1967, p. 12).
Performance	The degree to which a software development project accomplishes its cost, schedule and business goals during the development of the software. [<i>Author's Note that this definition</i>

does not include the performance of the software after delivery.]

Phase	The 2 nd level of the research design. The distinction between phases is made based upon the major activities performed.
Phenomena	Central ideas in the data presented as concepts (Strauss & Corbin, 1998, p. 101).
Prediction	A hypothesis' ability to foretell phenomena which have yet to be observed; at least all phenomena of the same kind as those which the hypothesis was invented to explain. The prediction of results, even of the same kind as those which have been observed, in new cases, is the proof of real success in the inductive processes. It is important to note that prediction, as used in this study, assumes no causal relationship (Whewell, 1858, pp. 86-87).
Project Management System	The structured set of technical and human entities that interact both formally and informally within a specific context to produce results. The products of interaction are patterns of decision, action, and interpretation that drive project performance (Keating & Varela, 2004).
Properties	Characteristics of a category, the delineation of which defines and gives it meaning (Strauss & Corbin, 1998, p. 101).
PSP	Personal Software Process.
PMBOK[®]	Project Management Body of Knowledge. (PMI, 2004)
Reliability	The extent to which an experiment, test, or any measuring procedure yields the same results on repeated trials (Carmines & Zeller, 1979, p. 11).
Research Design	The overall protocol for the research. This includes the research purpose, as articulated in the research questions and propositions. It also includes the detailed research plan and its three tier structure of elements, phases, and steps.
Satisficing	The principle that states that there is a choice mechanism that will lead to a path that will permit satisfaction at some specified level of all needs (Simon, 1956, p. 136).
SEI	Software Engineering Institute

Selective Coding	The systematic technique for integrating and refining categories. Categories are reviewed to identify the central category that represents the main theme of the research. The central category has analytic power because it can pull the other categories together to form an explanatory whole. (Strauss & Corbin, 1998, pp. 143-161).
Software Engineering	The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software (IEEE 610.12, 1990, p. 67).
Software Engineering Management	The application of management activities – planning, coordinating, measuring, monitoring, controlling, and reporting – to ensure that the development and maintenance of software is systematic, disciplined, and quantified (Abran & Moore, 2004, p. 8-1).
SPI	Software Process Improvement.
SPM	Software Project Management.
Step	The 3 rd and lowest level of the research design. A step is a specific and unique technique, procedure, or method taken in conducting the research. A step supports a phase.
STS	Sociotechnical Systems. These systems have both a technical subsystem, made up of the facilities, tools, equipment and knowledge necessary to execute processes in support of product development, and a social subsystem, which is made up of the people working on the processes.
Subcategories	Concepts that pertain to a category, giving it further clarification and specification (Strauss & Corbin, 1998, p. 101).
Sub-optimization	The principle that states that if each subsystem, regarded separately, is made to operate with maximum efficiency, the system as a whole will not operate with utmost efficiency. (van Gigch, 1974)
Superinduction	Bringing new (superinducing) concepts to a phenomenon that involve identifying what a set of facts share in order to construct general principles, laws, or propositions about the phenomena (Snyder, 1999, p. 542).

SWEBOK®	Software Engineering Body of Knowledge. (Abran & Moore, 2004)
System	An assembly of elements related in an organized whole. An element is the representation of some phenomena of the natural or social world by a noun or by a noun phrase that informed observers agree exists, or could exist. An element must normally be capable of behavior such that it has some significant attributes that may change. Relationships exist between elements if the behavior of either influences or controls the other (Flood & Carson, 1993, p.7).
Systems Practice	Using the product of systems thinking to initiate and guide actions we take in the world Systems Practice (Checkland, 1993, p. 4).
Systems Principles	Systems knowledge, in the scientific hierarchy that includes laws, principles, theorems, hypotheses, and axioms.
Systems Science	Any effort to employ a systemic outlook in doing basic or applied science according to the conventional ideas of non-reflective positivistic empirical-analytical rationality [objective data, testable hypotheses, valid modeling and so on.] (Flood, 1990, p. 217).
Systems Theory	There exist models, principles, and laws that apply to generalized systems or their subclasses, irrespective of their particular kind, the nature of their component elements, and the forces between them. A consequence of the existence of general system properties is the appearance of structural similarities or isomorphisms in different fields. There are correspondences in the principles that govern the behavior of entities that are, intrinsically, widely different. This correspondence is due to the fact that the entities concerned can be considered, in certain respects, as systems, i.e., complexes of elements standing in interaction (von Bertalanffy, 1968, pp. 32-33).
Systems Thinking	Conscious use of the particular concept of wholeness captured in the word systems, to order our thoughts (Checkland, 1993, p. 4).
Theoretical Saturation	The point in category development at which no new properties, dimensions, or relationships emerge during analysis (Strauss & Corbin, 1998, p. 143).

Theory	A set of well-developed concepts related through statements of relationship, which together constitute an integrated framework that can be used to explain or predict phenomena (Strauss & Corbin, 1998, p. 15).
Theories of the Middle Range	Middle range theories are solutions to problems that contain a limited number of assumptions and considerable accuracy and detail in the problem specification. The scope of the problem is also of manageable size. To look for theories of the middle range is to prefigure problems in such a way that a number of opportunities to discover solutions are increased without becoming infinite (Weick, 1989, p. 521).
TSP	Team Software Process.
User	The person or people who will actually use the system: the ones who sit at the terminal or submit the data or read the output (Pfleeger, 1998, p. 14).
Validity	The extent to which any measuring instrument measures what it is intended to measure (Carmines & Zeller, 1979, p. 17).
VCF	Virtual Case File
VSM	Viable Systems Model.

PREFACE

The genesis for this work was two-fold: my professional involvement in the analysis, design, development and implementation of complex software systems and my exposure to the emancipating concepts and principles of systems science in my doctoral studies.

As a classically trained engineer with degrees from the engineering schools at Rutgers and MIT, I had been taught in the reductionist mold; to approach problems by breaking them into bite sized chunks, solving the problem, and returning them to the larger system. This was reinforced during twenty years of highly successful practical, hands-on experience as a Navy submarine officer responsible for the operation, maintenance, and supervision of nuclear submarines. I applied the same methods when, late in my Navy career, I was assigned to an organization responsible for the design and development of complex software systems for the Department of Defense. This endeavor did not respond well to the traditional reductionist approach to engineering that had been so successful in my earlier career. The software development project literature was rife with improvement models and methodologies, none of which could provide the sought after *Silver Bullet* (Brooks, 1987) promised in the marketing hype. After retirement from the Navy my work with complex software systems continued, and much to my dismay, the commercial contractors had the same problems (although paid more money) delivering software development projects of requisite quality, on-time and within budget.

My exposure to the modern works associated with systems science was a revelation. The conscious use of the concept of wholeness captured in the word *system* was an antithetical approach to my classical training and experience as an engineer. The

words holism and systemic, so frequently used by systems practitioners, are founded on the basic understanding of wholeness. I methodically absorbed the early classical works in systems science by Smuts (1926), Ashby (1947, 1956), Boulding (1956), Churchman (1968), Emery (1969), von Bertalanffy (1969), Ackoff (1971, 1974, 1979a, 1979b) and Beer (1979, 1981, 1984). These systems-based approaches to real-world problems included the rich contextual environment that surrounded the systems they were investigating. I moved on to the modern works in systems science by van Gigh (1974), Jackson (1991), Checkland (1993), Flood & Carson (1993) and Gharajedaghi (1999). Based on these readings I found the beginnings of an entirely new worldview based on systems science. My new, holistically based worldview encompassed how *systems principles* (the scientific hierarchy that includes laws, principles, theorems, hypotheses, and axioms) support *systems theory*, which in turn promotes *systems thinking*, which can be used in *systems practice* to improve effectiveness in management and systems problem solving.

This was the genesis for the application of systems principles to the problems now facing the software engineering community. But why hasn't this been done before? There are plenty of smart software engineers that must understand the linkage between systems science and software development project performance. I think not, but why not?

Thomas Kuhn (1922-1996) states (1996):

A scientific community consists, on this view, of the practitioners of a scientific specialty. To an extent unparalleled in most fields, they have undergone similar educations and professional initiations; in the process they have absorbed the same technical literature and drawn many of the same lessons from it. (p. 177)

Wernick & Hall (2004) examined the underlying belief system controlling the mindset of software engineering and found that:

Software engineering theory, viz. the development of methods, and tools to support the development of software, is currently in a state analogous to a Kuhnian pre-paradigm discipline. (p. 241)

With software engineering stuck in a pre-paradigm mode, a shift is required. The use of a Kuhnian approach may provide the mechanism through which the software engineering community may embrace systems-based thought as part of their shared belief system or disciplinary matrix. Wernick & Hall (2004) propose a long-term vision of a discipline they call software-based systems engineering. This is my new worldview. The application of systems science to help create a new paradigm in software engineering, one that will permit software engineering to use mechanisms and principles that will enable the discipline to produce software with predictable results and high levels of confidence. The research to develop and apply a systems-based framework for the analysis of software development project performance is but a step in developing the new paradigm for software engineering.

CHAPTER I – INTRODUCTION

The rapid and unprecedented growth in software has brought with it some of the most spectacular and costly project failures in modern history. Various studies have shown the extent of these failures. The Standish Group's Bi-Annual Chaos Research Study (2003) estimated that in 2002 American companies and government agencies will have spent \$38 billion for cancelled software projects and that these same organizations will have paid an additional \$17 billion for software projects that will be completed, but will exceed their original time estimates by an average of 82%. An earlier empirical study of 72 information systems development projects in 23 major U.S. firms, reported an average effort overrun of 36% and an average schedule overrun of 22% (Genuchten, 1991). The crisis has not limited itself to large diversified corporations and government agencies but affects systems from baggage handling to satellite navigation (Gibbs, 1994). In the past 15 years alone, software defects have wrecked a European satellite launch, delayed the opening of the hugely expensive Denver airport for a year, destroyed a NASA Mars mission, killed four Marines in a helicopter crash, induced a U. S. Navy ship to destroy an airliner, and shut down ambulance systems in London, contributing to as many as 30 deaths (Mann, 2002).

Why do software development projects have such a poor record of completion? Why do some projects succeed where others fail? Software development projects use a wide-range of formal software engineering methods and techniques to deliver software products to their customers. The record of accomplishment over the last 40 years has been less than stellar and the subject of countless studies, commissions, methods and panaceas for improvement. The improvement efforts have used traditional, reductionist

engineering analysis to dissect problems into small, bite-sized elements which are carefully analyzed, improved and returned to the larger effort as part of a generalized improvement process. The elemental approach provides focused improvement in the area of study but contributes little to the larger software development project improvement process. Few improvement efforts have taken a systemic, holistic view of the problem.

This chapter provides an introduction to the research by presenting the purpose of the study and the research questions with an explanation of the intent of each question. The chapter is concluded with a discussion of the significance of the research and the study limitations and delimitations.

PURPOSE OF THE STUDY

The purpose of this research was to develop and apply a systems-based framework for the analysis of software development project performance. Software development project improvement is continually strived for at the project level; that is, improvements in cost, schedule, and product quality that affect the overall project. To date, most improvement efforts have been focused on individual processes within the overall software development project management system. The project management system is defined as “. . . the structured set of technical and human entities that interact both formally and informally within a specific context to produce results. The products of interaction are patterns of decision, action, and interpretation that drive project performance.” (Keating & Varela, 2004, p. 2) Making improvements to project sub-elements, processes, or sub-systems without regard for the overall project management system is a classic misbehavior entered into by practitioners who fail to use a holistic, systemic approach when managing a project. A project is a system, made up of

subsystems. Attempts to optimize sub-system behavior are at odds with *The Principle of Sub-optimization* which states that if each subsystem, regarded separately, is made to operate with maximum efficiency, the system as a whole will not necessarily operate with utmost efficiency (van Gigch, 1974).

The traditional method of predicting software development project performance, in terms of sub-system performance may be too restrictive. A new holistic, systemic view may reveal a better way to look at project performance. The present research included an inductive theory building component where a systems-based framework was constructed for software development. The framework was used for analysis of software development projects. Applicability to completed software development projects was considered to support validation of the structured systems-based framework produced by the research effort. Projects that meet business objectives, are completed on-time and within budget constitute the generally accepted standard definition of project success (Pinto & Slevin, 1988; Jones, 1995; Baccarini, 1999; Linberg, 1999; Jiang, Klein & Discenza, 2002a) Therefore, consistent with this preponderance of literature, performance, as used in this research, has been defined as the degree to which a software development project accomplishes its cost, schedule and business goals during the development of the software (note that this definition does not include the performance of the software after delivery).

The overall structure for the inquiry is presented in Figure 1. The research purpose was supported by two research objectives and two research questions which are detailed in the following sections.

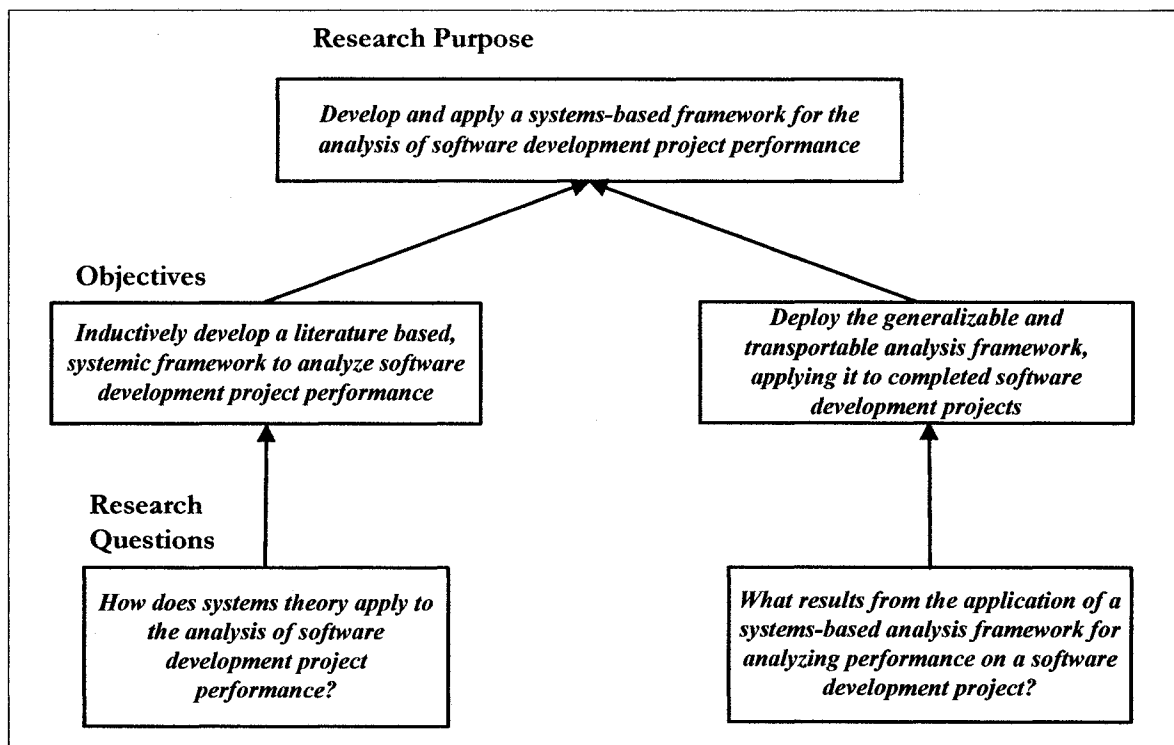


Figure 1: Structure for the Inquiry

RESEARCH OBJECTIVES

Supporting the purpose of the research were two focused objectives. The first objective was to:

Inductively develop a literature based, systemic framework to analyze software development project performance.

The framework was developed using a literature-intensive research effort where the existing literature on the object of the study served as the input to the inductive method. Using relevant scholarly literature was important for establishing validity in the research and confidence in the findings (Patton and Appelbaum, 2003).

The second objective was to:

***Deploy the generalizable and transportable analysis framework,
applying it to completed software development projects.***

In order to support validation of the framework, it was deployed on two completed, real-world software development projects. The framework's ability to predict software development project performance was the focus of the validation. Two completed software development projects were used because "... multiple-case designs allow for cross-case analysis and the extension of theory." (Benbasat, Goldstein & Mead, 1987, p. 373)

Each objective was supported by a single, focused research question which guided that element of the research.

RESEARCH QUESTIONS

The application of a structured systemic framework for software development may provide insight into the failure to achieve overall software development project (system) improvement despite improvements to individual software development processes (sub-systems). To address the purpose of the study, the research was designed with two elements. The first element was to build upon the existing foundation of systems theory by focusing on answering the following research question:

How does systems theory apply to the analysis of software development project performance?

The research used an inductive method to develop a theoretical framework for software development. The framework was literature-based and developed using an inductive method called *Discoverers' Induction* (Snyder, 1997a). The framework was a conceptual model that could be applied to software development projects to enhance project

performance. The framework was not a detailed step-by-step methodology, but a model that could serve as an outline for the articulation of software development project processes using systems theory. The overall goal was to produce a generalizable and transportable framework for the analysis and evaluation of software development projects by articulating systems theory within the software engineering body of knowledge. The strength of the framework was established from grounding in the theoretical constructs derived from the systems theory body of knowledge.

The goal of the second element of the research was to validate the inductively developed holistic, structured, systemic framework for software development projects on actual real-world software projects by answering the following question:

What results from the application of a systems-based analysis framework for analyzing performance on a software development project?

Because the validation used real-world software development projects, a case study method was selected. The scientific basis for case study generalization was differentiated from the more familiar experimental generalization where data is generalized to larger samples or populations. The case study approach used a method of generalization called analytic generalization in which “. . . the investigator is striving to generalize a particular set of results to some broader theory.” (Yin, 2003, p. 37) Analytic generalization involved generalizing to a theory or in this case a framework—not to a population—and was based on validating theory-driven or framework-driven predictions with evidence collected in a variety of real-world settings in the case studies. Analytic generalization can reveal contextual conditions under which the framework-based predictions would be

considered to apply and served to increase confidence in the theory as instantiated in the framework. For the research the inductively developed systems-based framework for the analysis of software development project performance was used as a template for comparing the empirical results (i.e. data) of both case studies to the inductively developed theoretical framework. This element of the research was centered on analysis of the empirical data from the case studies and comparison with the descriptive theory presented in the framework.

THEORETICAL FOUNDATION FOR THE RESEARCH

Systems Science is the theoretical foundation for the research. The formal definitions associated with systems science are essential in understanding the relationship between systems theory, systems principles, systems thinking, and systems practice; and in this case, their relationship to software engineering and software engineering management.

Software Engineering: *The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (IEEE 610.12, 1990, p. 67)*

Software Engineering Management: *The application of management activities – planning, coordinating, measuring, monitoring, controlling, and reporting – to ensure that the development and maintenance of software is systematic, disciplined, and quantified. (Abran & Moore, 2004, p. 8-1)*

Systems Theory: *There exist models, principles, and laws that apply to generalized systems or their subclasses, irrespective of their particular kind, the nature of their component elements, and the forces between them. A consequence of the existence of general system properties is the appearance of structural similarities or isomorphisms in different fields. There are correspondences in the principles that govern the behavior of entities that are, intrinsically, widely different. This correspondence is due to the fact that the entities concerned can be considered, in certain respects, as systems, i.e., complexes of elements standing in interaction. (von Bertalanffy, 1968, pp. 32-33)*

Systems Principles: *Systems knowledge, in the scientific hierarchy that includes laws, principles, theorems, hypotheses, and axioms.* (see Skyttner, 2001, pp. 88-101; Clemson, 1991, pp. 199-257)

Systems Thinking: *Conscious use of the particular concept of wholeness captured in the word systems, to order our thoughts.* (Checkland, 1993, p. 4)

Systems Practice: *Using the product of systems thinking to initiate and guide actions we take in the world.* (Checkland, 1993, p. 4)

The preceding definitions are essential elements in understanding the relationship between systems science and software engineering. It is important to note that systems principles (i.e., the scientific hierarchy that includes laws, principles, theorems, hypotheses, and axioms associated with systems) are the foundation for all systems endeavors. These principles (see Skyttner, 2001, pp. 88-101 and Clemson, 1991, pp. 199-257) form the body of theory related to systems. Boulding (1956) categorizes them as:

“... a body of systematic theoretical constructs which will discuss the general relationships of the empirical world. (p. 197)”

The model in Figure 2 shows how systems principles are the foundation for systems theory, which in turn promotes systems thinking, which can be used in systems practice to improve effectiveness in software engineering and software development project management.

The model in Figure 2 served to guide the research. The model's importance is derived from its' ability to relate systems principles to the goal of the research; *development and application of a systems-based framework for the analysis of software development project performance.* Additional value was derived from the model's ability to depict the generalizability of the research goal to both software project management and the larger field of software engineering.

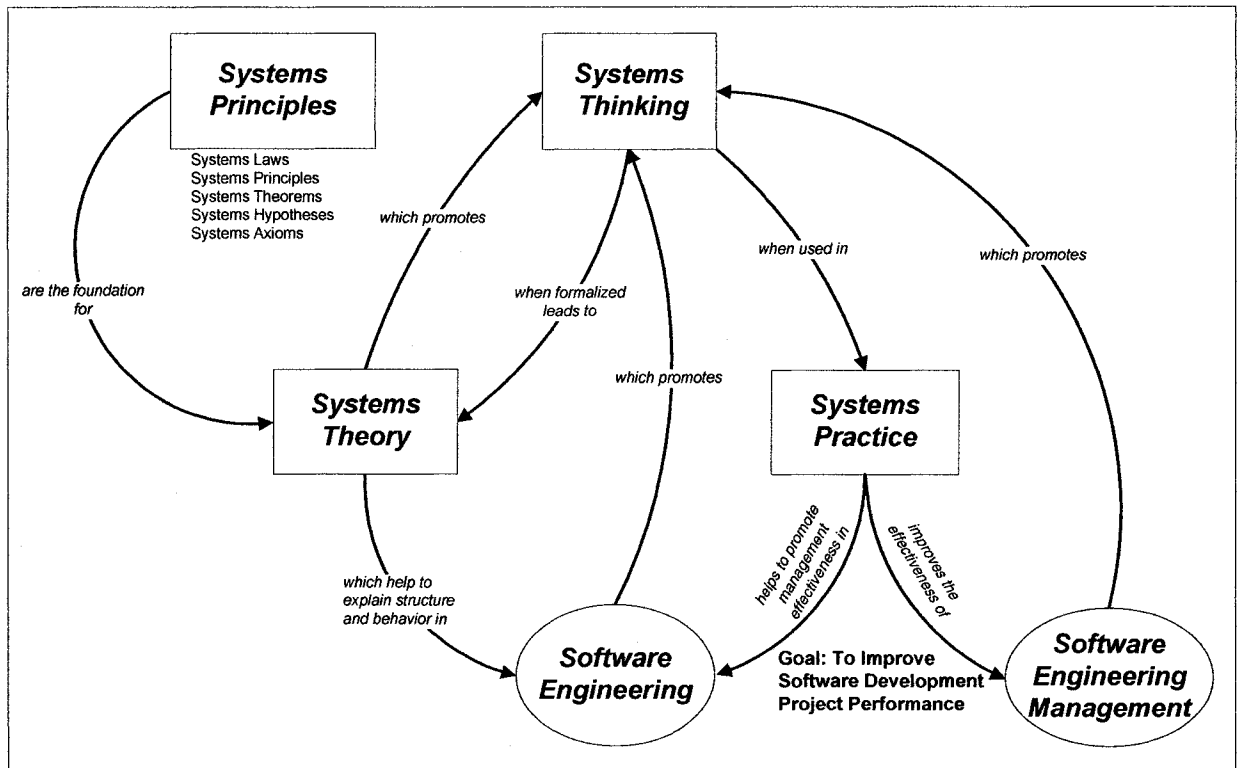


Figure 2: Systems Science and Software Engineering Project Management

RESEARCH SIGNIFICANCE

As will be further elaborated in the next chapter, the literature has established that a major gap in the recent research on software project management exists in the treatment of software development projects as an organized or complex whole; a system. The software engineering community has been unable to coherently integrate their knowledge of the individual software development and management processes (sub-systems) in order to better understand the overall socio-technical system in which each of the development and management processes exists.

This research makes four significant contributions to systems and software engineering and one to the body of knowledge on qualitative research. First, it has added to the existing body of knowledge in systems theory, systems-based methods, and

software engineering by developing an extensible framework, grounded in systems principles, for evaluating and assessing software development projects. Secondly, it has expanded the domain of systems methodologies by providing a systems-based framework for the assessment and evaluation of complex software engineering development projects as part of the overall software development performance improvement process. Third, the research has made a significant contribution to software project management practitioners who, as part of their discipline, now have a generalizable and transportable framework that can act as a *systems lens* for use in assessing and evaluating software development project performance. Fourth, this research has provided areas for future research that include the conduct of additional case studies and/or expanded use of the framework. Finally, this research has contributed to the body of knowledge on qualitative research through an elaboration of Whewell's (1858) Method of *Discoverers' Induction*. Whewell's method has been augmented with modern techniques for decomposing and classifying facts and constructing the conception while remaining loyal to his concept of induction and the colligation of facts by means of a conception.

STUDY LIMITATIONS

This section addressed three research limitations required to ensure the study maintained the proper research focus and accomplished the research purpose. The limitations to the research were: (1) the use of a qualitative element where a subjective approach and inductive methodology were used to build the framework, (2) the use of a quantitative element where an objective approach and case study methodology were used to validate the utility of the framework on real-world software development projects, and

(3) the ability to generalize from a case study. All three limitations will be explored in detail below.

The common challenges to the inductive method are registered by those who adhere to hypothetico-deductive positions. As discussed in the section *Challenges to the Inductive Method* in Chapter 3 on *Research Methodology*, induction is a *rational method of discovery*. Whewell (Snyder, 1994) addresses the hypothetico side of the argument by stating that knowledge is antithetical and consists of inseparable ideal and empirical elements. He goes on to state that there is no permanent line to be drawn between *theory* (the ideal element) and *fact* (the empirical element) and states that a true theory is itself a fact, and can be used to form theories of even greater generality. Sutherland (1973) notes that Reichenbach addresses the deductive side of the argument by declaring that inductively predicated allegories express probabilistic behavior, such that an allegory may predict a phenomenon's behavior under the assumption that it will behave according to certain empirically-generated generalizations with some significant probability.

As discussed in the section *Challenges to Using the Case Study Method* in Chapter 3 on *Research Methodology*, there are those who significantly and substantively challenge the validity of the case study method. More specifically, the case study method has been considered to be a weak sibling as a research methodology based upon claims that the method does not have sufficient precision (i.e., quantification), objectivity, or rigor. In order to mitigate such criticism, a positivist case study design (Yin, 2003) has been selected. The use of a positivist case study design permits the researcher to: (1) study software development projects within their real-world context, and (2) invoke the objectivist framework's natural science model. The natural science model invokes

construct validity, internal validity, external validity and reliability as the key measures of design quality. These measures add significant relevance to this element of the research (Lee, 1989a, 1989b).

In summary, the mixed method research design was purposively selected in direct response to the research questions in Figure 1. Because no single method could adequately address each of the questions a mixture of a qualitative (subjective) and quantitative (objective) approaches was determined to best meet the goals of the research. The mixed method approach provided the research with significant strengths and limitations associated with the ontological assumptions and epistemological stances associated with each method. The limitations associated with each method were identified and accounted for in the research methodology and detailed design.

STUDY DELIMITATIONS

This section discusses two delimitations of the research. Delimitations are those ways in which the effort was constrained or narrowed to limit the overall scope of this specific research.

This research did not consider each of the detailed software development processes required to deliver a software product, but the superset of these processes where holistic, systems-based principles may be applied as part of an overall framework for improvement. As such, the focus of the research was not on how to develop software artifacts or recommendations for improvement or transformation of an individual development process or sub-system, but on the overall development process or system for developing software.

The research did not include completed software development projects from all applicable domains. In order to fully describe the large field of diverse software development projects the selection criteria included categories that were mutually-exclusive, exhaustive, and comparable (Gerring, 2001). The selection criteria were: (1) *project type* in which the software development project is delivering software to a commercial company, a government entity or a consortium of the two, and (2) *project duration* in which case the software development project had a specific duration, from start to finish, to complete delivery of the software. The research included projects in only two of the domain areas.

SUMMARY

This chapter has described how the study *developed and applied a systems-based framework for the analysis of software development project performance*. It has shown how the detailed research questions and higher level objectives support the purpose and fit within the structure of the overall inquiry in Figure 1. It has presented *systems science* as the theoretical foundation for the research and shown how *systems principles*, *systems theory*, *systems thinking*, and *systems practice* relate to both software project management and the larger field of software engineering. The chapter highlights the significance of five areas of the research to both the body of knowledge and the practice of software and systems engineering. It has provided bounds for the study and a discussion of limitations as well as delimitations.

By introducing the research purpose, objectives, and questions the chapter provides a smooth transition to the following chapter. The next chapter will frame the research setting within the literature and address how the research relates systems

principles and software development project management to software development project performance. Significant import will be given to the schema for the literature review, the breadth of the study, and exposure of gaps in the literature; highlighting the need for additional empirical research related to the structure of the inquiry.

CHAPTER II – LITERATURE REVIEW

The intent of this chapter is to establish the setting for the research; to frame it appropriately within the literature and address how the research relates systems principles to software systems development. The chapter presents the rationale and approach underlying the review and includes the search schema and breadth of the literature review. A detailed critique of the literature in each of the four focus areas was conducted and a concise report of the findings and themes present in the literature is presented. The final section summarizes the gaps in the research and the need for additional empirical research related to the research purpose and primary research questions that support the research design. The synthesized literature review serves as the database of empirical facts used in the inductive element of the research.

RATIONALE AND APPROACH UNDERLYING THE REVIEW

The focus of the literature review was to reduce the volume of information presented in the scholarly journals to that which was relevant and necessary for the research. The schema and breadth of the literature review ensured that the researcher was exposed to an appropriate range of ideas, concepts and theories. The literature review has additional meaning when the research includes induction and theory development. Lewis and Grimes (1999) state that:

Reviewing relevant literature enhances traditional induction by helping theorist's link emerging theory to extant work recognizing the influence of their own theoretical inclinations. (p. 678).

However, this was only one side of the initial boundary for the literature review. The other side of the boundary was the researcher's conceptual lens or worldview. This was the side that acted as a filter affecting the importance placed on the observations made by

the researcher and the decision to include or exclude individual journal articles and published manuscripts in the literature review.

The researcher was tasked with ensuring that the underlying assumptions and boundaries of the literature review were made explicit. This had added significance because the outputs of this early stage of the research were the factual information used in the induction in the first element of the research. The schema for and the scholarly journals included in the literature review were explicitly stated. However, the rationale used to discriminate journal articles and published manuscripts for the induction was problematic and required explicit guidelines that addressed inclusion and exclusion.

Because an inductive method was used, in this case William Whewell's *Discoverers' Induction* (1858), a great deal of importance was placed on how Whewell viewed and treated *the facts* in an induction. Whewell's epistemology required that certain ideal conceptions, as well as facts, are necessary materials of knowledge (L.J. Snyder, personal communication, May 8, 2006). Whewell stated that ideas or conceptions are crucial in the discovery of empirical laws and addressed them as follows (Snyder, 2006):

1. Conceptions are involved in the very process of perception; Whewell claimed that all perception is *conception laden*.
2. Conceptions are necessary to form theories from facts in the process of colligation. The appropriate conception must be superinduced upon, or applied to, the facts in order to bring the facts together under a general law.
3. Some conceptual framework is necessary in order to serve as a guide in the collection of empirical data. That is, we cannot and do not collect facts blindly, without some theory or conception guiding our choices for what to include and exclude from the collection of data.

The third point was of major significance. Whewell was stating that the idea or conception, in this case *how does systems theory apply to the analysis of software development project performance*, serves as a guide when considering what facts (i.e. journal articles and published manuscripts) to include or exclude in the induction.

The explicit rationale for inclusion and exclusion of journal articles and published manuscripts in the synthesized literature review must ensure that the results include “. . . facts that are both theory-laden and value-laden.” (Guba & Lincoln, 1994, p. 105) The following guidelines were invoked as explicit guidance:

1. The researcher rigorously reviewed the scholarly journals in Table 1 searching for articles on: (a) systems principles, (b) systemic improvement frameworks for software development, (c) the application of systems principles to software development, and (d) software development project performance.
2. Journal articles from topical areas (a) through (d) were evaluated against the conception *how does systems theory apply to the analysis of software development project performance*.
3. The researcher used his academic knowledge and training in systems and software engineering to ensure that journal articles and published manuscripts are of high-quality and contain sufficient empirical rigor to warrant selection and inclusion in the synthesized literature review. It is important to note that for inductive research *the researcher* is the instrument of the study.

Finally, prior to the start of the actual induction, an expert review was conducted to verify that the information synthesized in the literature review was sufficient and appropriate. The use of an expert, outside of the researcher, was intended to decrease

research risk by ensuring that the information selected by the researcher was sufficient to provide a firm foundation for the induction.

LITERATURE SEARCH SCHEMA

The multi-disciplinary nature of software development project management required the inclusion of a variety of scholarly literature from the management, information systems, software, and systems fields of study. The literature search within these was focused on four areas: (1) systems principles, (2) systemic improvement frameworks for software development, (3) application of systems principles to software development, and (4) software development project performance.

Figure 3 depicts the schema for the literature review and how the wide body of knowledge was narrowed to support the development of a generalizable assessment framework for software systems development.

BREADTH OF REVIEW

The literature search included appropriate scholarly journals in the fields associated with the research purpose and primary research questions. A clear distinction was made between published literature that was founded on empirical research and that which was published with no empirical basis, with the latter excluded from the review. As stated, the sources included in the schema were from a wide variety of disciplines and include the scholarly journals listed in Table 1.

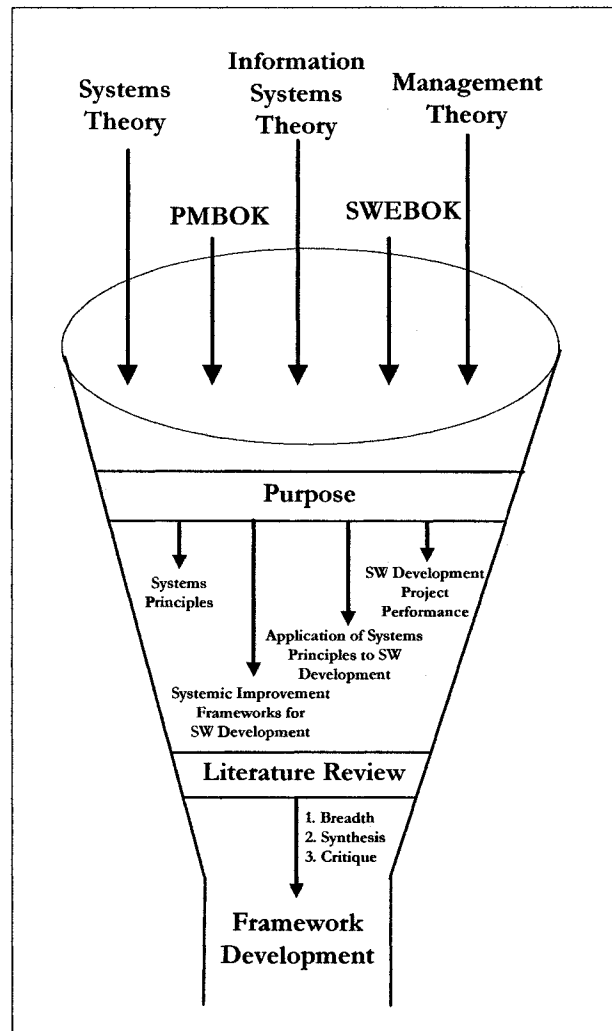


Figure 3: Schema for Literature Review

The scholarly journals selected for the literature review were included to describe the theoretical perspectives and previous research findings related to the research purpose. Table 1 includes the primary scholarly journals in management, software, information systems, and systems. Journal articles related to the research purpose were classified within four areas: (1) systems principles, (2) systemic improvement frameworks for software development, (3) application of systems principles to software development, and (4) software development project performance. A scholarly review and a concise report of the findings and themes present in the literature was conducted.

The synthesis of the literature in each of the four primary threads of the research purpose is presented in the following sections.

Discipline	Journal Title	ISSN	Article Retrieval Source
Dissertations	<i>Doctoral Dissertations</i>	N/A	Digital Dissertations
Management	<i>International Journal of Project Management</i>	0263-7863	Science Direct
Management	<i>Journal of Operations Management</i>	0272-6963	Science Direct
Management	<i>Engineering Management Journal</i>	1042-9247	ABI/INFORM Global (Proquest)
Management	<i>Project Management Journal</i>	8756-9728	ABI/INFORM Global (Proquest)
Management	<i>European Journal of Operational Research</i>	0377-2217	Science Direct
Management	<i>European Management Journal</i>	0263-2373	Science Direct
Management	<i>International Journal of Operations & Production Management</i>	0144-3577	ABI/INFORM Global (Proquest)
Management	<i>Journal of General Management</i>	0306-3070	Business Source Premier (EBSCO)
Management	<i>Harvard Business Review</i>	0017-8012	Business Source Premier (EBSCO)
Management	<i>Management Science</i>	0025-1909	Business Source Premier (EBSCO)
Software	<i>Communications of the ACM</i>	0001-0782	ACM Digital Library
Software	<i>Journal of the ACM</i>	0004-5411	ACM Digital Library
Software	<i>IEEE Computer</i>	0018-9162	IEEE Digital Library
Software	<i>IEEE Transactions on Software Engineering</i>	0098-5589	IEEE Digital Library
Software	<i>IEEE Software</i>	0740-7459	IEEE Digital Library
Information Science	<i>Decision Sciences</i>	0011-7315	ABI/INFORM Global (Proquest)
Information Science	<i>Decision Support Systems</i>	0167-9236	Science Direct
Information Science	<i>MIS Quarterly</i>	0276-7783	Business Source Premier (EBSCO)
Information Science	<i>Information and Management</i>	0378-7206	Science Direct
Information Science	<i>Journal of Management Information Systems</i>	0742-1222	Business Source Premier (EBSCO)
Information Science	<i>Information Systems Research</i>	1047-7047	Business Source Premier (EBSCO)
Information Science	<i>European Journal of Information Systems (old Journal of Applied Systems Analysis 1969-1991)</i>	0960-085X	ABI/INFORM Global (Proquest)
Systems	<i>Journal of the Operational Research Society</i>	0160-5682	JSTOR
Systems	<i>Journal of Systems and Software</i>	0164-1212	Science Direct
Systems	<i>Kybernetes: The International Journal of Systems & Cybernetics</i>	0368-492X	Emerald Fulltext
Systems	<i>Systems Research and Behavioral Science</i>	1092-7026	ABI/INFORM Global (Proquest)
Systems	<i>Systemic Practice and Action Research</i>	1094-429X	ABI/INFORM Global (Proquest)
Systems	<i>Crosstalk</i>	0000-0000	www.stsc.af.mil/crosstalk

Table 1: Scholarly Journals in Literature Review

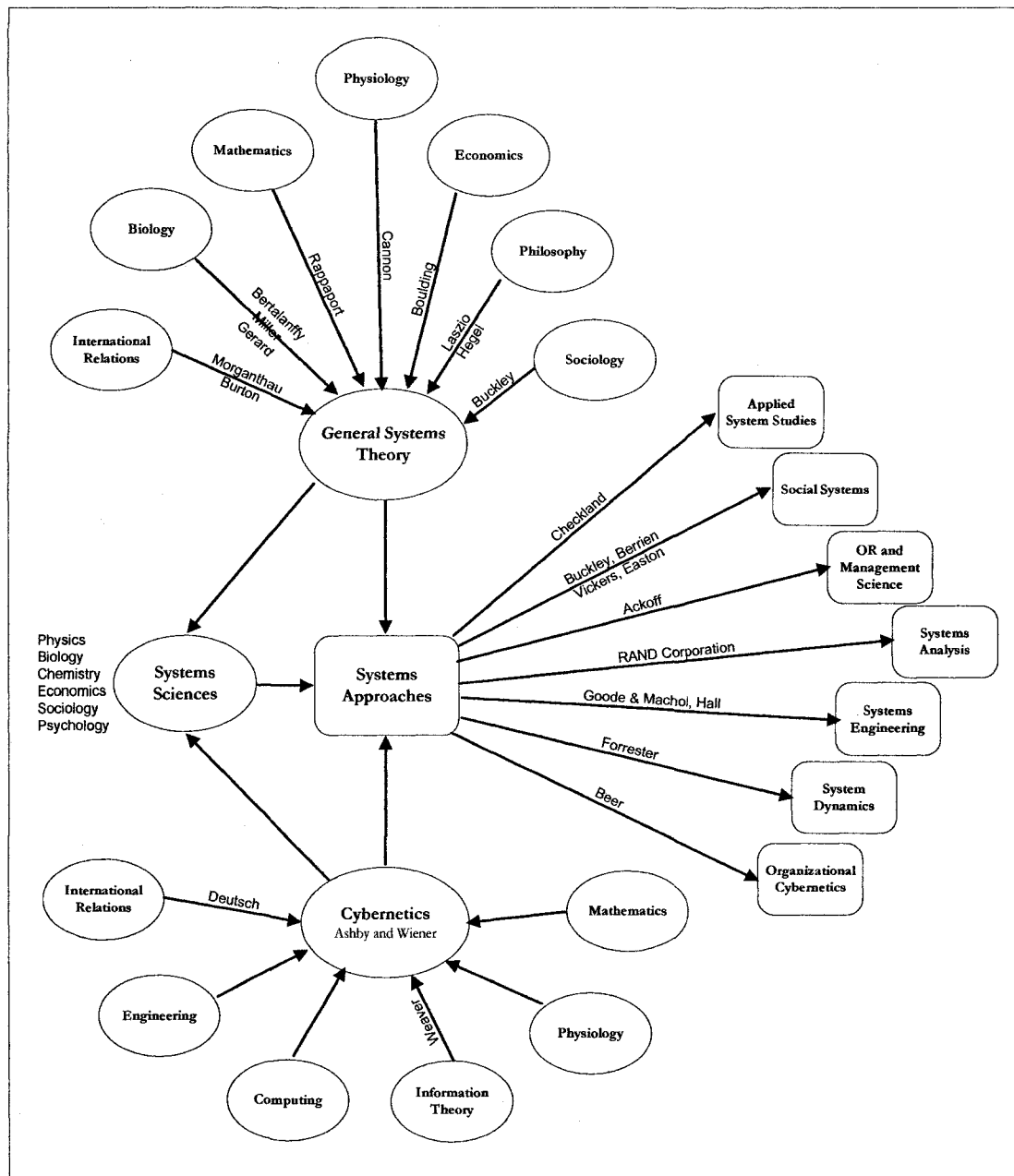


Figure 5: Systems Approaches

Modified from Flood, R. & Carson, E. (1993). *Dealing with Complexity: An Introduction to the Theory and Application of Systems Science* (2nd ed.). New York: Plenum Press, p. 7.

Bertalanffy, a biologist along with Anatol Rapoport, a bio-mathematician; Ralph Gerard, a physiologist; and Kenneth Boulding, an economist formed the Society for General Systems Research in 1954 to "... further the development of theoretical systems which are applicable to more than one of the traditional departments of knowledge."

(Bertalanffy, 1968, p. 15) Bertalanffy's unchanging vision was that "... there would arise as a result of work in different fields a high-level meta-theory of systems, mathematically expressed." (Checkland, 1993, p. 93) However, this has not been the case. Because general systems theory is, by design, very *general*, it has suffered from a lack of content and not achieved the status predicted by its major proponent, Bertalanffy. Progress in the systems movement has come from those who have used systems ideas to solve problems; this is applied systems science.

Six of the systems approaches in Figure 5 contain methods and techniques that have been applied to software development on real-world projects. The approaches include the branches on: (1) Applied Systems Studies, (2) Operations Research and Management Science, (3) Systems Analysis, (4) Systems Engineering, (5) System Dynamics, and (6) Organizational Cybernetics.

Applied Systems Studies

One of the major figures in applied systems science is Peter Checkland. Checkland (1993) states that when we think about systems we must "... make conscious use of the particular concept of wholeness captured in the word *system* as a means to order our thoughts." (p. 4) This type of thinking, focused on the world outside of the self by means of the concept of a system, is where the journey toward what Checkland calls *systems thinking* begins.

Systems Thinking:

Understanding the concept of wholeness captured so elegantly in the word *system* is the first and most essential step in applying systems principles. The words *holism* and *systemic*, so frequently used by systems practitioners, are founded on this basic understanding. In order to apply the full range of systems principles a holistic language,

a language of systems, interaction, and design exists to help understand and frame the wide variety of problems that surround systems as a matter of course. Gharajedaghi (1999) elegantly states:

The systems language, by necessity, will have two dimensions. The first will be a framework for understanding the beast, the behavioral characteristics of multi-minded systems. The second will be an operational systems methodology, which goes beyond simply declaring the desirability of the systems approach and provides a practical way to define problems and design solutions. (p. 26).

Systems Thinking is promoted by *Systems Theory* which is founded on *Systems Principles*. The characteristics of multi-minded systems are explained in *systems principles*, which are a scientific hierarchy of laws, principles, theorems, hypotheses and axioms associated with systems. A large number of the systems principles reported by Skyttner (2001) and Clemson (1991) could be applied to this research. However, only a few have direct applicability to the research question regarding the application of systems theory to software development project performance. The rationale for the inclusion of individual systems principles has been addressed during the synthesis discussion of the branch of systems approaches (i.e. Applied Systems Science, Operations Research and Management Science, Systems Analysis, Organizational Cybernetics) in which each principle is most widely applied.

There is one paper in the literature review that addressed the application of systems science in software development. West (2004) proposed the use of systems thinking when addressing software development process improvement. He stated that people (social systems), tools and technology (environmental systems), and policies and processes (process systems) on a software development project are an integrated system of systems and must be treated using a systemic approach.

Principle of Holism:

The modern protagonist for system holism was Jan Christian Smuts [1870-1950].

Smuts (1926) made the following basic observation about holism:

Both matter and life consist of unit structures whose ordered grouping produces natural wholes which we call bodies or organisms. This character of wholeness meets us everywhere and points to something fundamental in the universe. Holism (from ὅλος = whole) is the term here coined for this fundamental factor operative towards the creation of wholes in the universe. Its character is both general and specific or concrete, and it satisfies our double requirements for a natural evolutionary starting-point. (p. 86)

Smuts believed that wholes and wholeness should not be confined to the biological domain but are extendable to both inorganic substances and the highest manifestations of the human spirit. His treatise was concerned with the relationship between holism and evolution but there are other valuable concepts proposed in his work. Perhaps his most clearly stated concept applicable to systems is as follows (Smuts, 1926):

It is very important to recognize that the whole is not something additional to the parts: it is the parts in a definite structural arrangement with mutual activities that constitute the whole. The structure and the activities differ in character according to the stage of development of the whole; but the whole is just this specific structure of parts with their appropriate activities and functions. (p. 104)

For applied systems studies a complete and thorough understanding of this statement is an entering argument to practice. The import placed upon the concept of holism cannot be overemphasized, particularly for Cartesian reductionist thinkers whose history of practice has been limited to sub-system elements. Faced with modern complex systems holism is the starting point when addressing the wider concerns of the larger system.

Principle of Complementarity:

Niels Bohr [1885-1962], the 1922 Nobel Laureate in Physics, felt that the classical and quantum mechanical models were two complementary ways of dealing with physics, both of which were necessary (1928). He expressed this in what he called his Principle of Complementarity (Bohr, 1937):

The apparently incompatible sorts of information about the behavior of the object under examination which we get by different experimental arrangements can clearly not be brought into connection with each other in the usual way, but may, as equally essential for an exhaustive account of all experience, be regarded as 'complementary' to each other. (p. 291)

Bohr was stating that certain physical concepts are complementary. If two concepts are complementary, an experiment that clearly illustrates one concept will obscure the other complementary one. For example, an experiment that illustrates the particle properties of light will not show any of the wave properties of light. This principle also implies that only certain kinds of information can be gained in a particular experiment. Some other information that is equally important cannot be measured simultaneously and is lost.

Complementarity is a very valuable systems principle. Knowledge that there is no single correct or incorrect perspective of a software development system and that all perspectives reveal some truth about the system provides knowledge that permits the researcher to apply systems theory and systems principles to the first research question: *How does systems theory apply to the analysis of software development project performance?* Invoking the principle of complementarity allows the researcher to include the full spectrum of possible system solutions from *Systems Practice* as alternatives during the inductive framework development. Clemson (1991) notes that “. . . it is a mistake to inquire as to which perspective is *right*. The proper question is *given our current practical purpose, which perspective is most useful?*” (p. 206)

Principle of Satisficing:

Herbert A. Simon [1916-2001], the 1978 Nobel Laureate in Economics, conducted an investigation where he casts serious doubt on the usefulness of economic and statistical theories of rational behavior as bases for the characteristics of human and other organismic rationality. Simon (1956) proposes the following:

Both from these scanty data and from an examination of the postulates of the economic models it appears probable that, however adaptive the behavior of organisms in learning and choice situations, this adaptiveness falls far short of the ideal of 'maximizing' postulated in economic theory. Evidently, organisms adapt well enough to "satisfice," they do not, in general, 'optimize'. (p. 129)

Simon believes that a great deal can be learned about rational decision making by observing, at the outset, two key points: (1) the limitations upon the capacities and complexity of the human decision-maker; and (2) taking into account the fact that the environments to which human decision-makers adapt possess properties that permit further simplification of the choice mechanisms. He points out some suggestions as to the kinds of *approximate* rationality that might be employed by decision-maker possessing limited information and computational facilities (1955).

Much of Simon's treatise on rational decision making is based on his desire to construct a simple mechanism of choice that would suffice for the behavior of a decision-maker confronted with multiple goals. He goes on to state (1956):

Since the organism, like those of the real world, has neither the senses nor the wits to discover an 'optimal' path – even assuming the concept of optimal to be clearly defined – we are concerned only with finding a choice mechanism that will lead it to pursue a 'satisficing' path, a path that will permit satisfaction at some specified level of all its needs. (p. 136)

Simon's Principle of Satisficing has many direct applications in the world of applied systems science. Ackoff (1974) provides a real-world example:

In many cases models are constructed for which algorithms cannot be found; that is no systematic way of extracting optimal solutions from them is available. Such models, however, can be put to effective use. They can be used to 'compare' alternative solutions that are proposed by the decision maker. Thus a manager and a model can engage in a dialogue through which the decision maker can systematically improve a proposed solution to a problem even if he can't find the best one possible. The decision maker can try and err or experiment with the model rather than the real world, thereby accelerating and reducing the cost of the learning process. (p. 10)

Satisficing is an emancipating systems principle. Knowledge that there is no single absolute optimized solution for the complex software development system allowed the researcher to abandon the quest and find a solution that permitted overall system satisfaction. Invoking the Principle of Satisficing provided the researcher with the rationale to search for and invoke a systems framework where good enough was not only conceivable, but acceptable. At this point it is important to note that Simon's work on rational choice (1955, 1956, 1979) serves as the foundation for the work in human decision making done by 2002 Nobel Laureate (Economics) Daniel Kahneman (2003a, 2003b) and his partner Amos Tversky [1937-1996]. Their work will be addressed in more detail in the section on systems analysis.

In summary, systems thinking, and the systems principles that address holism, complementarity and satisficing provided insight about how to view software project performance. The consideration of these systems principles, and the structured systemic method of *systems thinking*, are considered as part of the systemic framework.

Operations Research and Management Science

The field of Operations Research and Management Science has, as one of its principal figures, Russell L. Ackoff. Ackoff is responsible for making many contributions to management science, one of which has a direct bearing on the research.

Principle of Systems Context:

Russell Ackoff (1974) uses the terms machine-age and systems-age to refer to eras that were concerned with two different types of systems. The machine-age was concerned with simple systems, and the systems-age is concerned with complex systems. Table 2 contrasts the most basic characteristics of the machine and systems ages.

	<i>Machine Age</i>	<i>Systems Age</i>
<i>Characteristic</i>	Simple System	Complex System
<i>Boundary</i>	Closed	Open
<i>Elements</i>	Passive parts	Purposeful parts
<i>Observable</i>	Fully	Partially
<i>Method of understanding</i>	Scientific method of reductionism	Cannot use reductionism

Table 2: Ackoff's Machine-Age and Systems-Age Characteristics

Ackoff (1979a) recognized that the traditional reductionist engineering methods would be incapable of coping with what he termed the *messy* situations present in human organizational endeavors. Ackoff coined the concept of a *mess* and *messes* in 1979 when he used the idea in two papers where he was arguing that operational research was *passé* and that a more holistic treatment of problems was required and that a wide variety of disciplines would be necessary (1979a, 1979b). Ackoff's (1979a) definition of a mess and messes is worthy of review:

Because messes are systems of problems, the sum of the optimal solutions to each component problem taken separately is not an optimal solution to the mess. The behavior of the mess depends more on how the solutions to its parts interact than on how they interact independently of each other. But the unit in OR is a problem, not a mess. Managers do not solve problems, they manage messes. (p. 100)

Ackoff uses the systems principles of *Hierarchy* and *Emergence* to describe the types of problems facing problem solvers in the real world. The bottom line is that *real world* complex systems problems must include a definition of human activity in the development of the contextual framework for the problem. For Ackoff (1979a), context

was the essential element that modern systems problem solvers would need to include in each problem definition if complex systems were to be understood. He argued that the utility of operations research had been diminished because most of the established techniques were unable to account for the real-world complexity present in *systems-age* problems. Burrell & Morgan (1979) support Ackoff's contention, stating:

Mechanical models of social systems, therefore, tend to be characterized by a number of theoretical considerations and are thus of very limited value as methods of analysis in situations where the environment of the subject is of any real significance. (p. 61)

In short, the methods and techniques of traditional operations research are “. . . mathematically sophisticated but contextually naïve and value free.” (Hughes & Hughes, 2000, p. 10) Ackoff's work established the need for a clear understanding of specific system context as fundamental to understanding and analyzing complex systems and complex system problems across all of the different systems-based disciplines.

Many of the most significant problems facing the software engineering community are a product of the complexity associated with developing software systems. Most of the problems can only be resolved or addressed by understanding the complex socio-technical elements of the development environment within which the software systems are developed. According to Quade and Miser (1985)

Many of society's problems emerge from processes associated with structures that combine people and the natural environment with various artifacts of man and his technology; these structures can be thought of as systems. Such problems, and the systems of which they are aspects, abound in modern society. (p. 1)

The problem system includes “. . . the social and technical elements, their formal and informal relationships, emergent patterns, and the unique context of the problem.” (Keating, Kauffmann & Dryer, 2001, p. 773) Modern complex systems require a

systemic approach, one that includes two central ideas (Keating, Kauffmann & Dryer, 2001, p.773):

1. Problems cannot be isolated from the system that is producing the problematic behavior; and
2. The problem system cannot be understood independently from the context within which it is embedded

“The use of systems principles requires a holistic perspective of the system under investigation as part of all *systems-based* problem solving methods and requires the systems analyst to use a systems view to understand *problem systems in context*.”

(Keating, Kauffmann & Dryer, 2001, p. 773)

In summary, systems context is another systems principle applicable to software development project management. Knowledge that the problems associated with software development project performance can not be addressed apart from the surrounding context required the researcher to consider software development projects as complex social and technical system operating in real-world environments. Invoking the principle of systems context challenged the researcher to include the full contextual environment that surrounds real-world software development projects.

Systems Analysis

During the Second World War the American military used large numbers of scientists and engineers to help solve complex logistical and strategic bombing problems related to the war effort. This field was termed *systems analysis* and was principally concerned with the analysis of small-scale systems and the interactions of components within those systems. The distinguishing characteristic that applied to systems analyses of that period was that the systems inquiry was undertaken to help decision makers

identify a course of action and make economic decisions. Checkland (1978) defines systems analysis as:

The systematic appraisal of the costs and other implications of meeting a defined requirement in various ways. (p. 107)

Systems analysis was closely related to operations research because it used many of the techniques, mathematical models, and practitioners from the field of operations research to complete the systems analysis. Many of these efforts made significant contributions to the philosophy and techniques of what was then called Operations Research.

One systems analysis project was the source of a systems principle that has a direct bearing on the research and that is the Principle of Suboptimization.

Principle of Suboptimization:

The word *sub-optimize* was coined by Charles. J. Hitch [1910-1995] while working at the RAND Corporation (RAND, 1952). Hitch's study, one of the classics of operations research, was performed during World War II on the optimum size of a merchant-ship convoy. The problem was the sinking of Allied merchant ships by groups of German submarines known as wolfpacks. Hitch found that the ratio of submarines sunk to merchant ships sunk, varied as the square of the size of the convoy. The recommendations of this study were put into effect and the number of merchant-ship sinkings decreased dramatically, contributing importantly to the winning of the Battle of the Atlantic, and consequently to the winning of the war (Machol, 1965). Hitch stated that although his analysis of the wartime problem (where he showed that convoys should be made as large as possible) produced a final answer that was approximately correct, it was arrived at for the wrong reasons. He argued that the reasons were wrong because too much emphasis was placed on the sub-system, namely the battles between an individual

convoy and a submarine wolfpack, rather than on the wider system, that of winning the Battle of the Atlantic or more importantly still, optimizing an even wider system, namely winning the war (Hitch, 1953).

Hitch reported that his own study was guilty of suboptimization, not because the study was faulty, but because the study was unable to encompass the larger worldview of the problem. The relative point of view, within the hierarchy of systems, forms the framework for suboptimization. Hitch suggested “. . . that approach involves the analysis of relations between sub-optimizations at lower and higher levels. Operations researchers must understand the general characteristics of the higher level optimization if they are to exercise good judgment in the selection of criteria at the lower levels - that is, if the sub-optimizations are to contribute even indirectly to the high level objectives.” (Hitch, 1953, p. 98)

The principle of suboptimization was the most important systems principle associated with this research. The principle of sub-optimization permitted the researcher to state that optimizing each software development subsystem or process independently would not in general lead to a system optimum, or more strongly, improvement of a particular software development subsystem or process may actually worsen the overall development system. The work by Hitch (1953; RAND, 1952) touches on two additional elements from the field of operations research and systems analysis; decision makers within problem contexts and human cognitive bias in decision making.

In the first area, Jackson and Keys (1984) address the importance of decision makers within problem contexts and how decision makers greatly influence the type of solution needed and ultimately, the problem solving methodology required to reach an

adequate solution. The criterion Jackson and Keys used to classify decision makers is whether they are unitary, pluralistic, or coercive with respect to their objectives.

Decision makers are classified as *unitary* if they all agree on a common set of goals for the system and make their decisions in accordance with these goals. A set of decision makers is *pluralistic* if they cannot all agree on a common set of goals and they make decisions which support differing objectives, but an accommodation or compromise can be reached upon which all agree. Decision makers are classified as *coercive* if decisions are achieved by the exercise of power and domination of one or more groups over others. "In the case where coercive behavior is demonstrated it is impossible for any compromise solution to bring about a genuine accommodation among the parties." (Jackson, 1990, p. 658) In a later work, Jackson (1991) has changed the classification heading from decision makers to participants and, he notes that the terms unitary, pluralistic, and coercive (or radical) are common in the industrial-relations literature for describing the relationship among the various stakeholders with an interest in organizations.

In the second area, human cognitive bias in decision making, Tversky & Kahneman (1974) have shown that people making judgments under uncertainty "... rely on a limited number of heuristic principles which reduce the complex tasks of assessing probabilities and predicting values to simpler judgmental operations. In general, these heuristics are quite useful, but sometimes they lead to severe and systematic errors." (1974, p. 1124) Tversky & Kahneman (1971) also identified human limitations when processing statistical information and dealing with small sample sizes. Kahneman, Slovic & Tversky (1982) have assembled the seminal papers on judgment under

uncertainty, in a volume of the same name that provides extensive coverage of heuristics and bias.

The value of this work is the recognition that cognitive and perceptual bias exists, regardless of motivational factors, in all human decision making. Faced with this condition, Kahneman and Tversky (1979) challenged the previously accepted notions of decision making and the associated principal theory, *utility theory*. Their alternative theory, called *prospect theory*, addressed a number of violations of classical rationality that they had uncovered in their earlier empirical studies. They cite two major inconsistencies in *utility theory* that *prospect theory* is designed to address (1979, p. 263):

1. *People underweight outcomes that are merely probable in comparison with outcomes that are obtained with certainty. They found that this irrational human behavior, which they call the 'certainty effect', contributes to risk aversion in choices involving sure gains and to risk seeking in choices that are sure losses.*
2. *People generally discard components that are shared by all prospects under consideration. This tendency, called the 'isolation effect', leads to inconsistent preferences when the same choice is presented in different forms.*

In order to address these inconsistencies *prospect theory* views decision making under risk as a choice between prospects or gambles. To overcome the cognitive limitations Kahneman and Tversky (1979) assign values to intermediate gains and losses rather than to final assets or losses and replace the probabilities with decision weights. Prospect theory has provided a number of very tangible benefits in econometrics and found wide acceptance in the field of economics.

It is easy to imagine how the application of prospect theory and the knowledge associated with human cognitive bias can be applied to the engineering of a large software development projects and the supporting project management systems. As such,

this body of work is noteworthy and was considered when reviewing systems analysis methods.

At the same time that systems analysis problems were being solved, the need for many novel types of electronic gear for airborne use gave rise to a wide variety of component devices, popularly known as *black boxes*. These were ingenious devices, but their application in terms of the entire system of which they were merely parts was a matter of improvisation (Engstrom, 1957). Inevitably, many of the engineers and scientists working on these *black boxes* were required, by necessity, to look ahead to the ultimate goal – the system. The need to address a complete system, and the increasing complexity of systems gave rise to a new discipline; *systems engineering* (Roy, 1960).

Systems Engineering

The definition of *Systems Engineering* has evolved since its first formal definition in the 1960s. Table 3 shows the evolution of the definitions to include the words *complex* and *customers* in the formal definition. These words have been included because systems engineers must include *customers* and the *messy* contextual situations that real-world systems engineering problems present when they define problems for solution. Complex man-made systems require a holistic, systemic understanding of both the technical problem and the contextual framework present in order to arrive at satisfactory solutions. The evolution of systems engineering and its establishment as a separate discipline is recounted by the following observation (Hughes & Hughes, 2000):

After World War II, a systems approach to solving complex problems and managing complex systems came into vogue among engineers, scientists, and managers. In 1964, the 'Engineering Index' had no entry for 'systems engineering' and only two pages for 'operations research,' both variations upon a systems approach. By 1969 the number had jumped to eight pages of citations for 'systems engineering' and to ten for 'operations research'. (p. 1)

Definition	Source
...engage in the analysis of complex man and machine systems or one may also say man and machine operations, utilize multi-discipline teams, employ the scientific method, emphasize the "whole system" rather than the component approach . . .	Flagle, Huggins & Roy (1960, p. 23)
The design of systems in which the output is a set of specifications suitable for constructing a real system out of hardware.	Machol (1965, p. 1-4)
. . . the overall problem of systems engineering is composed of two parts, one being the systems engineering associated with the way that the operating system itself works and the other with the systematic process of performing the engineering and associated work in producing the operating system.	Chestnut (1967, p. 12)
The set of activities that together lead to the creation of a complex man-made entity and/or the procedures and information flows associated with its operation.	Checkland (1993, p. 138)
An interdisciplinary collaborative approach to derive, evolve, and verify a life-cycle balanced system solution which satisfies customer expectations and meets public acceptability.	IEEE 1220 (1998, p. 11)
The function of systems engineering is to guide the engineering of complex systems.	Kossiakoff and Sweet (2003, p. 3)
An interdisciplinary approach and means to enable the realization of successful systems.	INCOSE (2004, p. 12)

Table 3: Evolution of the Systems Engineering Definition

Systems engineering and its association with software have followed a curious path.

Systems engineering concepts and principles were routinely applied to the development of complex software systems throughout the late 1960s and early 1970s. However, not long after the software community coined the term *software engineering* (Naur & Randell, 1969) a new independent field called *software engineering* was created.

Software Risk Management author Dr. Robert Charette (1989, 1991) points out that the application of systems engineering concepts and principles were abandoned by the software community because of the perception [although incorrect] that systems engineering was focused only on hardware (personal communication, May 16, 2006).

Numerous systems and software experts feel that the abandonment of the systems approach and the corresponding failure to address software from a holistic, systemic perspective has contributed to the numerous problems associated with the development and operation of software systems (R.N. Charette, personal communication, May 16,

2006). However, systems engineering is beginning to make a comeback in the software engineering community. The following recent events lend credence to this statement (Schaaf, 2005):

The Software Productivity Consortium changed its name to Systems and Software Consortium. Likewise, the IEEE's Software Engineering Standards Committee added the word systems to its name to become the Software and Systems Engineering Standards Committee. The US Department of Defense's Software Technology Conference morphed into the Systems and Software Technology Conference. (p. 104)

In addition, the IEEE and the ISO/IEC are actively engaged in an effort to integrate their software and systems engineering standards (Moore, 2006). One of software engineering's most prolific and respected researchers has called for the unification of software and systems engineering (Boehm, 2000, 2006). Thayer (2002) states:

The application of systems engineering principles to the development of a computer software system produces activities, tasks, and procedures called software systems engineering, or SwSE. (p. 68)

Coallier (2003) describes the activities associated with the international standardization in software and systems engineering. Mathieu (2002), in his article on the *Top-Down Approach to Computing* opines that

The systems engineering approach is integral to large-scale information technology projects common in modern business organizations. (p. 139)

Rozenblit and Kumar (1997) describe how computer systems should be synergistically developed using innovative solutions and engineering methodologies that address:

- ✓ Complexity,
- ✓ Model-based engineering, and
- ✓ Process management

All three of these elements are central concepts in systems engineering.

Sage has made significant contributions in the application of systems engineering to software (Sage & Palmer, 1990 and Sage, 1995). His contribution has been to address the engineering of software and information technology through the use of established lifecycle management techniques from systems engineering. He states that (Sage & Palmer, 1990)

... the major problems associated with the production of trustworthy software are more concerned with the 'organization and management of complexity' than with direct technological concerns that affect individual programmer productivity. (p. 8)

His view of *software engineering in the large*, or software systems engineering, is concerned with the processes that invoke *systems management* for software development projects. These include the process of software design, production and maintenance required to "... ensure client needs are satisfied in an efficient, effective, and otherwise productive manner." (Sage & Palmer, 1990, p. 9)

Table 3 shows how systems engineering has evolved to include *complex* systems and *customers* in the formal definition of its domain. However, during this evolution the two principal educational texts (Blanchard & Fabrycky, 1998; Kossiakoff & Sweet, 2003) on systems engineering have eliminated topics on the fundamental concepts and properties associated with systems that were included in the foundation texts (Goode & Machol, 1957; Hall, 1962). The modern texts include few *soft* topics to encompass *customers* and Ackoff's *messy* situations that real-world systems engineering problems present. The techniques addressed in the two texts are prescriptive in nature and focus principally on the management of the systems life cycle. Systems engineers require access to solutions based upon formal principles, methodologies, and supporting techniques or methods. Complex man-made systems require a holistic, systemic

understanding of both the technical problem and the contextual framework present in order to arrive at satisfactory solutions.

Complexity:

Complexity is present, to some extent, in every system. The ability to observe, understand, and apply complexity has been addressed by Warren Weaver [1894-1978] (1948) and Simon (1962). In an interesting epistemological argument entitled *Liberating Systems Theory*, Robert Flood (1990) provides a section that includes paradigmatic interpretations that relate systems and complexity. These are at least three ways to relate system and complexity (Flood & Carson, 1993, p. 34):

1. *Systems are real and tangible things. They are groups of elements related to the whole. Boundaries are easy to identify. Complexity is measured in terms of the number of elements, number of relationships and attributes of these such as linearity, symmetry, and nonholonomic constraints. Complexity and system are therefore synonymous in a real sense. System is prime.*
2. *Systems are real but are difficult to access and know. Their reality is known through interpretations. Complexity and systems are not synonymous because people factors such as interpretation muddle system identification. Neither system nor people is prime.*
3. *The realness and existence of systems is questioned. 'Systems' are people's actions and the social rules and practices that define those actions. Systems therefore are contingent on there being people. Take away the people and systems do not exist. Complexity and system have no clear relationship other than system being a structure through which we organize our thoughts about the world. People are prime.*

The software engineering community can benefit from the application of complexity theory in order to gain insight and understanding of the complexity present in their software systems and the management of projects that deliver these systems. "Current mainstream computer and (often) communications structures contain significant unnecessary complexity. This complexity stems from two main factors: (1) The provision of too many software functions and (2) the mapping of application functions

onto poor or inappropriate application platforms.” (Lawson, 1990, p. 120) While Lawson is speaking of the causes of some very specific technical complexities associated with the application software, there are additional complexities present in the system itself. Simon (1962) provides a simple definition of complexity that is focused on the system:

Roughly, by a complex system I mean one made up of a large number of parts that interact in a nonsimple way. In such systems, the whole is more than the sum of the parts, not in an ultimate, metaphysical sense, but in the important pragmatic sense that, given the properties of the parts and the laws of their interactions, it is not a trivial manner to infer the properties of the whole. (p. 468)

Sommer and Loch (2004) follow Simon’s definition and explicitly state that complexity has two separate dimensions: system size (number of variables) and interactions (correlation of neighboring points). While it is a relatively straight forward exercise to imagine the complexity of a physical system of N parts and K interactions a less obvious complexity is present. This is the complexity present in the contextual domain associated with the development of the software. This domain includes the problem solving process that is an essential part of overall systems engineering process. Mihm, Loch and Huchzermeier propose a mathematical model for use in complex projects (2003).

In summary, complexity is present in every systems engineering endeavor. The ability to observe and understand complexity is purposefully built into every systems engineering solution methodology as the precursor to attenuation. The software engineering community can benefit from the application of systems engineering methods in order to gain insight and understanding of the complexity present in their software systems and the management of projects that deliver these systems.

Entropy:

The entropy parameter, so common in most of the sciences and engineering, has utility in software engineering development projects as well (Campbell, 1982). When the software development cycle is modeled as a process with inputs and outputs it is easy to show how entropy comes into play. Figure 6 shows how the outputs of the process are less than the inputs and that the difference is entropy.

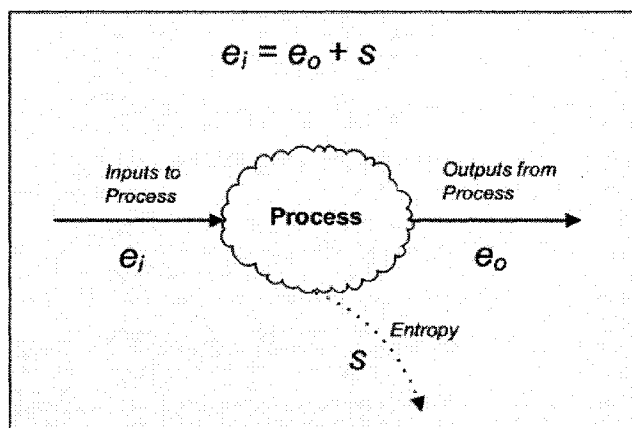


Figure 6: Simplified View of Entropy

This complies with the 2nd *Law of Thermodynamics* which states, in effect, that useful work out is always *less than* energy in. In nature entropy is related to things such as friction and resistance, heat loss, turbulence, and random motion and disorder. Jensen & Tonies (1979) label the "... corresponding effects in the human realm as Type 2 entropy, which is induced by causes that need not be accepted and include uncooperativeness, incoherence, confusion, and undirected or misdirected action." (1979, p. 50) They go on to state that (1979):

The primary objectives of software management are to identify the causes of entropy in the system and to control them effectively. (p. 55)

The removal of Type 2 entropy throughout the system is "... analogous to solving a system of simultaneous differential equations; the answer is found by addressing the total

intradependent system, not the individual parts. Local optimization of a software development sub-process may not (in fact, usually does not) contribute to a reduction of the total entropy integral.” (Jensen & Tonies, 1979, p. 53)

Model-Based Engineering and Process Management:

Modeling of real-world systems is an essential tool in systems engineering. The systems engineer uses models to represent actual real world systems that are a reflection of knowledge about the system. “Systems engineering has always had an interdisciplinary flavor and has been associated with modeling.” (Schaaf, 2005, p. 102) Models are used in order to both understand the system and communicate the system to others. Models can take many forms and can be iconic, symbolic, or analogous. (Flood and Carson, 1993) In all cases models are not systems, but representations of the systems they represent, which are real.

Wallace, Stockenburg & Charette (1987) provide a formal method for modeling in their Systems Engineering Methodology. This model is unique because it provides for formal communication as an integral part of the model. They stress that (1987):

The communication orientation can not be over emphasized, since one of the major reasons for system failures today is the lack of understanding of the system on the part of the individuals involved in its development. (p. 23)

The communications-oriented characteristics of this modeling technique strongly influence the behavior and utility of the unified methodology for developing systems.

Information modeling plays a central role during information systems development (Mylopoulos, 1998). Four *worlds* need to be understood and modeled during the development process (Jarke, Mylopoulos, Schmidt & Vassiliou, 1992):

1. ***Subject world.*** The system's application domain which consists of the subject matter for the information system (i.e. the world about which information is maintained by the system).
2. ***System world.*** The information system itself.
3. ***Usage world.*** The organizational environment within which the system is intended to function (i.e. the context).
4. ***Development world.*** The process that created the information system.

All of the information in the worlds needs to be represented in an effort to offer a comprehensive framework for information systems development. In order to accurately represent all of these worlds in a single model, the investigator must integrate a great deal of information and complexity, a task that may actually limit complete understanding.

Because software/information systems development is a human activity it can be modeled as a social or organizational system. The work of Dr. Bela Banathy (1996), a systems practitioner who focused on problems in social systems, offers additional insight. Banathy recommended a three-lens approach for two general types of models (1996).

In systems and design inquiry we work with both product and process models. Product models describe the outcome of an inquiry. Process models set forth the processes, the activities, by which to conduct the inquiry. (p. 51)

The three lenses can be considered to be types of views, each with a specific focus. The name of the lens and associated focus are as follows (Walton, 2004).

1. Systems-Environment Lens. *What is the system of interest?*
2. Function/Structure Lens. *What is the system about?*
3. Process Lens. *How does the system transform inputs to outputs?*

The Shinayakana Systems Approach (Nakamori & Sawaragi, 2000) is an environmental modeling approach based on systems thinking in China and Japan. “Shinayakana systems methodology stresses the adaptive learning and stimulation of intuition and creativity of people. One of the main tasks of systems analysts is to develop decision or thinking support systems that provide system models and system methods with which people can make decisions taking into account social aspects or human relations.” (Nakamori & Sawaragi, 2000, p. 182) This methodology appears to include both objective and subjective elements in the modeling methodology but is not detailed enough to warrant evaluation.

Another modeling methodology emerged from the literature search; the Boardman Soft Systems Methodology (BSSM). The BSSM was developed from Checkland’s and Scholes’ (1999) work in soft systems thinking and has been combined with action-learning principles. “The modeling theory includes systems concepts that govern the application of modeling techniques in order to make sense of the modeled situation. The modeling range in sophistication from *notation* level, through *template* to *framework*.” (Ramsay, Boardman & Cole, 1996, p. 33). This methodology has been used successfully to model a project management risk framework.

Finally, the Physical System Theory (PST) modeling methodology has been eliminated from consideration based on the literature review. “PST is based on the hard and analytic system philosophy. It follows the basic analysis-synthesis cycle of systems approach, i.e. dividing whole into parts and resynthesizing back into the whole to infer about the whole more rationally, systematically and objectively.” (Sushil, 2002, p. 502) PST (Sushil, 2002), as represented in the literature, fails to make adequate provisions for

inclusion of the rich contextual environment that surrounds a software development project in its models and is eliminated from consideration for use in constructing the framework.

In summary, systems engineering is the branch that addresses entire systems; with an emphasis on life-cycle management, from concept to retirement. System engineering includes methods and techniques that have evolved to address increasing system complexity. The use of the entropy concept, model-based engineering, and process management has introduced formal techniques that increase understanding and system knowledge. The systems engineering topics on complexity, entropy, and model-based engineering and process management were considered as part of the systemic framework.

System Dynamics

Computer pioneer Jay Forrester is the father of system dynamics. He developed systems dynamics after joining MIT's Sloan School of Management in the mid-1950s. Systems dynamics derived its roots from the principles of systems thinking. It applies systems thinking by holistically observing systems using comprehensive models of all facets of the system. Forrester was able to construct and automate these models as detailed and interdisciplinary causal loop diagrams. Forrester used his early models to simulate factory production systems using process flows and feedback loops that included critical resources such as information, materials, manpower, capital equipment and money. The success of Forrester's systems dynamics models was based upon three features (Edwards, 2000):

1. Models should be comprehensive.
2. Sorting out the structure and dynamics of a system using a computer model was the key to understanding.

3. Growth is a developmental stage and continued exponential growth is impossible.

Forrester's models and books on *Industrial Dynamics* (1958, 1961) and *Urban Dynamics* (1969) went on to receive wide acclaim and served to institutionalize this branch of systems science as contained in his influential book *World Dynamics* (1973).

A review of the literature shows that system dynamics has been used extensively to model various aspects of software development projects and include: software project scheduling (Abdel-Hamid & Madnick, 1983), software project staffing (Abdel-Hamid, 1989a), staff turnover (Abdel-Hamid, 1989b), cost/schedule tradeoff (Abdel-Hamid, 1990), software project control (Abdel-Hamid, Sengupta & Ronan, 1999), managerial turnover/succession on project performance (Abdel-Hamid, 1992), productivity improvement (Abdel-Hamid, 1996), and project management (Rodrigues & Williams, 1997). Ruiz, Ramos & Toro (2001) provide a simplified systems dynamics model in response to the text by Abdel-Hamid and Madnick (1991).

System dynamics “. . . assumes that the behavior of a system is principally governed by its structure, and flow structure is the most effective way of viewing an organization.” (Sushil, 2002, p. 518) Systems Dynamics (Abdel-Hamid, 1993 and Abdel-Hamid & Madnick, 1989, 1991), as represented in the literature, fails to make adequate provisions for inclusion of the rich contextual environment that surrounds a software development project, and as such was eliminated from consideration in construction of the framework for this research.

Organizational Cybernetics

W. Ross Ashby [1903-1972] has been described as the individual whose contributions to systems research are so important that systems researchers have rated

him as the single most influential person in the systems movement (Klir, 2001). Ashby introduced the important concept of variety in his most popular book, *An Introduction to Cybernetics*, in 1956. Ashby's concept of variety addressed the possible number of states that a system may be capable of exhibiting. In his 1956 text Ashby also formulated the *Law of Requisite Variety*, which is "... regarded by some as being as important to management as Newton's or Einstein's laws are to physics." (Jackson, 1991, p. 93)

Law of Requisite Variety:

Ashby's important law states (1956, p. 207):

"Variety can destroy variety."

This may require some explanation.

Variety is a measure of complexity that may be mathematically computed as the number of different possible system states that may exist. A trivial formula for Variety (Flood & Carson, 1993, p. 26) is:

$$V = Z^n$$

where V = Variety, n = number of system elements and Z = number of possible states of each element.

A simple example shows how *variety* may be used in systems development and control. Suppose there is a man-machine interface that is made up of 6 operators working on 6 different machines. This has 36 possible system elements. The machines can have only two (2) states: operational or non-operational. The formula for Variety can be used to calculate the system variety:

$$V = Z^n = 2^{36} = 68,719,476,736$$

So, for 36 system elements, and two states, there are sixty eight billion, seven hundred nineteen million, four hundred seventy six thousand, seven hundred and thirty six possible system states.

In order to positively control a system the *Law of Requisite Variety* (Ashby, 1956) requires that:

The variety of the controller must be at least as large as the variety of the system to be controlled. (p. 207)

This important law shows that the variety can quickly out-run the bounds of what is controllable, for even mildly complex systems. The real lesson is that *variety* is a function of the system inputs and outputs. For an unbounded system the *variety* is infinite. In order to control the system variety the system inputs and outputs must be controlled. Through the careful definition of the system boundary and the use of a regulator (input attenuators and/or amplifiers) systems designers may attempt to control the systems' *variety*. The Conant-Ashby Theorem states that (Conant & Ashby, 1970, p. 97):

Any regulator must model what it regulates.

For the systems engineer this means that a design must possess an amount of *variety* that is at least equal to the *variety* of the problem being addressed; and if it is to handle unexpected perturbations the design must have additional *variety*. This requires individuals or groups engaged in designing solutions to *messy*, real-world, complex systems problems to gain control over designs by making appropriate specifications in all the dimensions of the design, thus reducing the *variety*. The *Law of Requisite Variety* and the *Conant-Ashby Theorem* are important elements of the work in organizational cybernetics. Organizational Cybernetic methodologies, such as Beer's Viable Systems

Model (1979, 1981 & 1984) and the Organizations-as-Systems approaches (Cherns, 1976, 1987) are the principal methodologies found in the literature.

The Viable Systems Model (VSM):

The Viable Systems Model (VSM) was developed by Stafford Beer [1926-2002]. The central theme for the VSM is the essential organization of the system. The VSM is concerned with the definition of the system and what enables it to maintain its identity and to remain viable. It is a highly sophisticated organizational model based on cybernetic principles that is superior to the more traditional human relations models. Jackson states that the model's strengths are (1991, p. 118-120):

1. The approach is highly generalizable focusing on the definition of the system and the mechanisms that allow it to maintain itself in a viable state;
2. The approach is capable of dealing with organizations that have parts that are both vertically and horizontally interdependent;
3. It requires the analyst to address command and control within the system;
4. It is highly suitable as a starting point for the design of information systems;
and
5. It is highly effective as a diagnostic tool to make recommendations for improving the performance and efficiency of organizations.

"The principal shortcoming of the VSM is that it underplays the purposeful roles of individuals in organizations, which suggests an autocratic method, one that can be subverted for authoritarian use." (Jackson, 1991, p. 122) Espejo (2004) presents the VSM as an effective problem solving methodology for use in complex social systems. Yolles (2004) explores Beer's VSM from a system/meta-system viewpoint. Because the

VSM is a model, one that looks at enterprises using cybernetic principles in order to understand their viability, it has been included for review during the inductive development of the theoretical framework for software development.

Sociotechnical Systems (STS):

Sociotechnical systems theory sees organizations as pursuing tasks that can best be realized if their social, technological, and economic dimensions are jointly optimized, and if they are treated as open systems and fitted into their environment. (Jackson, 1991, p. 60)

Sociotechnical systems (STS) have both a technical subsystem, made up of the facilities, tools, equipment and knowledge necessary to execute processes in support of product development, and a social subsystem, which is made up of the people working on the processes. STS design is centered on the concept of positive integration between the technical and social subsystems in support of the larger production system. The literature adds a third subsystem, the environmental subsystem, which accounts for the contextual framework within which the design exists (Shani, Grant, Krishnan & Thompson, 1992). The literature has examples of where STS was used to implement an information system in an organization (Adman & Warren, 2000), in a high-technology production system (Jacobs, Keating & Fernandez, 2000) and in a software development firm (Shani & Sena, 1994).

Albert Cherno (1976), the father of modern Sociotechnical Design offered a number of basic principles he felt were essential in the design and/or redesign of organizations. In the update to his initial paper Cherno (1987) states that his goal has been:

In targeting engineers as designers of organizations, we sought to provide them with a new perspective, better understanding, and some guidelines so that they could better design organizations as social systems. (p. 154)

Based on his goal Cherns has authored an important systems principle.

Principle of Minimum Critical Specification:

Cherns' 2nd principle strives to limit the design role and prevent the age old problem of over design. It is entitled the Principle of Minimum Critical Specification (1987):

This principle has two aspects, negative and positive. The negative simply states that no more should be specified than is absolutely essential; the positive requires that we identify what is essential. (p. 155)

This principle has wide utility in all of engineering where engineers are constantly faced with decisions about what is good enough. Engineers tend to produce designs that include significant over design in order to both reduce uncertainties and ensure success. By applying the principle of the Minimum Critical Specification engineers are required to design as little as possible and only specify what is essential.

There are several reasons for placing bounds on a design; because there is never complete knowledge (principle of system darkness) of a system or total control of the resources required to completely specify a design. Whatever benefits are purposefully included in specifications become obsolete (often at a rapid pace) as the contextual elements surrounding the design become better defined. In many cases, early over specification may have a crippling effect on the ability of the design team to adapt to evolving changes in context.

The principle of Minimum Critical Specification, while stringent, recognizes that the essential must be specified. A system must be sufficiently well specified if it is expected to be viable. A strategy that selects alternatives that keep the most adaptive options open early in the design may prove to be more successful than one that chooses alternatives that permit few options. Cherns (1987) goes on to state:

This premature closing of options is a pervasive fault in design; it arises, not only because of the desire to reduce uncertainty, but also because it helps the designer to get his own way. We measure our success and effectiveness less by the quality of the ultimate design than by the quantity of our ideas and preferences that have been incorporated into it. (p. 155)

In summary, Cherns echoes Gibson's call for the inclusion of alternatives in each and every design (Gibson, 1991). The inclusion of alternatives requires analysis and ranking of each alternative, an exercise that often provides insight into solutions that would never have been considered. Oftentimes, barriers to previously insurmountable problems become feasible alternatives when viewed as part of the overall design.

In summary, organizational cybernetics provides very important elements for understanding and controlling software development projects. The use of the Law of Requisite Variety, the Viable Systems Model, and the concepts of socio-technical system design are given important consideration as part of the systemic framework.

Summary of the Literature on Systems Principles

The literature search on systems principles provides a solid theoretical foundation that may be applied when understanding software development project performance. The richness of the language, methods, and techniques available to those who work with complex systems, is directly applicable to large software engineering projects, which are themselves, complex systems. The principles of holism, complementarity, satisficing, context, suboptimization, and variety, and the formalism included in model based methods and sociotechnical design are essential elements used in the inductive development of the theoretical framework for software development in the first element of the research.

Table 4 summarizes journal articles from the systems principles thread of the literature schema that have direct application to this research. Table 4 provides a

preliminary view of how each article will add to the theoretical and applied knowledge required to inductively build the framework for software development.

Systems Principles	Journal Article
Holism	West (2004)
Complementarity	Bohr (1928, 1937)
Satisficing	Simon (1955, 1956)
Context	Ackoff (1979a, 1979b); Keating, Kauffmann & Dryer (2001)
Sub-optimization	Hitch (1953)
Variety	Ashby (1956); Conant & Ashby (1970)
Viability	Beer (1984), Espejo (2004) and Yolles (2004)
Model-based Methods	Walton (2004); Ramsay, Boardman & Cole (1996)
Sociotechnical Design	Shani, Grant, Krishnan & Thompson (1992); Shani & Sena (1994); Jacobs, Keating & Fernandez (2000)

Table 4: Summary of the Literature on Systems Principles

SYNTHESIS OF THE LITERATURE ON SYSTEMIC IMPROVEMENT FRAMEWORKS FOR SOFTWARE DEVELOPMENT

Systemic improvement frameworks for software development is the second thread in the literature review. As in the previous section, Figure 7 is provided to re-orient the reader during the extensive literature review. There is a great deal of information in the

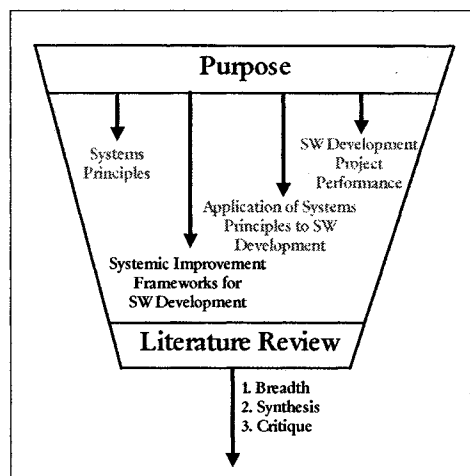


Figure 7: Systemic Improvement Frameworks for Software Development

literature about both software frameworks and software process improvements. In order to better understand how the software engineering community approaches improvement efforts, the literature in this area it has been split into four categories: (1) software

development frameworks, (2) software development methodologies, (3) software process improvement, and (4) software project management.

Software Development Frameworks

An excellent starting point in understanding the “. . . dizzying array of software and process standards, recommended practices, guidelines, maturity models, and other frameworks” is the article *Evolution of the Frameworks Quagmire* by Sheard (2001).

Figure 8 shows the evolution of the major software and systems standards that incorporate compatibility with one-another as an element of their design.

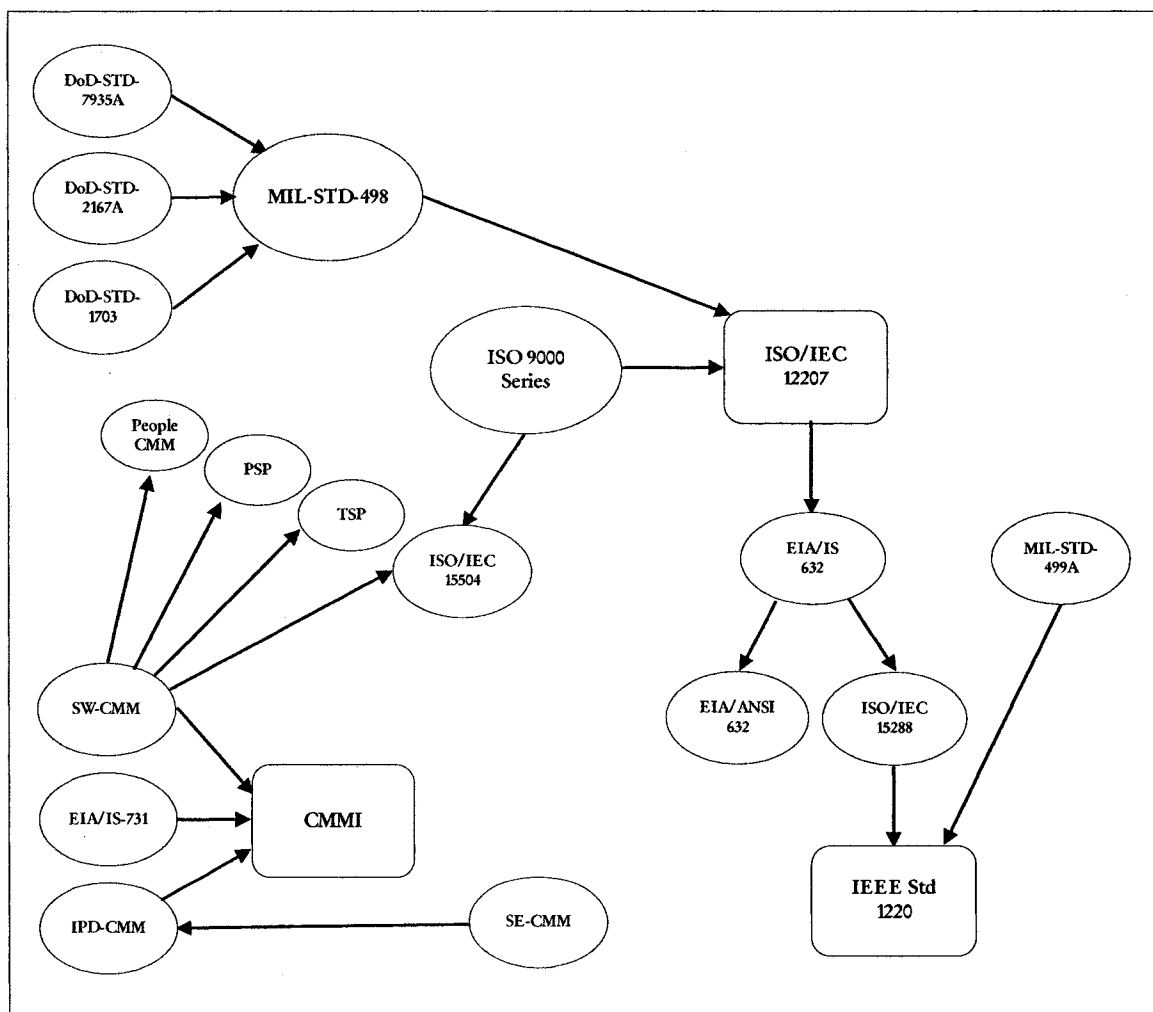


Figure 8: Evolution of Major Software and Systems Standards

In a follow-on article Sheard (2002) goes on to show that capability models and other process standards such as ISO 90003 and EIA/IS 731 generally ask the organization to do similar or identical things and that prior to adopting any new model an organization should map their current processes to the new model as part of the transition process. This approach is much easier than starting fresh, and when combined with an organization's existing process improvement methods and teams, can be an effective method of adoption. Clouse & Wells (2000) compare EIA/IS-731 and the CMMI[®] and recommend a well-planned transition for organizations adopting the CMMI[®]. The most significant of these models are:

1. Department of Defense/Software Engineering Institute Capability Maturity Model (CMM[®]).
2. International Organization for Standardization and International Electro-technical Commission joint standard ISO/IEC 12207, standard for Information Technology Life-Cycle Processes.
3. International Organization for Standardization and International Electro-technical Commission joint standard ISO/IEC 90003, quality in software-based products.
4. International Organization for Standardization and International Electro-technical Commission joint standard ISO/IEC 15504, Software Process Assessment.

Minnich (2002) conducted a similar comparison of EIA/IS 731 and the CMMI[®] concluding that the CMMI[®] contained more descriptive material.

DoD/SEI Capability Maturity Model:

Most of the early work in software process improvements was started at the Department of Defense's Federally Funded Research and Development Center, the

Software Engineering Institute (SEI), at Carnegie Mellon University (Lieblein, 1986). The early works of Watts Humphrey, who founded the Software Process Program at the SEI, were directed at improving software development by treating the entire development task as a process (1988, 1989). The early work on the CMM[®] was based upon the five stages of quality maturity espoused by Crosby (1979). Version 1.1 of the CMM[®] (Paulk, Curtis, Chrissis, & Weber, 1993) presented a set of recommended practices in a number of key process areas that proposed to enhance software-development. “The CMM[®] framework was a path of improvements to increased software process capability.” (Paulk et al, 1993, p. 24) The five levels in the CMM[®] each addressed different key process areas but failed to address the entire software process at any one level. Card (2000) and Murugappan & Keeni (2003) recommend blending the CMM[®] and the quality method of Six-Sigma to address precise operational definitions for continuous improvement in order to match process improvement goals with customer expectations and to predict and measure the capability in schedule, effort, and quality.

In 2003 the SEI issued the Integrated CMM[®] (CMMI[®]) in an attempt to integrate existing models and bodies of knowledge from four disciplines: (1) systems engineering, (2) software engineering’s software capability maturity model (SW-CMM[®]), (3) integrated product and process development (IPPD) and, (4) supplier sourcing. The stated goal was to improve practices from the four source models based on lessons-learned. With over 100 participants from 30 organizations in government and industry the CMMI[®] is a model built upon existing models and arrived at through compromise and agreement; not from first principles or focused research.

The CMMI[®] has two representations; the familiar five-level *staged* model and the newer *continuous* model. The continuous model allows organizations to conduct comparisons on a process-area-by-process-area basis and provides an easy migration from EIA Standard 731. The more familiar staged model provides a sequential improvement path which permits comparisons across and among organizations. Table 5 is a high-level comparison of the two model representations (SEI, 2005, p. 43).

	Staged Representation	Continuous Representation
Improvement method	Predefined and proven path with case study and ROI data	Maximum flexibility for order of process improvement
Model focus is upon	Organizational improvement	Improvement within process areas
Results presentation	Overall results summarized in a maturity level	Improvement of process areas can occur at different rates
Comparison	Maturity levels are common discriminators	Source selection investigation can target risky areas at any level

Table 5: Comparison of CMMI[®] Model Representations

Since its release in December of 2001, the CMMI[®] has been adopted for use by an increasing number of organizations. The SEI conducted a preliminary review of the experiences in twelve (12) large organizations that were early adopters of the CMMI[®] suite of products (SEI, 2003). The evaluation categorized the results into four primary classes of benefits: cost, schedule, quality, and customer satisfaction and a fifth class that addressed evidence about return on investment and related cost benefit matters. The following summarizes the results from the twelve cases for each of the five classes of performance measures (SEI, 2003, pp. ix-x.):

- *Cost: Six cases provide nine examples of cost-related benefits, including reductions in the cost to find and fix a defect, and overall cost savings.*

- *Schedule: Eight cases provide evidence of schedule-related benefits, including decreased time needed to complete tasks and increased predictability in meeting schedules.*
- *Quality: Five cases provide evidence of measurable improvements in quality, mostly related to reduction of defects over time or by product life cycle.*
- *Customer Satisfaction: Three cases show improvements in customer satisfaction, including demonstration of customer satisfaction through award fees.*
- *Return on Investment: Three cases claim a positive return on investment from their CMMI-based process improvement.*

The improvements for the early adopters of the CMMI® are consistent with those reported for the software CMM® in the literature. The staged model representation of the CMMI® will be referenced throughout the research study in order to allow comparison to the large body of work on the SW-CMM® which uses a similar staged representation.

ISO/IEC Standard 12207 Information Technology – Software Lifecycle Processes

This standard describes the major component processes of a complete software life cycle from conceptualization of ideas through retirement. The standard has primary, supporting, and organizational process areas, which group the activities that are performed during the software life cycle.

1. **Primary Processes:** Acquisition, Supply, Development, Operation, and Maintenance.
2. **Supporting Processes:** Documentation, Configuration Management, Quality Assurance, Verification, Validation, Joint Review, Audit, and Problem Resolution.
3. **Organization Processes:** Management, Infrastructure, Improvement, and Training.

The standard includes the tasks, the specific responsibilities, and outputs to complete each activity. The standard does not imply any specific life cycle model, and as such, includes no linkages or relationships between activities. Organizations that adopt ISO/IEC 12207 select an appropriate subset of the activities which can be tailored based on the scope, size, complexity and criticality of the software product under development. Ferguson & Sheard have conducted a comparison of ISO/IEC 12207 against the CMM[®] and found that “. . . organizations at CMM[®] levels 3 through 5 may need only minor additions to their software processes to achieve ISO/IEC 12207 compliance.” (1998, p. 28) These findings highlight the close relationship between the recommended practices in the CMM[®] key process areas and the life-cycle processes in ISO/IEC 12207.

ISO/IEC Standard 90003 Software Engineering – Guidelines for the Application of ISO 9001 to Computer Software

ISO/IEC 90003 (2004) is a quality management standard that provides guidance for organizations in the application of ISO 9001 to the acquisition, supply, development, operation and maintenance of computer software and related support services. ISO/IEC 90003 identifies the quality issues which should be addressed and, like ISO/IEC 12207, is independent of the technology, life cycle models, development processes, sequence of activities and organizational structure used by an organization. It is used to develop an ISO 9001 quality management system which can be used to apply for an ISO 9001 certificate. Mark Paulk (1995), the SEI CMM[®] product manager, compared ISO 9001 with the CMM[®] and found that “. . . although the CMM[®] does not adequately address some specific issues, in general it encompasses the concerns of ISO 9001. The converse is less true. ISO 9001 describes only the minimum criteria for an adequate quality-

management system, rather than addressing the entire continuum of process improvement.” (1995, p. 82). In a more recent study, van der Pijl, Swinkels & Verrijdt found that “although not perfect, [the] CMM[®] offers more possibilities than ISO 90003.” (1997, p. 273) Based on these findings ISO/IEC 90003 has limited utility as a framework for the development of software and should be limited to addressing quality management issues related to software.

ISO/IEC Standard 15504 Information Technology – Process Assessment

ISO/IEC 15504 (2004) provides a framework for the assessment of software processes. This framework can be used by organizations involved in planning, managing, monitoring, controlling, and improving the development of software. Process assessment has two principal contexts for its use; (1) process assessment and, (2) process capability determination.

Process assessment provides a means by which the current practices within an organization can be characterized vis-à-vis the selected processes. Analysis of the results identifies strengths, weaknesses and risks inherent in the processes. The assessment outcomes can help the assessed organization to determine whether the processes are effective in achieving their development goals, and to identify which processes are causes of poor quality, schedule slippages or cost overruns. These provide the drivers for prioritizing improvements to the identified processes.

Process capability determination is concerned with the capability of selected processes against targeted baseline process capabilities in order to identify the risks involved in undertaking a project using the selected processes.

El Eman and Birk (2000) conducted an empirical investigation of the relationship between the capability of a project's software design, code, integration and testing processes, as defined in the emerging ISO/IEC 15504 and the performance of the project. The findings indicate that the *develop software design* process is related to the project's ability to meet schedule commitments in small projects and that in large projects this is related to five different project performance measures.

Other Frameworks:

Humphrey, in association with the SEI, continues to write about and train software engineers in the integrated software development process. He has written texts on the Personal Software Process (PSP) to provide a disciplined way for individual software engineers to do their work (1995, 1996a, 1996b), the Team Software Process (TSP) which emphasizes the larger software development team (2000), and the role of the executive in winning with software (2002). Case Studies on the use of the PSP in three industrial software groups showed significant improvements (Ferguson, Humphrey, Khajenoori, Macke & Matvya, 1997). An additional empirical study of the PSP reports on the critical factors affecting the PSP (Zhong, Madhavji & Eman, 2000). A final paper reports on the implementation and use of the PSP and TSP at Microsoft (Grojean, 2005).

Boloix and Robillard (1995) constructed a software system evaluation framework that includes three dimensions; the *system*, which is subject to several *projects* during its lifetime, the *users*, and the *environment*, within which it operates. This framework uses a top-down approach that includes the software's producers, operators and users as essential elements.

Ibrahim & Weszka (2004) report on the use of multiple standards at the Federal Aviation Administration (FAA) and Lockheed-Martin. They provide guidelines for adopting the CMMI[®], ISO/IEC 9001, ISO/IEC 12207, and ISO/IEC 15504 in an integrated enterprise improvement framework.

In summary, the literature search on software development frameworks provides both historical background and details for the major frameworks in the literature. The application of many of the practices and processes contained within these frameworks may provide additional understanding about performance when applied to software development projects. The CMM[®] and ISO/IEC 12207 contain all of the requisite software development processes and practices and were included as essential elements used in the inductive development of the theoretical framework for software development in the first element of the research. In addition, the work by Boloix & Robillard (1995) which included critical social elements (i.e., software's producers, operators and users) as essential elements of their three-dimensional, project-based framework was given serious consideration.

Software Development Methodologies

There are three empirical studies on software development methodologies in the literature. The first describes a taxonomy of software development methods that uses conceptual and formal models and problem-oriented versus product-oriented methods as axes on the matrix. Blum provides the following definitions for the axes (1994, p. 83-85):

- ✓ **Conceptual models:** descriptive models that establish the response to the application domain need.

- ✓ **Formal models:** prescriptive models that set out the behavior of the software to be realized and are based on mathematics and logic.
- ✓ **Problem-oriented methods:** concentrate on producing a better understanding of the problem and its proposed solution.
- ✓ **Product-oriented methods:** center on the correct transformation from a formal specification into a maintainable implementation.

Blum arrives at two conclusions: (1) the tension between mathematical and conceptual approaches has an analogue: the tension between decomposition (top-down) and composition (outside-in). The former is consistent with the laws of mathematics, and the latter is closer to the way humans think, and (2) problem-oriented formal methods are the category with the greatest potential for process improvement.

The second paper is an empirical analysis of the fundamental philosophical assumptions of five contrasting Information Systems Development (ISD) approaches (Iivari, Hirschheim & Klein, 1998) and draws upon earlier research which sought to understand the dominant philosophical assumptions about the nature of information systems development (Orlikowski & Baroudi, 1991; Iivari, 1991; Hirschheim, Klein & Lyytinen, 1996; Iivari & Hirschheim, 1996). The paper has two distinct findings: (1) that the most appropriate unit of analysis is not an ISD methodology but classes of similar methodologies called *approaches*, and (2) a paradigmatic framework to support the information systems development approaches.

The third paper is a theoretic analysis of the intellectual structures of ISD (Hirschheim, Klein & Lyytinen, 1996). The paper provides an interesting generic framework for structuring and understanding ISD using five two-dimensional

frameworks that relate Habermas' (1984) social action types (instrumental, strategic, communicative and discursive) against Etzioni's (1968) domains of change (technology, language and organization). This paper provides a framework based on social action theories which conceptualizes ISD in terms of domains, orientations, object systems, and development strategies.

Additional papers on the factors that impact the implementation of system design methodologies (Roberts, Gibson, Fields & Rainer, 1998; Hardgrave, Davis & Riemenschneider, 2002), selecting a project methodology (Cockburn, 2000), the Rational Unified Process (Jacobsen, Booch & Rumbaugh, 1999; Manzoni & Price, 2003), and a framework for managing software development in small companies (Rautiainen, Lassenius & Sulonen, 2002) are of general interest but did not provide information that had utility in this research.

Software Process Improvement (SPI)

Software process improvement efforts at Hughes Aircraft (Humphrey, Snyder & Willis, 1991), Raytheon (Haley, 1996; Bowers, 2001), Motorola (Diaz & Sligo, 1997; Fitzgerald & O'Kane, 1999), an undisclosed firm (Harter, Krishnan & Slaughter, 2000), and Computer Sciences Corporation (McGarry & Decker, 2002) have been reported in the literature. Each improvement effort used the CMM[®] to measure the organizations software maturity. Each organization made improvements in their software quality, as reported in Table 6.

In addition to the improvement projects a number of specific SPI frameworks (aside from the CMM[®] which serves as both an improvement framework and *de facto* development methodology) are reported in the literature (Saiedian & Chennupati, 1999;

Wilson, Hall & Baddoo, 2001; Jiang, Klein, Hwang, Huang & Hung, 2004; Niazi, Wilson & Zowghi, 2005a; Niazi, Wilson & Zowghi, 2005b; Dyba, 2005). While of general interest, only two of these papers provide information that has utility in this research.

The paper by Jiang et al (2004) discusses the managerial processes that can be used to attack software development problems at each maturity level. The paper by Dyba (2005) is an empirical investigation of key factors in SPI and reports that six organizational factors have significant bearing on SPI.

Organization	Initial CMM® Level	Final CMM® Level	Major Measure of Software Quality Improvement
Hughes Aircraft Corporation (Humphrey, Snyder & Willis, 1991)	II (1987)	III (1990)	2. Cost Performance Index (CPI) improved from 0.94 to 0.97, saving \$2M annually.
Raytheon (Haley, 1996)	II	III	1. Rework or non-conformance decreased from 41% to 20%.
Raytheon (Bowers, 2001)	III	V	1. Productivity improvement of 144% going from level II to level IV. 2. Investment of 6% annual budget for process improvement. 3. ROI of 6 to 1.
Motorola GED (Diaz & Sligo, 1997)	Various	Various	1. Reduced defect density by a factor of 2 with each CMM® level increase.
Undisclosed firm (Harter, Krishnan & Slaughter, 2000)	Various	I, II, and III	1. Higher product quality. 2. Increased cycle time. 3. Increased development effort. Note: Marginal reductions in cycle time and effort outweigh the marginal increases from achieving higher levels of process maturity
Computer Sciences Corporation (McGarry & Decker, 2002)	I (1991)	V (1998)	1. Quality improvement of 10% per year. 2. Productivity improvement of 10% per year. 3. Cycle time improvements. 4. Estimation accuracy improvements.
Motorola CIG (Fitzgerald & O'Kane, 1999)	I (1993)	IV (1997)	1. 13 Critical success factors

Table 6: Software Process Improvements and CMM® Level

A final theme in software process improvement emerged from the review; people. Turner & Boehm (2003) address the fact that the most critical success factors facing software managers involve people factors; citing staffing, culture, values,

communications and expectations management. They conclude that people factors are critical to successful software development and management.

Software Project Management (SPM)

There is little empirical research on Software Project Management (SPM) in the literature. Sussman & Guinan (1999) proposed a theoretical model for SPM that identifies a characteristic of the technology and a characteristic of the team development process as effective elements in minimizing the adverse effects of complexity and ambiguity present in most software development projects. Abdel-Hamid, Sengupta & Swett (1999) explore the impact of goals on SPM and found that managers do make planning and resource allocation choices in order to meet the assigned goals. Ibbs & Kwak (2000) developed an assessment framework for project management maturity using the key elements of the Project Management Body of Knowledge (PMBOK). Hartman and Ashrafi (2002) reported on the current project management practices in the information systems and information technology industries. Jiang, Klein & Discenza (2002b) propose altering the pre-project activities of information systems projects to include the project manager and the team. They found that pre-project partnering; project manager performance and effective project team characteristics had positive effects on project outcomes. Chiang & Mookerjee (2004) propose a fault threshold policy to manage software development projects. The appearance of faults during system construction would act as the precursor to team meetings and management intervention. Kendra & Taplin (2004) review project success factors and find that success of project management relies on four dimensions of project success: the project manager skills and competencies, organization structure, measurements systems, and management practices that represent an organization's culture. Purvis, McCray & Roberts (2004) address the

use of heuristics in addressing complex decision situation sin information systems project management. Nidumolu & Subramani (2004) propose a synthesis of control that uses both the process and structure approaches for software development projects. The matrix of control uses four nodes; behavior control and outcome control (process control structure choices) and standardization control and decentralization control (structural control choices). A recent paper by Nguyen (2006) addresses project management from a decision making framework where a four-dimension decision model links task status in the development process to the responsible project authorities. While of general interest they did not provide information that had utility in this research.

Summary of the Literature on Systemic Improvement Frameworks for SW Development

The literature search on systemic improvement frameworks for software development provided both historical background and details for the major frameworks, methodologies and improvement processes in the literature. The application of two of the frameworks, two of the methodologies and two of the improvement processes provided additional understanding about performance and were utilized in to inductively build the framework.

Table 7 summarizes journal articles from the systemic improvement frameworks

Systemic Improvement	Journal Article
Frameworks	CMMI®, ISO/IEC 12207, Boloix & Robillard (1995)
Methodologies	Iivari, Hirschheim & Klein (1998); Hirschheim, Klein & Lyytinen (1996);
Process Improvement	Jiang, Klein, Hwang, Huang & Hung (2004); Dyba (2005);

Table 7: Summary of the Literature on Systemic Improvement Frameworks for Software Development

for software development thread of the literature schema that have direct application to this research. Table 7 provides a preliminary view of how each article will add to the

theoretical and applied knowledge required to inductively build the framework for software development.

SYNTHESIS OF THE LITERATURE ON THE APPLICATION OF SYSTEMS PRINCIPLES TO SOFTWARE DEVELOPMENT

Application of systems principles to software development is the third thread in the literature review. As in the previous section, Figure 9 is provided to re-orient the reader during the extensive literature review.

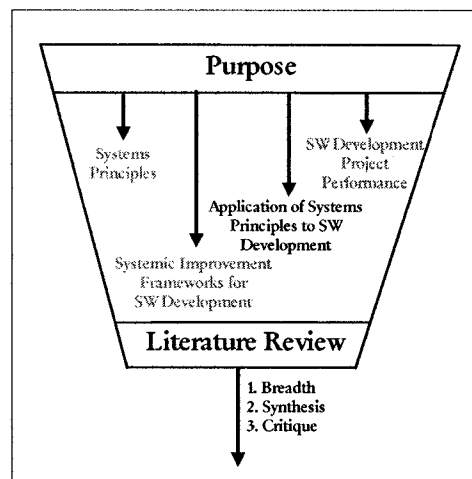


Figure 9: Application of Systems Principles to Software Development

The scholarly literature concerning the application of systems principles to software development was almost non-existent. Seven articles, all empirically based, were found. Table 8 categorizes the papers by the systems-based method being applied. The papers are listed chronologically in order to present the research thought as it was presented. All seven papers provide alternatives to traditional systems development using a variety of systems-based methodologies. Of particular interest are the papers by Bai & Lindberg (1999) and Bennetts, Wood-Harper & Mills (2000). Both papers approach information systems development holistically and provide approaches that were

considered during the inductive development of the theoretical framework for software development.

System Principle	Research Paper
System Dynamics	Abdel-Hamid, T.K. & Madnick, S.E. (1989). "Lessons Learned from Modeling the Dynamics of Software Development," <i>Communications of the ACM</i> , Vol. 32, No. 12, pp. 1426-1438.
Socio-cybernetics	Bai, G. & Lindberg, L. (1999). "A Sociocybernetic Approach to Information Systems Development," <i>Kybernetes</i> , Vol. 28, No. 6/7; pp. 792-809.
Soft Systems Methodology	Bennetts, P.D.C., Wood-Harper, A.T. & Mills, S. (2000). "An Holistic Approach to the Management of Information Systems Development--A View Using a Soft Systems Approach and Multiple Viewpoints," <i>Systemic Practice and Action Research</i> , Vol. 13, No. 2; pp. 189-205.
Oriental systems theory	Zhu, Z. (2000). "WSR: A Systems Approach for Information Systems Development," <i>Systems Research and Behavioral Science</i> , Vol. 17, No. 2, pp. 183-203.
Analogy	Day, J. (2000) "Software Development as Organizational Conversation: Analogy as a Systems Intervention," <i>Systems Research and Behavioral Science</i> , Vol. 17, No. 4, pp. 349-358.
Model validation	Petkova, O. & Petkov, D. (2003). "A Holistic Approach Towards the Validation and Legitimation of Information Systems," <i>Kybernetes</i> , Vol. 32, No. 5/6; pp. 703-714.
Viable System Model	Ríos, J.P. (2004). "A self-organizing network for the systems community," <i>Kybernetes</i> , Vol. 33, No. 3/4, pp. 590-606.

Table 8: Systems-Based Methods for Software Development

Software Engineering Texts and Body of Knowledge

A review of the major software engineering texts was conducted to see if systems principles or systems theory was mentioned. The review included the generalized texts on software engineering and the specialized texts on software engineering management. The generalized texts included Humphrey (2000), Pfleeger (1998), Pressman (2001), and Somerville (2005) and the specialized texts written by Bennatan (2000), Dorfman & Thayer (2002), Futrell, Shafer & Shafer (2002), Gilb (1998), Phillips (1997), Reifer (2002), Schwalbe (2002), and Thayer (1997). None of the texts make reference to either systems principles or systems theory.

Finally, a review of the Software Engineering Body of Knowledge (Abran & Moore, 2004) was conducted and no mention of either systems principles or systems theory was made. However, the SWEBOK does mention complex systems. It states that in one sense it should be possible to manage software engineering in the same way as any other complex system but that there are some uniquely inherent aspects to software that truly complicate its management. The four specific aspects are (Abran & Moore, p. 8-2):

1. The perception of clients that there is a lack of appreciation for the complexity inherent in software engineering, particularly in relation to the impact of changing requirements.
2. Related to the point just made, it is almost inevitable that the software engineering process itself will generate the need for new or changed client requirements.
3. As a result, software is often built in an iterative process rather than a concrete sequence of closed tasks.
4. Software engineering necessarily incorporates aspects of creativity and discipline – maintaining an appropriate balance between the two is often difficult.

While these aspects may seem unique to a software engineer, they are a matter of routine to a systems engineer involved with the process of producing and managing complex systems. The lack of literature relating systems principles or systems theory to software engineering points out a major gap in the literature and will serve as the focus for this research.

Summary of the Literature on the Applications of Systems Principles to Software Development

The literature search on the applications of systems principles to software development revealed two papers that approach information systems development

holistically and provide approaches that would be considered during the inductive development of the theoretical framework for software development.

Table 9 presents the two journal articles from this thread of the literature schema that have direct application to this research.

Application of Systems Principles	Journal Article
Information Systems Development	Bai & Lindberg (1999); Bennetts, Wood-Harper & Mills (2000);

Table 9: Summary of the Literature on the Application of Systems Principles to Software Development

SYNTHESIS OF THE LITERATURE ON SOFTWARE DEVELOPMENT PROJECT PERFORMANCE

Software development project performance is the final thread in the literature review. As in the previous section, Figure 10 is provided to re-orient the reader during the extensive literature review.

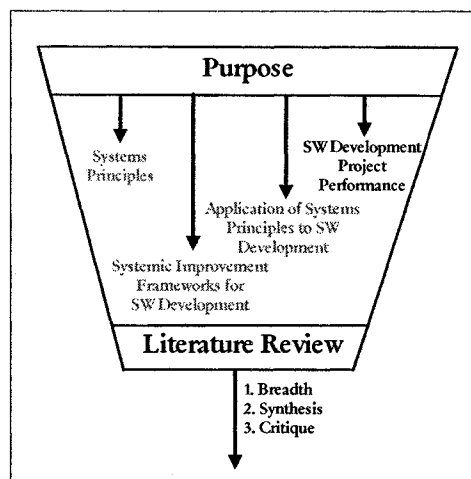


Figure 10: Software Development Project Performance

There is a great deal of information in the literature about software development project performance. In order to better understand how the software engineering community approaches project performance, the literature in this area it has been

organized into two categories: (1) worldwide software development and (2) development project performance.

Worldwide Software Development

Software researchers have been surveying global development activities since the 1990s. Cusumano (1989) was the first to report on the Japanese software factory approach to software development and the use of procedures and tools, refined project management techniques, and advanced technology for reuse support and automated programming. Cusumano & Kemerer (1990) conducted an empirical study of the state of practice in the United States and Japan and found that the Japanese spent more time in the design and testing of software. Duvall (1995) conducted a qualitative grounded theory study and found that software development was a sociotechnical discipline with large social aspects. Cusumano, MacCormack, Kemerer & Crandall (2003) found that Indian software development organizations are doing an admirable job with conventional best practices.

Software Development Project Performance

An excellent starting point in understanding software development project performance is van Genuchten's (1991) empirical study of why software is late. This early study found that over-optimistic planning was cited as a probable cause for late delivery in all the studies. Banker & Kemerer (1992) follow this with an empirical study of performance evaluation metrics and recommend a formal model that defines criteria to predict the choice of a performance metric. Deephouse, Mukhopadhyay, Goldenson & Kellner (1996) conducted an empirical investigation of software processes and project performance finding that project planning and cross-functional teams were consistently associated with favorable outcomes. Reichelt & Lyneis (1999) conduct an empirical

analysis of the drivers of cost and schedule overruns and find that rework is the most significant factor. Keil & Robey (1999) review the factors surrounding de-escalation of troubled projects and found that the principal factor was the communication of “. . . bad news” from those that find it to those who are in a position to do something about it, overcoming the *mum effect*. Keil, Mann & Rai (2000) follow-up on this topic and use constructs from theories to explain the escalation phenomenon and to test various regression models for their ability to discriminate between projects that exhibit escalation and those that do not. Aladwani (2002) proposes a theoretically driven performance model of information systems development projects in which the research focuses on organizational teams, placing social context in the forefront of information systems project performance. Harter & Slaughter (2003) review quality improvement and infrastructure costs and find that infrastructures costs make up 42% of firms total information technology expenditures, suggesting that these activities should be included in quality improvement endeavors. The study also finds that the greatest margin of cost savings are realized in infrastructure processes that are closely allied to the development processes and that occur late in the software development life cycle processes. Wallace, Keil & Rai (2004) conducted an empirical study of project risks and how they affect project performance. They use sociotechnical system theory to develop an exploratory model that describes six dimensions (subsystems of risk) of software project risk. The results suggest that social subsystem risk increases technical subsystem risk, as determined by requirements and technical complexity. Jones (2004) conducted an empirical study of 250 large software development projects between 1995 and 2004 and found that “. . . those that ran late, were over budget, or were cancelled without

completion, six common problems were observed: poor project planning, poor cost estimating, poor measurements, poor milestone tracking, poor change control, and poor quality control.” (Jones, 2004, p. 5) The final paper takes a holistic view of the root causes of complex program and project failures in the Department of Defense (DoD) and is addressed by Charette, Dwinnell & McGarry (2004). Their analysis was conducted as part of the DoD Tri-Service Assessment Initiative and concluded that as a project team’s level of *systemic understanding* matured, their ability to address problems was significantly improved.

Summary of the Literature on Software Development Project Performance

The literature search on software development project performance provides a number of empirical studies that address software performance with respect to a model, metrics, escalation, risk and the causes of failure. Six of these journal articles provide additional understanding about performance that would be considered during the inductive development of the theoretical framework for software development

Table 10 summarizes the six journal articles from the software development project performance thread of the literature schema that have direct application to this research.

Application of Systems Principles	Journal Article
Performance Model	Aladwani (2002)
Performance Metrics	Banker & Kemerer (1992)
Escalation	Keil & Robey (1999); Keil, Mann & Rai (2000)
Risk and Performance	Wallace, Keil & Rai (2004)
Failure Causes	Jones (2004)

Table 10: Summary of the Literature on Software Development Project Performance

SYNTHESIS OF THE LITERATURE – THE RELATIONSHIP OF RESEARCH TO THEORY AND PRACTICE

This section provides a review of the literature contained in all four threads of the review; systems principles, systemic improvement frameworks for software development, application of systems principles to software development, and software development project performance. It focuses on the empirical studies in Table 11 that have contributed to the research and provided the foundation for the development of the theoretical

Literature	Systems Principles	Systemic Improvement Frameworks for SW Development	Application of Systems Principles to SW Development	SW Development Project Performance
West (2004)	X			
Bohr (1928, 1937)	X			
Simon (1955, 1956)	X			
Ackoff (1979a, 1979b); Keating, Kauffmann & Dryer (2001)	X			
Hitch (1953)	X			
Ashby (1956); Conant & Ashby (1970)	X			
Beer (1984), Espejo (2004) and Yolles (2004)	X			
Walton (2004); Ramsay, Boardman & Cole (1996)	X			
Shani, Grant, Krishnan & Thompson (1992); Shani & Sena (1994); Jacobs, Keating & Fernandez (2000)	X			
CMM [®] , CMMI [®]		X		
ISO/IEC 12207		X		
Boloix & Robillard (1995)		X		
Iivari, Hirschheim & Klein (1998)		X		
Hirschheim, Klein & Lyytinen (1996)		X		
Jiang, Klein, Hwang, Huang & Hung (2004)		X		
Dyba (2005)		X		
Bai & Lindberg (1999)			X	
Bennetts, Wood-Harper & Mills (2000)			X	
Banker & Kemerer (1992)				X
Keil & Robey (1999); Keil, Mann & Rai (2000)				X
Aladwani (2002)				X
Wallace, Keil & Rai (2004)				X
Jones (2004)				X

Table 11: Literature Relationship to Research Purpose

framework for software development. All of the citations in Table 11 have some bearing on the research and relate theory and practice to the principal research questions. Table 11 shows the gap in the literature surrounding the application of systems principles to software development which served as the focal point for the research.

The purpose of the research was to develop and apply a systems-based framework for the analysis of software development project performance. Because the traditional method of predicting software development project performance, in terms of sub-system performance may be too restrictive, a new holistic, systemic view may reveal a better way to look at performance. The framework provides the conceptual basis for understanding the context surrounding software development projects, but will also support the development of formal methodologies that can be used by software practitioners to improve software development project performance.

The strength of the framework has been based upon being grounded in the theoretical constructs derived from the application of systems theory. Development of the framework used *Discoverers' Induction*, with the categories, attributes, relationships, and dimensions of the framework drawn directly from the literature in Table 11. Figure 11 has been included to illustrate how the four research threads come together to shape the development of the framework. Five important concepts were drawn from the synthesis of the literature. The concepts [followed by specific references] as are follows:

1. A number of systems-based principles and concepts exist in the literature that can be applied to the research questions. [They include the principles of complementarity (Bohr, 1928, 1937), satisficing (Simon, 1955, 1956), context (Ackoff, 1979a, 1979b, Keating, Kauffman & Dryer, 2001), sub-optimization

(Hitch, 1953), requisite variety (Ashby, 1956 and Conant & Ashby, 1970), and socio-technical system design (Shani, Grant, Krishnan & Thompson (1992); Shani & Sena (1994); Jacobs, Keating & Fernandez (2000).]

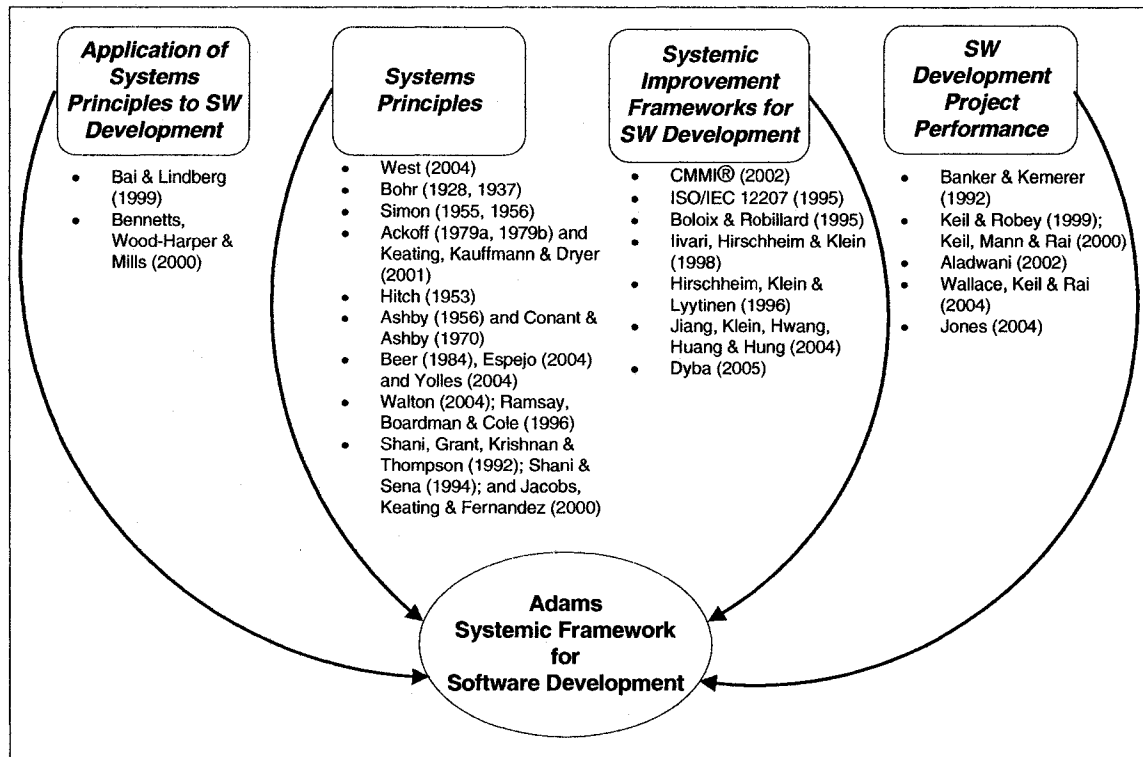


Figure 11: Literature Threads

2. Systems-based methods and models exist that may be adequate to holistically describe the software development process. [Two systems-based cybernetic methods, the Viable Systems Model (Beer, 1984, Espejo, 2004 and Yolles, 2004) and the soft systems method of Ramsay, Boardman & Cole (1996) as well as Banathy's three-lens model (Walton, 2004) provide framework elements applicable to the research questions.]
3. Few existing software development frameworks and/or methodologies address the overall development process holistically. [Exceptions include (1) an empirical analysis of the fundamental philosophical assumptions of five

contrasting Information Systems Development (ISD) approaches (Iivari, Hirschheim & Klein, 1998) and (2) the generic framework for structuring and understanding information systems development reported by Hirschheim, Klein & Lyytinen (1996) that uses Habermas' (1984) social action types and Etzioni's (1968) domains of change.]

4. There has been limited application of systems principles to the problems associated with software development. [Only Bai & Lindberg (1999) and Bennetts, Wood-Harper & Mills (2000) approach information systems development holistically.]
5. The literature on software development project performance does not address the root causes of poor performance. [Aladwani (2002) proposes a theoretically driven performance model of information systems development projects in which the research focuses on organizational teams, placing social context in the forefront of information systems project performance. Dyba (2005) addresses specific factors affecting software project performance.]

The five concepts encompass all of the relevant journal articles from the literature review. These five concepts provide a structure that has been used as an endorsement of the research by supporting the unique nature of the purpose, objectives and research questions.

CRITIQUE OF FINDINGS

A review of the body of literature has been conducted with a focus on gaps in the research and the need for additional empirical research related to the research purpose and primary research questions. The use of the synthesized literature in the inductive

element of the research included a discussion of its purpose and the explicit boundaries it sets for the researcher.

Gaps in the Existing Literature

A major gap in the literature existed in the treatment of software development projects as an organized or complex whole; a system. The software engineering community has been unable to coherently integrate their knowledge of the individual software development and management processes (sub-systems) in order to better understand the overall socio-technical system in which each of the development and management processes exists. Notable exceptions are the work of Abdel-Hamid (1984, 1988, 1992, 1993, 1996), Abdel-Hamid & Madnick (1989, 1990, 1991), Thayer (1979, 1997, 2002), Thayer & Christensen (2002), Thayer & Dorfman (2002), and Sengupta & Abdel-Hamid (1996) who have approached the management of software development projects from a holistic, systemic perspective. An early text on software engineering includes the following statement (Jensen & Tonies, 1978):

There is much attention on individual phases and functions of the software development sequence, but little on the whole life cycle as an integral, continuous process – a process that can and should be optimized. (p. 25)

They go on to state:

A systems treatment of the whole process from conceptual stage through product installation and operation is needed. (p. 25)

Since these statements were made limited research has been conducted on the integrated software development life-cycle process. In fact, the widely accepted literature on software process improvement provides conflicting advice. Of particular concern is the widespread adoption and implementation of the Software Engineering Institute (SEI) Capability Maturity Model (CMM[®]) and the Integrated CMM[®] (CMMI[®]). The SEI

CMM[®] and CMMI[®] have been used as frameworks for assessing an organization's ability to produce software as well as a *de facto* software development standard. CMM[®] and CMMI[®] assessments routinely evaluate software development projects and organizations by reviewing the individual processes that are described in the CMM[®] or CMMI[®], without regard for the integrated process. The CMM[®] and CMMI[®] and their associated assessment methodologies, run contrary to the *Principle of Suboptimization*, where optimization of the individual software development and management processes (sub-systems) occur at the expense of the larger software development project (system).

Applications of the Literature to the Research

The journal articles synthesized from the scholarly literature provided the empirical facts required to meet the first objective of the research:

Inductively develop a literature based, systemic framework to analyze software development project performance

The framework was developed using an inductive method that is based on a literature-intensive research effort where the synthesized literature frames the study. The synthesized literature also established two specific boundaries for the researcher:

1. It ensured that the researcher was exposed to a range of ideas, concepts and theories from the extant literature. "Reviewing relevant literature enhances traditional induction by helping theorist's link emerging theory to extant work recognizing the influence of their own theoretical inclinations." (Lewis & Grimes, 1999, p. 678)

2. It acted as a filter affecting the importance placed on the observations made by the researcher and the decisions to include or exclude particular journal articles as elements of the research.

By including these elements as explicit statements in the research design the researcher ensured that the underlying assumptions and boundaries of the research are the principal factual information/data sources for the induction. By ensuring that the inductively developed framework was linked to the literature the researcher enhanced the internal validity, generalizability, and theoretical level of theory building (Eisenhardt, 1989).

A new holistic, systemic view may reveal a better way to look at performance. The application of the synthesized literature, within a structured systemic framework for software development, provides insight into the failure to achieve overall software development project (system) improvement despite improvements within a number of the individual software development processes (sub-systems).

SUMMARY

This chapter has shown how the research related systems principles to software development project performance. It has presented the schema and breadth of the literature review. It has provided a synthesis of the salient facts and exposed gaps in the literature; highlighting the need for additional empirical research related to the research purpose and primary research questions. The chapter provides a solid literature-based foundation for the overall research effort and the extant literature required to build the inductive framework. The importance of the literature review cannot be overemphasized, as it is the foundation for the development of the framework for software development. The inclusion of an expert review, outside of the researcher, decreased research risk and

added validity to the literature used for the induction. The additional literature sources recommended by the expert reviewer provided additional empirical facts used in the induction.

The next chapter will provide an outline of the high-level research and dissertation concept as well as a detailed description of the research paradigm, research methodology and how the research complied with the Canons of Science.

CHAPTER III – RESEARCH METHODOLOGY

This chapter reviews the high-level research and dissertation concept and provides a detailed description of the research paradigm in terms of the researcher's view and the problem under study. It provides the rationale for the selection of a mixed-method design and reviews the challenges presented by both the inductive and case study elements of the research. It concludes by stating how the research complied with the Canons of Science.

EMPIRICAL SCIENCE AND METHODOLOGY

The principal role of the researcher is “. . . the creation of theory and the providing of empirical support for theory.” (Camilleri, p. 170) The methodology used for this research study embraced all aspects of the scientific quest and provided a solid base for conducting empirical science. Herbert Blumer [1900-1987] (1970) identified six elements that are indispensable to inquiry in empirical science (pp. 22-23):

1. *The Possession and Use of a Prior Picture or Scheme of the Empirical World under Study.* A review of the literature related to and context surrounding the phenomena as it exists in the empirical world.
2. *The Asking of Questions of the Empirical World and the Conversion of Questions into Problems.* This is beginning of the inquiry where the structure of the problem determines the broad methodological approach to be used.
3. *Determination of the Data to be Sought and the Means to be Employed in Getting the Data.* The data requirements help solidify the specific methodology and technique used to collect empirical data for the inquiry.
4. *Determination of Relations between the Data.* The data connections form the basis for the findings. Specific techniques and procedures for understanding the connections are selected and invoked based on the form and character of the data connections.
5. *Interpretation of Findings.* The findings of the study are related to the outside body of knowledge that transcends the study.
6. *The Use of Concepts.* Significant elements that the researcher invokes that act as anchor points in the interpretation of findings.

The methodology described in this chapter and the research design and detailed procedure in the next chapter specifically addressed each of the six fundamental elements of empirical investigation. In so doing, the researcher was able to invoke principles of science that ensure that the temperamental and obdurate empirical world could be studied within an acceptable framework of scientific investigation.

RESEARCH AND DISSERTATION CONCEPT

The research methodology for this study was mixed-method. The research methodology used both qualitative and quantitative analysis methods to achieve the study purpose by answering the two principal research questions. The value of a mixed method design was that the strengths of each method were applied to the applicable question.

The nature of the first question

How does systems theory apply to the analysis of software development project performance?

required the use of a qualitative element where a subjective approach was used to understand the question within its rich contextual environment. “Interpreting information technology in terms of social action and meanings is becoming more popular as evidence grows that information systems development and use is a social as well as a technical process that includes problems related to social, organization, and conceptual aspects of the system.” (Kaplan & Duchon, 1988, p. 574) The second question

What results from the application of a systems-based analysis framework for analyzing performance on a software development project?

required the use of a quantitative element where an objective approach was used to validate the utility of the framework on real-world software development projects.

Qualitative Element

The qualitative element of the research methodology used an inductive theory building method to develop the theoretical framework for analyzing software development project performance. The development of the framework was founded upon the inductive method of William Whewell [1794-1866] called *Discoverers' Induction* (Whewell, 1858). The method used the literature-intensive research effort in Chapter 2 to provide the empirical facts used in the process of *colligation* where colligation is defined as (Snyder, 1997a):

Colligation is the mental operation of bringing together a number of empirical facts by superinducing upon them some idea or conception that unites the facts and renders them capable of being expressed by a general law. (p. 585)

The process of colligation is the purposeful action in which the researcher supplies something *to* the facts (in this case it is the holistic treatment of software development using systems theory), which causes them to be seen from a *new point of view*. The initial idea or conception regarding the use of the holistic principles of systems theory, for software development projects were *superinduced* upon the empirical facts generated in the literature review in Chapter 2. The qualitative inductive element was used to answer the 1st research question *How does systems theory apply to the analysis of software development project performance?* “The rationale for conducting an exploratory study is to develop pertinent hypotheses and propositions for further inquiry.” (Yin, 2003, p. 6)

Quantitative Element

The quantitative element of the research methodology used an objective case study design to validate the inductively developed framework by using two real-world

software development projects as validation cases. A positivist case study design was selected in order to study software development projects within their real-world context.

Case studies are applicable to inquiries that will (Yin, 2003, p. 13-14):

- ✓ Investigate a contemporary phenomenon within its real-life context especially when the boundaries between the phenomenon and context are not clearly evident.
- ✓ Cope with the technically distinctive situation in which there will be many more variables of interest than data points, and as one result relies on multiple sources of evidence, with data needing to converge in a triangulating fashion, and as another result.
- ✓ Benefit from the prior development of theoretical propositions to guide data collection and analysis.

The systems based framework developed in the qualitatively-based exploratory element was validated in the quantitatively-based explanatory element by answering the 2nd research question *What results from the application of a systems-based analysis framework for analyzing performance on a software development project?* The explanatory element included a quantitative analysis where the framework was evaluated through the use of two case studies. Figure 12 is a high-level view of the research concept and is reviewed in the following sections.

Inputs to the Research Design

The research design was affected by five inputs.

1. Contextual Compatibility. For researchers using qualitative inquiry, context plays a major role in each of the various analytical approaches. Researchers must

understand the importance of context and its essential nature in being able to understand and interpret contemporary phenomenon that includes a societal element.

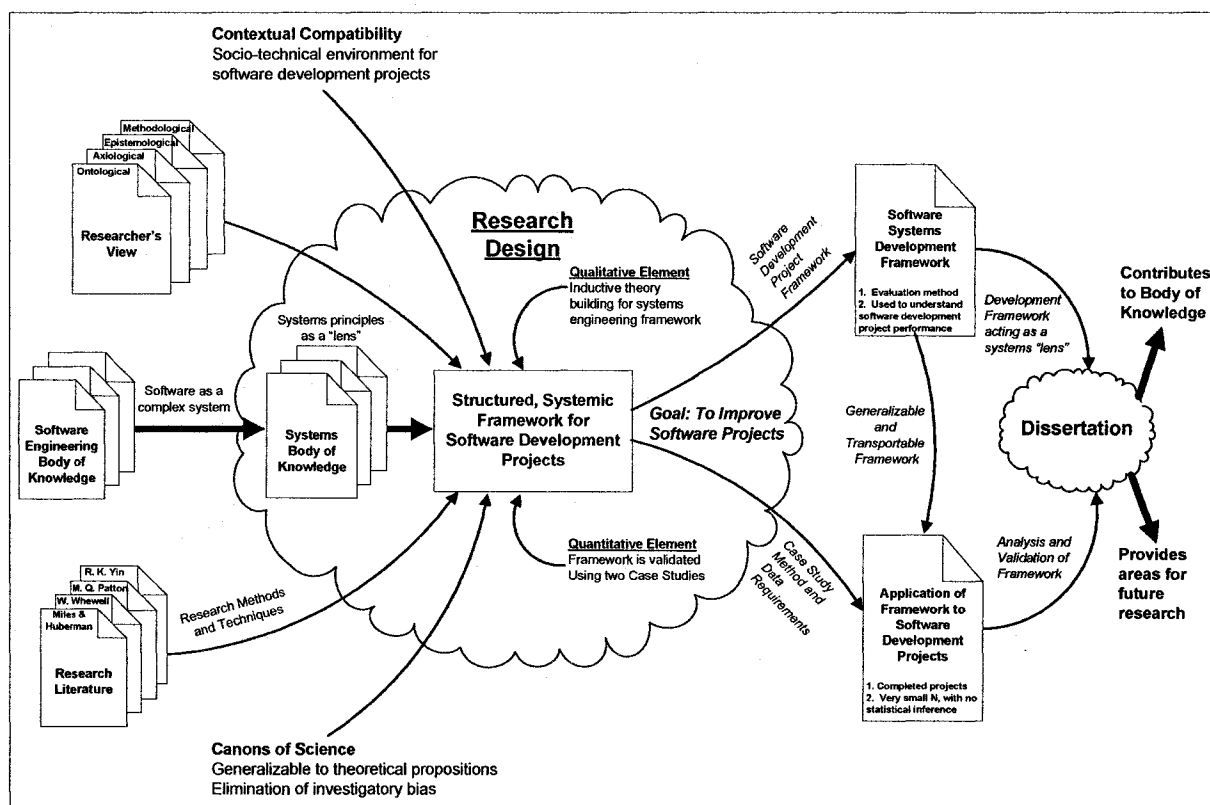


Figure 12: Research and Dissertation Concept

As such, context becomes an important part of the research methodology. Mishler (1979) notes that meaning is always within context and contexts incorporate meaning. Miles & Huberman (1994) believe that "... understanding contexts is critical. Even that adjective is too mild." (p. 102) Accounting for the rich contextual environment present in the socio-technical system used for software development projects was an essential part of the analytic strategy in the research design. The viewpoint of the researcher and the ontological, epistemological, axiological and methodological views all directly affected the context of the research study.

2. *The Researcher's View.* The theoretical and philosophical perspectives of the researcher, represented in the ontological, epistemological, axiological and methodological views directly influenced the conduct of the research.

3. *The Software Engineering Body of Knowledge (SWEBOK).* The SWEBOK (Abran & Moore, 2004) contains the foundation material that were related to the body of knowledge on systems in order to provide a holistic, structured, systemic framework for software development projects.

4. *Research Literature.* The body of literature on research methods and techniques provided the researcher with proven methods for the conduct of high-quality research.

5. *The Canons of Science.* The Canons of Science provided a universal accepted scientific standard for the research.

All five of these inputs serve to influence the research design which governs the conduct of the research study. The next section discusses the high-level research design.

The Research Design

The research design contained the logic that linked the empirical data being studied to the initial questions of the study. The research design had both qualitative and quantitative elements. The qualitative element of the research methodology used an inductive theory building method called *Discoverers' Induction* to develop a theoretical framework for analyzing software development project performance and answer the 1st research question *How does systems theory apply to the analysis of software development project performance?* The data for this element of the research came from the intensive literature review in Chapter 2. The quantitative element of the research methodology used a positivist case study methodology to validate the inductively developed

framework using two real-world software development projects to answer the 2nd research question *What results from the application of a systems-based analysis framework for analyzing performance on a software development project?* The data for this element of the research came from two real-world case studies. “The case study methodology has five components that are especially important: (1) the study’s questions; (2) the propositions; (3) the units of analysis; (4) the logic linking the data to the propositions; and (5) the criteria for interpreting the findings.” (Yin, 2003, p. 21) Each of the case study elements will be included in the detailed research procedure in Chapter 4.

The research design was divided into five phases. The research questions and propositions are formalized and the framework was inductively developed from the literature in Phases 0 and 1. The validating Case Studies were conducted in Phases 2 and 3. The research report was compiled for presentation in Phase 4. The research design, phases, and detailed procedural steps are presented in Chapter 4.

Outputs of the Research Design

The outputs of the research design included the holistically developed, structured, systemic framework for software development projects and the two case studies used to validate the framework. This dissertation is the principal product. A secondary product, in the form of an article in a scholarly journal, will be produced in order to extend the research findings to a wider audience.

THE RESEARCH PERSPECTIVE

Creswell (2003) has conceptualized his ideas about research design into a model that shows the series of interrelated decisions that form the process of designing research.

Figure 13 is based on Creswell's model and depicts how the elements of the traditional research paradigm were translated into the design processes for purposeful research.

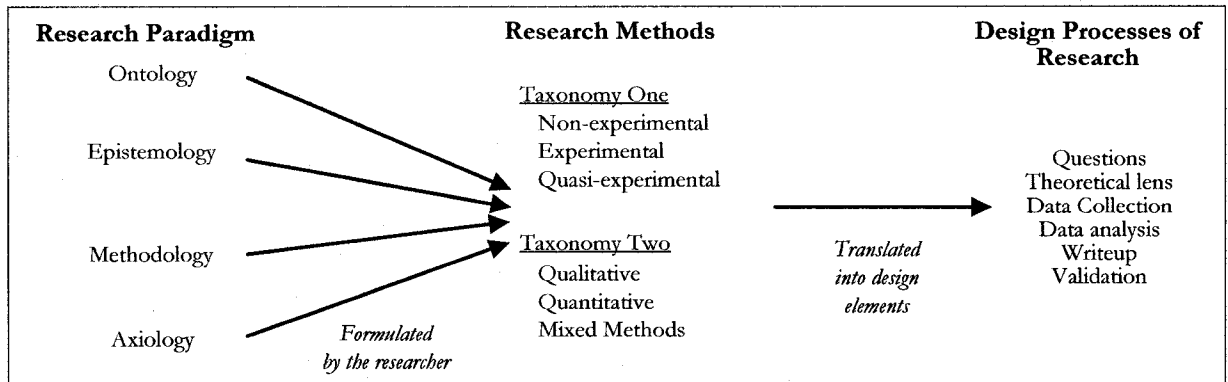


Figure 13: Elements of the Research Design

The research perspective is an alternative view of Creswell's model that relates two views; that of the researcher and of the problem under study. Both views are essential in all types and methods of research and provide for a solid understanding of the role that each play in the conduct of formal research. Figure 14 displays the elements of the research perspective used for this research. The theoretical and philosophical foundations that support the research methodology, within the research perspective in Figure 14, are explained in terms of the Researcher's View and the Problem under Study.

THE RESEARCHER'S VIEW

The research paradigm that underlies any research perspective describes the following set of basic assumptions for conducting research (Iivari, Hirschheim, and Klein, 1998):

- Ontology - the structure and properties of what is assumed to exist
- Epistemology - the nature of knowledge and the proper methods of inquiry

- Axiology (i.e., ethics) - the responsibility of a researcher for the consequences of his/her research approach and its results
- Research Methodology - the procedures used to acquire knowledge

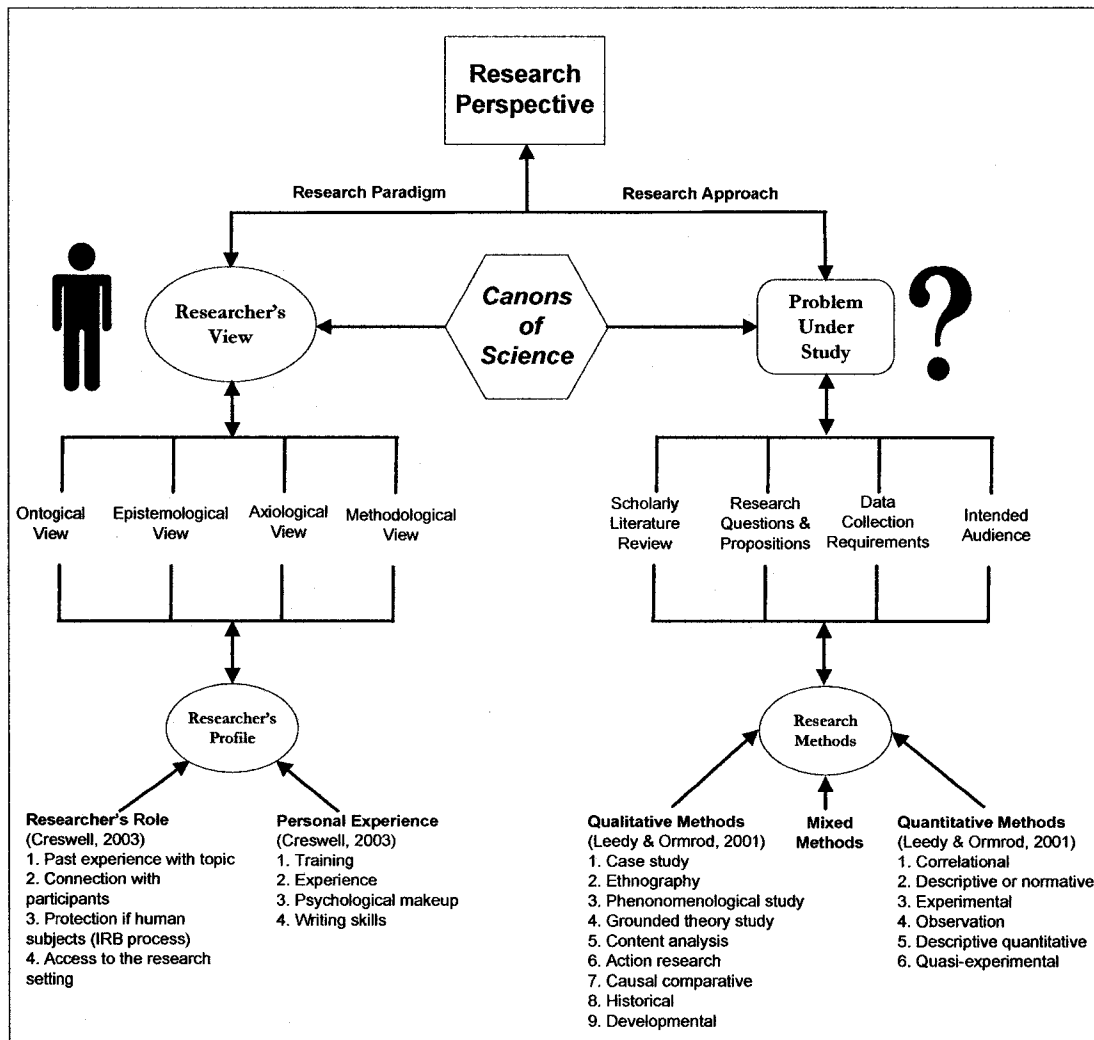


Figure 14: The Researcher's Perspective

The researcher's viewpoint was a function of his value system, normative behaviors, and perceived role. The role as a qualitative researcher was a new one and as such was affected by a number of important relationship values (Schein, 1992). However, the initial theoretical and philosophical perspectives that influenced the researcher were the ontological and epistemological views that he brought to the research.

Ontological View

Ontology is concerned with the theoretical perspective that lies behind the knowledge claims. While an explanation of the major classical ontological views provides interesting background material it is perhaps misleading to think that one must select from among just one of these views. A more realistic representation is that of a continuum between Idealism (the subjective school) and Realism (the objective school).

Figure 15 is the ontological continuum (Morgan and Smircich, 1980).

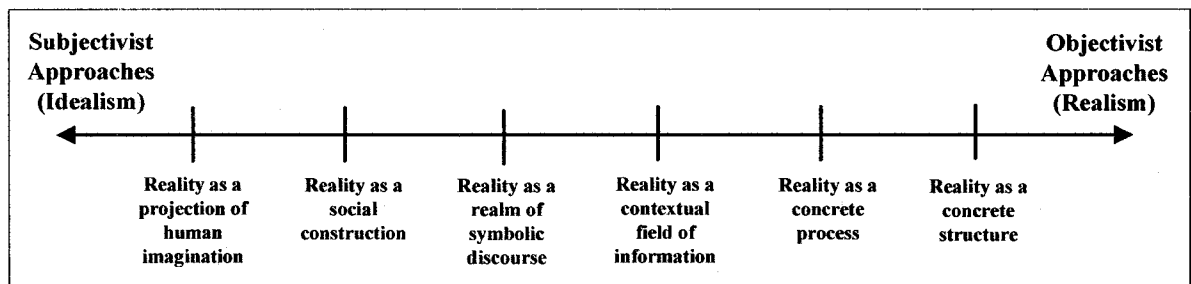


Figure 15: The Ontological Continuum

The ontological continuum in Figure 15 is a schema for thinking about the assumptions that underlie the research method. The assumption for this research was best described as *reality as a contextual field of information*.

Epistemological View

“In general, epistemological assumptions are concerned with the nature of knowledge and the proper methods of inquiry. By inquiry we mean the procedures or means by which we can obtain knowledge.” (Iivari, Hirschheim and Klein, 1998, p. 174) Creswell (2003) cites the four major schools of thought and the central elements of each position in Figure 16.

The choice of an epistemology was not made in a vacuum, but was an essential and interrelated part of the researcher’s view. A key element was the academic discipline

or area within which the research was being conducted or presented. For example, economists working in the field of econometrics would be shocked and most likely reject the use of the advocacy/participatory school of thought in a research study,

Positivism Determination Reductionism Empirical observation and measurement Theory verification	Constructivism Understanding Multiple participant meaning Social and historical construction Theory generation
Advocacy/Participatory Political Empowerment issue-oriented Collaborative Change-oriented	Pragmatism Consequences of actions Problem-centered Pluralistic Real-world practice oriented

Figure 16: Epistemological Schools of Thought

Adapted from Creswell, J.W. (2003). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. (2nd ed.). Thousand Oaks: Sage Publications, p. 6.

while a group of sociologists would tend to accept this as a valid knowledge claim. An understanding of the epistemological assumption, its applicability to the proposed research and associated data, and the degree of acceptance that the method will receive were elements of the selection process.

Finally, the epistemological emphasis can also be mapped on the ontological continuum between the subjective and objective. Figure 17 is a mapping of the basic epistemological stance along the ontological continuum (Morgan and Smircich, 1980).

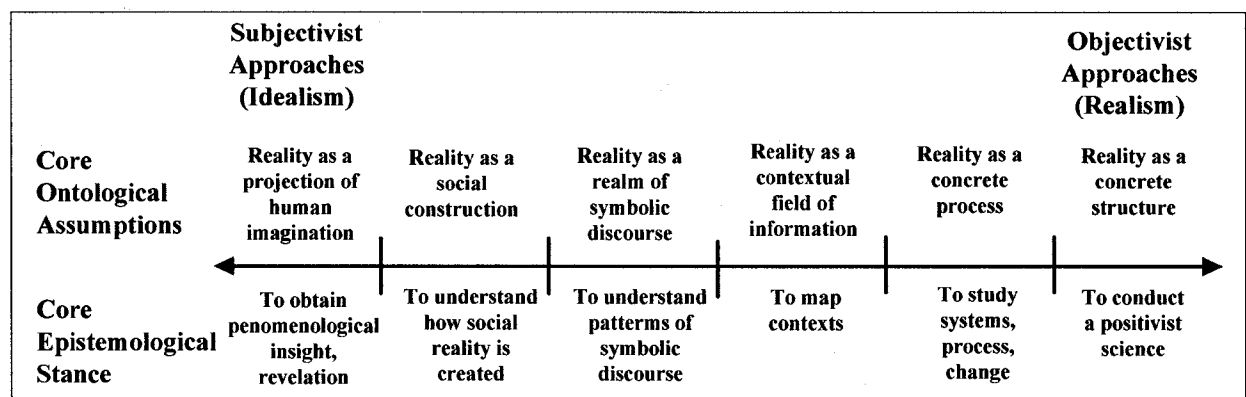


Figure 17: Continuum of Ontological Assumptions and Epistemological Stances

“Qualitative research stands for an approach rather than a particular set of techniques, and its appropriateness – like that of quantitative research – is contingent on the nature of the phenomena to be studied.” (Morgan & Smircich, 1980, p. 499) Because qualitative and mixed-method research took place in the natural setting, the researcher interacted with the participants of the research. “The qualitative researcher often goes to the site (home, office) of the participant to conduct the research. This enables the researcher to develop a level of detail about the individual or place and to be highly involved in actual experiences of the participants.” (Creswell, 2003, p. 181) The epistemological stance for this research fell between mapping contexts and studying systems, process and change.

Axiological View

Axiology refers to what is valued or considered as being right. This element is commonly referred to as ethics. Norman Augustine separates ethics into two categories: macro ethics, which involve ethical issues that affect large segments of the society; and micro ethics, that affect a smaller, more immediate group, such as one’s boss or one’s client (Augustine, 2002). Research can include both micro- and macro-ethical considerations and should be considered during the design of purposeful research. Singer & Vinson (2002) introduce ethical issues in empirical studies in software engineering and the use of ethical codes of conduct. “Most ethical issues in research fall into one of four categories.” (Leedy and Ormrod, 2001, p. 107)

Protection from Harm

Physical and psychological harm to research participants were primary considerations in the research design. The design of the research included safeguards and participants were fully informed ahead of time as to what to expect as part of the study.

A debriefing of participants after the study was also included as part of the research design.

Informed Consent

Participation in the research study was strictly voluntary. Participants were told the nature of the study and what specific activities their participation would involve.

Right to Privacy

“Any research study should respect participants’ rights to privacy. Under no circumstances should a research report, either oral or written, be presented in such a way that others become aware of how a particular participant has responded or behaved.” (Leedy and Ormrod, 2001, p. 108)

All three of these categories are related to research that involves human subjects. As such, the research required review and formal approval according to the guidelines established by the 1974 National Research Act. *The Belmont Report* (HEW, 1979) serves as the primary ethical framework for protecting human research subjects in the United States. The report sets guidelines that underlie the conduct of biomedical and behavioral research that involve human subjects and that are enumerated in public law. At Old Dominion University the Human Subjects Institutional Review Board (IRB) ensures that all research involving human subjects conforms to Federal, State, and any Local policies providing for the protection of human subjects. The IRB waiver and approval for this research are included in Appendix A.

Honesty with Professional Colleagues

The final category has two important elements. The first requires the researcher to report findings in a complete and honest fashion, avoiding misrepresentation of facts or

the intentional removal of information from the study. It also involves a central Canon of Science: *neutrality*. The research design purposefully includes assurance that personal prejudice and bias do not enter into the study. Separation of the researcher, who served as the instrument in the study, from the study findings, was an essential element of the design. The second element involved personal ethics and the requirement to properly cite ideas and concepts that belong to or have originated with others. This element has been meticulously applied throughout the research study.

Methodological View

The methodological view involves both the researcher's personal experience with qualitative, quantitative and mixed-methods research and the problem under study. In this case the researcher had prior academic experience producing a master's thesis based on an experimental study (Adams, 1986), but little experience with qualitative and mixed-method research. However, the problem under study necessitated the inclusion of the rich contextual environment in the analysis and as such required the use of qualitative methods. The researcher endeavored to ensure that the research design provided adequate rigor and complied with the Canons of Science. In addition, the researcher consciously steered clear of the qualitative case study method proposed by Robert Stake (1995) whose experience and interest was with more naturalistic, holistic, ethnographic, phenomenological and biographic research methods and "... does not pay much attention to quantitative case studies." (Stake, 1995, p. xi) The utilization of Yin's positivist case study method was a conscious effort to include a more empirical method that incorporated the design quality concepts for quantitative research with which the researcher was familiar.

Each methodology has a number of supporting methods and, as their central focus, the means to be used for data collection and analysis. A principal factor in the choice of methodology revolved around whether the intent was to specify the type of information to be collected in advance of the study or to allow it to emerge from participants during the research study. At this point the problem under study became the principal factor in the methodological selection.

THE PROBLEM UNDER STUDY

The problem under study, as the *raison d'être*, was focused on the research purpose. It was bounded by the technique of the research method and the Canons of Science. The details of the problem were contained in the scholarly literature, the research questions, applicable propositions, the data collection requirements, and the intended audience for the research study. Careful review of these details permitted the researcher, from within his previously described ontological, epistemological and axiological view, to make a rational decision as to which methodology to employ.

Scholarly Literature Review

The scholarly literature provided the empirical facts for Whewell's inductive method (*Discoverers' Induction*) where, through the process of *colligation*, the researcher supplied something *to* the facts, which caused the facts to be seen from a *new point of view*. In this case, the researcher's initial idea or conception regarding the use of the holistic principles of systems theory for software development projects was *superinduced* upon the empirical facts generated in the literature-intensive research. The initial literature review reported in Chapter 2 revealed five important threads that served as the baseline for the literature-intensive research in the first element of the design: (1) A

number of systems-based principles and concepts exist in the literature that can be applied to the research questions; (2) systems-based methods and models exist that may be adequate to holistically describe the software development process; (3) few existing software development frameworks and/or methodologies address the overall development process holistically; (4) there has been limited application of systems principles to the problems associated with software development; and (5) the literature on software development project performance does not address the root causes of poor performance.

Research Questions and Propositions

Leedy and Ormrod (2001) have provided a set of conditional questions that include elements from both the researcher's view and the problem under study, as a guide to the researcher. Table 12 is the set of conditional questions.

Question	Qualitative	Quantitative
<i>What is the purpose of the research?</i>	<ul style="list-style-type: none"> • To describe and explain • To explore and interpret • To build theory 	<ul style="list-style-type: none"> • To explain and predict • To confirm and validate • To test theory
<i>What is the nature of the research process?</i>	<ul style="list-style-type: none"> • Holistic • Unknown variables • Flexible guidelines • Emergent design • Context-bound • Personal view 	<ul style="list-style-type: none"> • Focused • Known variables • Established guidelines • Static design • Context-free • Detached view
<i>What are the methods of data collection?</i>	<ul style="list-style-type: none"> • Informative, small sample • Observations, interviews, questionnaires 	<ul style="list-style-type: none"> • Representative, large sample • Standardized instruments
<i>What is the form of reasoning used in analysis?</i>	<ul style="list-style-type: none"> • Inductive analysis 	<ul style="list-style-type: none"> • Deductive analysis
<i>How are the findings communicated?</i>	<ul style="list-style-type: none"> • Words • Narratives, individual quotes • Personal voice, literary style 	<ul style="list-style-type: none"> • Numbers • Statistics, aggregated data • Formal voice, scientific style

Table 12: Distinguishing Characteristics of Qualitative and Quantitative Approaches

Adapted from Leedy, P.D. & Ormrod, J.E. (2001). *Practical Research: Planning and Design* (7th ed.). Upper Saddle River: Prentice-Hall, p. 102

Whewell would argue that all research is *conception laden* and that there is a single *veil of theory over the whole of nature*. He would not support the notions

underlying a separate quantitative research methodology in the sense that it is purely and absolutely objective (L.J. Snyder, personal communication, May 7, 2006). This research used the modern qualitative and quantitative labels to clearly identify the established methods to the audience for the dissertation.

By answering these questions for the proposed research the researcher was able to determine that that this research required both qualitative and quantitative elements. To effectively address the research questions no single method would be adequate. Thus, a mixed-method approach was determined to best meet the goals of the research. A review of the research purpose helped to narrow the selection of specific qualitative and quantitative methods. In this case the proposed research was to *develop and apply a systems-based framework for the analysis of software development project performance*. The purpose was directed by the high-level research questions presented in Table 12.

The first question was:

How does systems theory apply to the analysis of software development project performance?

and the second question was:

What results from the application of a systems-based analysis framework for analyzing performance on a software development project?

Because the research questions contained both *how* and *what* questions, the study required both explanatory and exploratory elements.

The *how* question required an exploratory (subjective) approach where the rationale for conducting the study was to develop pertinent hypotheses and propositions for further inquiry (Yin, 2003). In this case the researcher used inductive theory building to derive and build a systems-based framework for the analysis of software development

project performance. *Discoverers' Induction* was the inductive method used to develop the framework. The method required a literature-intensive research effort to provide the empirical facts used in the process of *colligation* which was the purposeful action in which the researcher supplied the systems concepts of holism, complementarity, satisficing and suboptimization *to* the facts which caused them to be seen from a *new point of view*. The initial idea or conception regarding a holistic, structured, systemic framework for software development projects was *superinduced* upon the empirical facts generated in the literature review and served as the catalyst for the framework development.

The *what* question required the use of an explanatory (objective) approach where the researcher used an empirical method to validate the utility of the framework on real-world software development projects. Because the validation of the framework explored real-world processes, activities, and events in their natural setting the case study emerged as the method for validating the framework.

Data Collection Requirements

The data collection requirements for this research were purely qualitative, that is to say that the data must come from a real-world setting due to the social nature of the software development process. A formal framework that specified who and what was and was not studied was constructed using the guidelines developed by Miles and Huberman (1994). The conceptual data collection framework also included several specific qualitative methods for data collection and analysis related to software engineering (Seaman, 1999). Throughout, particular attention paid to the data design to ensure that the measures of design quality was met throughout the study.

Intended Audience

Software engineering research, like the field of software engineering, is continuing to emerge. The number of formally defined software engineering research methods is limited. The dearth of formal methods may be a direct function of the market forces that have propelled software engineering along at a pace unsurpassed by any other emerging profession in modern times. These seemingly irresistible forces have forced the industry to accept craft-based industrial practices, unacceptable in other engineering fields, as the norm (Potts, 1993). "Members of a *young* scientific field such as MIS, in search of legitimacy within the general scientific community, will generally see themselves in a position of followers, not leaders, as far as methodology is concerned. One can expect them to be more conservative and to *play it safe*." (Landry & Banville, 1992, p. 87) Software engineering has only begun to base its practices on solid academic and applied research, and as such a review of the research methods in use was warranted.

Mingers (2001) conducted an empirical review of six of the main IS journals to evaluate the extent of multi-method research. All articles between 1993 and 1998 were reviewed. Findings showed that roughly two-thirds of the papers contained some form of empirical research, and the predominant forms of research were: (1) surveys, (2) interviews, (3) experiments, and (4) case studies. These papers accounted for greater than 80% of the recorded examples. Participant observation, grounded theory, and soft systems methodology were almost entirely absent. Around 13% of the papers used a multi-method research design.

Finally, four large methodological examinations on the different types of research methods reported in the major information systems journals have been conducted. Table 13 is a summary of the data from each of the studies and shows that mixed method and

case study research are commonly used methodologies in information systems and software engineering. This supports the assertion by Kaplan and Duchon (1988) that:

Interpreting information technology in terms of social action and meanings is becoming more popular as evidence grows that information systems development and use is a social as well as technical process that includes problems related to social, organization, and conceptual aspects of the system. (p. 574)

Research Method	Orlikowski & Baroudi (1991)	Alavi & Carlson (1992)	Farhoomand & Drury (1999)	Chen & Hirschheim (2004)
Period Covered	1983-1988	1968-1988	1985-1996	1991-2001
Journals	Four	Eight	Eight	Eight
Number of Articles	155	908	2,098	1,893
Survey	49.1%	3.5%	32%	41%
Laboratory Experiment	27.1%	7.3%	10%	18%
Case Study	13.5%	4.4%	17%	36%
Mixed Method	3.2%			
Field Experiment	2.6%	2.0%	2%	2%
Instrument Development	2.6%			
Protocol Analysis	1.3%			
Action Research	0.6%			3%
Description		10.8%		
Ex-post Description		2.0%		
Development of tool		1.3%		
Secondary Data		0.7%		
Non Empirical		51.9%	39%	

Table 13: Information Systems Research Methods

RATIONALE FOR SELECTION OF THE MIXED METHOD DESIGN

The mixed method research design was selected in direct response to the research questions. Because no single method could adequately address each of the questions a mixed method approach was determined to best meet the goals of the research. The mixed method approach provided the researcher with the strengths present at both ends of the continuum of ontological assumptions and epistemological stances in Figure 17. The qualitative and quantitative research methodologies were able to directly associate the philosophical underpinnings that explain the subjective (idealism) and objective (realism)

perspectives, respectively. Each methodology brought distinct qualities and biases to the research and it was incumbent upon the researcher to ensure that both the principles of good research and the specific method were invoked. The rationale for each method was as follows.

Qualitative Method: Inductive Theory Building

The goal of the first element of the research was the production of a systems-based framework for the analysis of software development project performance. Glaser and Strauss (1967) cite four general approaches to the analysis of qualitative data that are depicted in Figure 18.

Because the research was generating and provisionally validating theory, an inductive approach was selected to develop the theoretical framework for software development. This was principal output of Phase 1 and was developed based on a literature-intensive research effort where the existing literature on the object of the study helped frame the study and was important for establishing validity in the research and confidence in the findings (Patton and Appelbaum, 2003). The initial idea or conception regarding a holistic, structured, systemic framework for software development projects was the object of the inductive method and was *superinduced* upon the facts from the literature. This phase was highly qualitative and relied on inductive theory building to construct the software systems engineering framework. Tying the emergent theory to the literature enhanced the internal validity, generalizability, and theoretical level of theory building (Eisenhardt, 1989).

from naturalistic, holistic, ethnographic, phenomenological and biographic research methods and “. . . does not pay much attention to quantitative case studies.” (Stake, 1995, p. xi) Yin treats the case study as an empirical method and incorporates the design quality concepts for quantitative research. Yin (2003) states:

Four tests, however, have been commonly used to establish the quality of any empirical social research. Because case studies are one form of such research, the four tests are also relevant to case studies. (pp. 33-34)

Yin’s assertion is supported by Lee (1989b) who makes a case for the use of case studies as natural experiments supported by the objectivist framework’s natural science model. Four key measures of design quality were observed and are presented in Table 14 (Yin, 2003, p. 34).

Measure of Design Quality	Definition
Construct Validity	Establishing correct operational measures for the study to ensure that <i>subjective</i> measures do not enter into the data collection.
Internal Validity	Establishing a causal relationship whereby certain conditions are shown to lead to other conditions.
External Validity	Establishing a domain to which a study’s findings may be generalized.
Reliability	The auditability and confirmability of the research is demonstrated by ensuring that the research study and data collection procedures can be repeated, with the same results.

Table 14: Measures of Case Study Design Quality

Yin (2003), in his book *Case Study Research: Design and Methods*, has purposely included the subtitle *Design and Methods* because it includes design procedures and methods to ensure that the case study researcher is able to systematically design and analyze case studies. The case study research design contains the logic that links the data to be collected to the initial questions of the study. The data for this research came from multiple case studies. The case study design incorporated five components that are especially important: “(1) the study’s questions; (2) the propositions; (3) the units of

analysis; (4) the logic linking the data to the propositions; and (5) the criteria for interpreting the findings.” (Yin, 2003, p. 21)

Both the qualitative and quantitative methods support the overall goal of the research which was the development of a framework to help those managing software development projects to gain a greater understanding of the concept of holism when applied to improving the software development process. Another important aspect of this research was for the developed framework to be applicable and understandable by software development practitioners. To ensure this, the researcher became intimately familiar with how holism in software process improvement is perceived and how the concept could be applied to the larger, real-world, systems-based software development process improvement effort. The need for focus on the real-world aspects of software development made this research project an appropriate instance for the application of a mixed method design of qualitative inductive and quantitative case study methods.

CHALLENGES TO THE INDUCTIVE METHOD

There are four challenges to the overall inductive method that focus on the method of discovery, reliability, verification, and epistemology related to the research.

Induction and Rational Discovery

A common challenge to the inductive method has been registered by those who adhere to hypothetico-deductive positions. The hypothetico element of this position holds that a theory may be developed non-inductively as follows (Kaplan, 1964):

The scientist, by a combination of careful observation, shrewd guesses, and scientific intuition arrives at a set of postulates governing the phenomenon in which he is interested; from these he deduces observable consequences; he then tests these consequences by experiment, and so

confirms or disconfirms the postulates, replacing them where necessary, by others, and so continuing. (pp. 9-11)

Robinson (1951) reports on a similar position by Znaniecki (1934) called Analytic Induction, which is more in line with the hypothetico-inductive school of thought than those that include inductive elements. Both Francis Bacon [1521-1626] and Whewell unequivocally reject such notions. "Bacon cites that in the method of induction the inference from data to hypothesis is performed by a process of *graduated and successive induction*." (Snyder, 1999, p. 541) Whewell (1857) states:

The first impulses of the human mind, even when it makes experience its starting point, are fallacious and delusive. (p. 158)

Claiming that

Bacon alone saw that knowledge . . . must be gained by a slow and gradual course . . . by a connected and gradual process. (p. 158)

In contrast, Karl Popper [1902-1994] supports the hypothetical tradition writing that:

The initial stage, the act of conceiving or inventing a theory, seems to me neither to call for logical analysis nor to be susceptible of it. The question how it happens that a new idea occurs to a man – whether it is a musical theme, a dramatic conflict, or a scientific theory – may be of great interest to empirical psychology; but it is irrelevant to the logical analysis of scientific knowledge. (Popper, 1959, p. 7)

Hans Reichenbach [1891-1953] counters Popper's analysis with the following observation.

The hypothetico-deductive method or 'explanatory induction,' has been much discussed by philosophers and scientists but its logical nature has often been misunderstood. Since the inference from the theory to the observational facts is usually performed by mathematical methods, some philosophers believe that the establishment of theories can be accounted for in terms of deductive logic. This conception is untenable, because it is not the inference from the theory to the facts, but conversely, the inference from the facts to the theory on which the acceptance of theory is based; and this inference is not deductive, but inductive. What is given are the observational data, and they constitute the established knowledge in terms of which the theory is to be validated. (Reichenbach, 1951, p. 230)

What emerges is a dichotomy of positions; with Bacon and Whewell in favor of a rational discovery method and Popper and the hypothetico school (*positivism*) in favor of an irrational discovery method. It is important to note that Bacon and Reichenbach support the “. . . ontological precept of the *tabula rasa*, with the direct corollary that all knowledge has its origins in experience.” (Sutherland, 1973, p. 58) Whewell, in his antithetical epistemology, stated that all knowledge has two sources: ideas and things, thoughts and sensations.

The successful rebuttal to the hypothetico-deductive challenge has major implications for the use of inductive methods in qualitative studies. Inclusion of the rich contextual environment that surrounds complex engineering problems is a missing element in the mechanical models of the quantitative method. The ability to develop theories of the middle range using *Discoverers' Induction* is an important step in the acceptance of qualitative methods in engineering research.

Induction and Reliability

The issue of reliability in the inductive method is concerned with the ability of the researcher to consistently apply a common coding scheme as part of the induction. In the inductive element of the research the researcher developed *information groupings* that proposed possible relationships between the empirical data and the idea or conception about the proposed relationship. The *chunks of information* contained in the *information groupings* were organized into themes or subcategories that the researcher felt had significance for the construction of the theoretical framework. The open coding and axial coding techniques in the research procedure are, by nature, highly iterative. The researcher moved back and forth among the open coded nodes and the axial coding *information groupings*, continually refining the information structure or typology. “In

most cases with the close involvement of the researcher, the codes are often based on an intimate knowledge of the field, and almost inevitably carry subjective interpretations.” (Kelle & Laurie, 1995, p. 25) “This is a problem of reliability, since a coding frame would only be regarded as reliable if in any subsequent re-coding exercise the same codes could be applied to the same incidents, which means that the coding could be repeated by a different coder within an acceptable margin of error.” (Kelle & Laurie, 1995, p. 24) To mitigate this concern the researcher ensured that: (1) coding of the empirical data was inclusive and exhaustive; (2) during coding, subcategories and categories were constructed that are mutually exclusive and unambiguous; (3) codes were applied to *chunks of information* in a systematic and consistent manner; and (4) employed an automated code-based theory-building software program to assist in the tasks of retrieving and coding data. All four actions have increased the internal reliability by enforcing repeatable behaviors (Seidel & Kelle, 1995).

Induction and Verification

The deductive element of the positivist position challenges the role of probability in inductive methods. The deductive element of the hypothetico-deductive position is founded upon the principle that hypothesis testing leads to one of two conclusions: either the proposition is false, or it was tested and not yet falsified or corroborated.

Reichenbach, who along with Rudolf Carnap [1891-1970] and Carl G. Hempel [1905-1997] were the leading logical empiricists, wrote to the *Vienna Circle* of Logical Positivists that “. . . the demand for absolute verifiability must be dropped for all synthetic propositions, because else we ought to drop the propositions of science altogether. Instead of that a continuous scale of probability is to be introduced.”

(Reichenbach, 1936, p. 124) In a later work Reichenbach provides an eloquent argument for why probability should be used for all inductive relations.

If the inductive relation from the observed consequences to the theory is once admitted, it cannot be denied that there is likewise an inductive relation which supports the theory 'before' the consequences are tested. The situation of the theory in respect to facts 'before the experimental test.' In both cases there are facts which do not 'verify' the theory, but which may confer a determinate probability on it; this probability may be small before the test, and great after it. But even before the test there must be facts on which the theory is based; and there must be, also before the test, a net of inductive relations leading from the facts to the theory – else the theory could not seriously be maintained. (Reichenbach, 1938, p. 27)

Reichenbach's argument supports the Inductive Treatise that inductively predicated allegories express probabilistic behavior, such that an allegory may predict a phenomenon's behavior under the assumption that it will behave according to certain empirically-generated generalizations with some significant probability (Sutherland, 1973).

Whewell's Antithetical Epistemology

The final challenge to the method may arise from the modern use of Whewell's epistemology. Whewell has been termed antithetical by Butts (1965) and Fisch (1985a, 1985b, 1991) for adopting an epistemology that combines seemingly opposed empirical and *a priori* elements. In a work too large to fully describe in this dissertation Snyder (1994) successfully argues that both Fisch and Butts misunderstood Whewell's philosophy. The key points of her arguments are as follows.

Whewell, in his discussion of philosophy, stated that knowledge is antithetical and consists of inseparable ideal and empirical elements. He went on to state that there is no permanent line to be drawn between *theory* (the ideal element) and *fact* (the empirical element) and stated that a true theory is itself a fact, and can be used to form theories of

even greater generality. His example is that for Kepler it was a theory that the planet Mars revolves in an elliptical orbit about the sun; while for Newton this was a fact that he used in inferring the law of force.

Whewell also applied the antithetical pair concept to experiential and necessary truths. He stated that *experiential truths* are laws of nature that are knowable only empirically and that *necessary truths* are axiomatic propositions knowable *a priori*. These statements are at the heart of the ultimate problem raised by Whewell's antithetical philosophy of science:

How is it possible that through any process, especially one that is partly empirical, a proposition previously knowable only empirically can become knowable a priori?

Whewell claims that science consists in a process called the *idealization of facts*, where experiential truths are *transferred to the side* of necessary truths. Snyder posits that the connection between the empirical (experiential truths) and axiomatic (necessary truths) is *analytic*. Snyder's analytic label refers to Whewell's insistence that human ideas are at the center of the process. It is the fundamental ideas of man that connect the facts of our experience, in a linguistic relationship with real-world objects and events. Whewell further stated that each fundamental idea is made up of *conceptions* which are *special modifications* of the idea applied to the particular circumstances. It is the *Idea* or *Conception* that is central to the formulation of an empirical law. Through the process of *colligation* empirical facts are brought together under a relation provided by the *Idea*.

In summary, the idealization of facts in which experiential truths are shown to satisfy the criteria for necessary truths is the central element of Whewell's epistemology. Whewell's epistemology is the foundation for his method of induction; *Discoverers' Induction*.

CHALLENGES TO THE CASE STUDY METHOD

The case study has long been (and continues to be) stereotyped as a weak sibling among social science methods. Investigators who do case studies are regarded as having downgraded their academic disciplines. Case studies have similarly been denigrated as having insufficient precision (i.e., quantification), objectivity, or rigor. (Yin, 2003, p. xiii)

That is the opening sentence in the Preface to Robert Yin's seminal work; *Case Study Research: Design and Methods*. These words to researchers who contemplate using the Case Study for dissertations serve as sufficient warning about the methodological risk. However, his words are followed by a powerful treatise on how the Case Study is an empirical method that conforms to the Canons of Science. Yin's candid and open approach to the case study provided the researcher with sufficient material to positively approach and mitigate the limitations included in the scholarly criticisms. The principal limitations and the strategies to mitigate the threat to scholarly research were as follows:

Lack of Rigor

The lack of rigor criticism is linked to the problem of bias, which can be introduced by the subjectivity of the researcher (the instrument of the study) and that of the participants on whom the researcher relies to get an understanding of the case. This criticism is not unique to case study research. "Quantitative research can also be affected by the bias of the researcher and of participants: samples can be manipulated, data can be tampered with or purposely excluded, surveys can be poorly constructed and respondents can answer dishonestly." (Patton and Appelbaum, 2003, p. 62). However, case study researchers are not relieved of the responsibility for conducting thorough, careful research. Yin's treatise on the subject is noteworthy in that it has purposely included the subtitle *Design and Methods* because it includes design procedures and methods to ensure

that the case study researcher is able to systematically approach the case study.

Additional formal methods and procedures have been developed, particularly the work of Miles and Huberman (1994), which address the collection and analysis of qualitative data and were essential elements of the case study design in this research. The United States General Accounting Office (1990) has a checklist for reviewing case study reports that is a useful tool in ensuring that the case study report contains sufficient rigor. The use of systematic methods and formal procedures were strategies adopted by the researcher to mitigate this criticism. However, “. . . systematic and careful work is always relevant, no matter the type of research.” (Stenbacka, 2001, p. 553)

Little Basis for Scientific Generalization

The criticism about scientific generalization is focused upon the method’s reliance upon a case study as the means for generalization. “Case studies are generalizable to theoretical propositions and not to populations or universes.” (Yin, 2003, p. 10) The scientific basis for case study generalization is differentiated from experiment generalization in Table 15.

	Experiment	Case Study
Conceptual Frame	A population of data	A theoretical proposition
Method of Generalization	Statistical Generalization	Analytic Generalization

Table 15: Scientific Basis for Generalization

“In analytic generalization, the investigator is striving to generalize a particular set of results to some broader theory.” (Yin, 2003, p. 37) Analytic generalization involves generalizing to a theory or in this case a framework—not to a population—and is based on validating theory-driven or framework-driven predictions with evidence collected in a variety of settings in the case studies. Analytic generalization can reveal contextual conditions under which the framework-based predictions are considered to

apply and serve to increase confidence in the theory or framework. Firestone (1993) stated “. . . analytic generalization attempts to show that a theory holds broadly across a wide variety of circumstances, but sometimes it identifies the scope of a theory — that is, the conditions under which it applies.” (p. 17) In this research the inductively developed systems-based framework was used as a template for comparing the empirical results (i.e. data) of both case studies. The research further extended this generalization by combining the inductive concepts generated in the literature review with insights from existing formal theory. The formal theory came from the literature which was a strategy recommended by Glaser and Strauss (1967).

Lee & Baskerville (2003) have constructed a generalizability framework to offer a clarification as to the type of generalizability appropriate to a specific research effort.

Figure 19 is the framework.

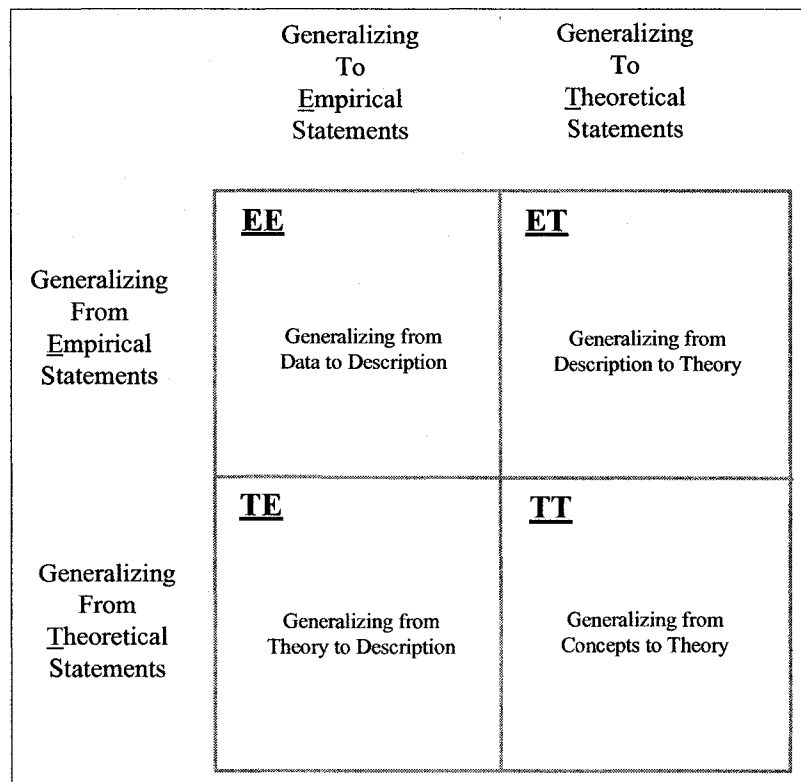


Figure 19: Lee & Baskerville's Generalizability Framework

The type of generalization used in this research is generalizing from description (the empirical results of the case studies) to theory (the inductively developed systems-based framework) and as such was characterized as Type ET. This type of generalizability has appeared in the literature with regularity and is reported by Yin (2003), Klein & Myers (1999), Strauss & Corbin (1998), Walsham, (1995), Dutton & Dukerich (1991), Leonard-Barton (1990), Eisenhardt (1989), and Glaser & Strauss (1967).

Yin provides three synonyms for generalizing from empirical to theoretical statements: “(1) Analytical generalization, (2) Level-2 inference, and, (3) Generalizing to theory.” (Lee & Baskerville, 2003, p. 236) Figure 20 illustrates three examples of generalizing from empirical to theoretical statements (Yin, 2003):

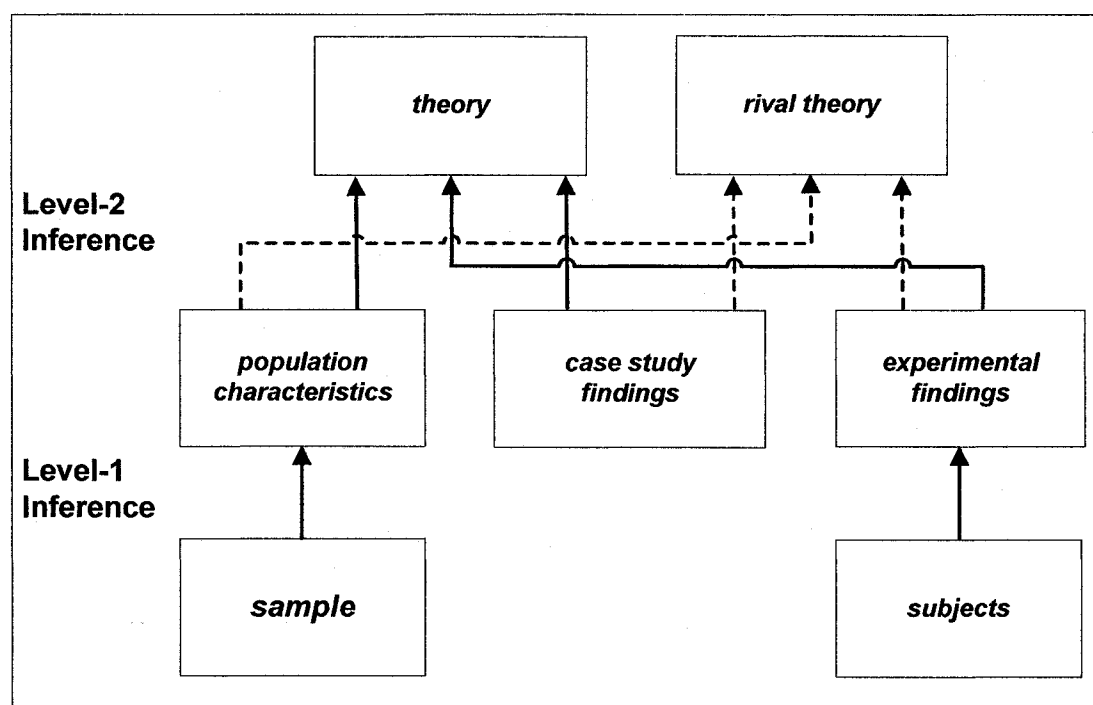


Figure 20: Making Inferences: Two Levels

Adapted from Yin, R.K. (2003). *Case Study Research: Design and Methods*. (3rd ed.). Thousand Oaks: Sage Publications, p. 32

1. Generalizing from population characteristics to theory,

2. Generalizing from case study findings to theory, and
3. Generalizing from experimental findings to theory.

The research developed a Level-2 inference in which the developed theory in the inductively developed systems-based framework was used as a template with which to compare the empirical results of the two case studies. Two case studies were selected because (1) if they support the theory, then replication may be claimed, and (2) more importantly, “. . . the empirical results may be considered yet more potent if two or more cases support the same theory and do not support an equally plausible, *rival* theory.”

(Yin, 2003, p. 33) This is the Level-2 inference in Figure 20.

Analytic generalization is an important concept for the case study researcher.

Type ET generalizability is considered to be well developed. “Hence, criticisms that case studies and qualitative studies are not generalizable would be incorrectly ruling out the generalizability of empirical descriptions to theory. Furthermore, such criticism could be incorrectly presuming that statistical generalizability is the only form of generalizability and will be included as an essential element of the case study design.” (Lee & Baskerville, 2003, p. 237)

Making Controlled Observations

Most natural science experiments routinely observe the influence of one factor on another factor. In so doing they have designed the experiment so that other factors that may potentially interfere with the experiment are strictly controlled or removed from the experiment altogether. The real-world setting for case studies does not permit such controls.

In order to mitigate this criticism the case study researcher has made controlled observations by utilizing natural conditions to validate predictions. Lee (1989a) remarks that:

The case researcher must actively apply his or her ingenuity in order to derive predictions that take advantage of natural controls and treatments either already in place or likely to occur. (p. 39)

The researcher scanned all of his objectively collected data for the presence of natural controls included in the observations.

Take Too Long and Result in Massive, Unreadable Documents

Yin reports that this complaint may be appropriate, given the way case studies have been done in the past, but this is not necessarily the way case studies must be done in the future (Yin, 2003). Yin's book includes a chapter that discusses an alternative method to the traditional, lengthy narrative, and how it can be avoided.

Table 16 lists each of the issues and the mitigation strategies that were included as purposeful elements in the case study design for this research.

Scholarly Criticisms of Case Studies	Mitigation Strategies	Research Design References
Lack of Rigor	1. Use of formal methodology 2. Use of formal data analysis methods 3. Use a guide to achieving quality in qualitative research 4. Case study review checklist	1. Yin, (2003) 2. Miles & Huberman (1994) 3. Cepeda & Martin (2005) 4. U.S. GAO (1990)
Little Basis for Scientific Generalization	1. Analytic generalization to theoretic propositions 2. Generalizability framework 3. Level-2 inference	1. Yin (2003) 2. Lee & Baskerville (2003)
Making Controlled Observations	1. Take advantage of natural conditions to validate predictions. 2. Use formal methods	1. Lee (1989a, 1989b) 2. Strauss & Corbin (1967) 3. Corbin & Strauss (1990)
Take too long and result in massive unreadable documents	1. Adopt alternative methods to the traditional, lengthy narrative	1. Yin (2003) 2. Darke, Shanks & Broadbent (1998)

Table 16: Case Study Criticisms and Mitigation Strategies

COMPLIANCE WITH THE CANONS OF SCIENCE

This section will conclude the discourse on the research methodology by showing how the ontological, epistemological, methodological, and axiological elements of the researcher's view combined to produce a belief system or paradigm that satisfied the generally accepted criteria for high-quality research.

The Canons of Science

"The major justification for the research enterprise is that we have the time and skills to develop approximations of the truth that have a firmer warrant than common sense." (Firestone, 1990, p. 123) The approximation of truth spans the paradigmatic continuum from subjectivist idealism to objectivist realism and includes "... fundamental properties which are regarded as essential for any empirical science whatsoever." (Kaufmann, 1942, pp. 458-459) Four generally accepted criteria for high-quality research compose the *Canons of Science* and answer the following questions (Guba & Lincoln, 1985, p. 290):

1. Truth Value: How can one establish confidence in the *truth* of the findings of a particular inquiry for the subjects (respondents) with which and the context in which the inquiry was carried out?

2. Applicability: How can one determine the extent to which the findings of a particular inquiry have applicability in other contexts or with other subjects (respondents)?

3. Consistency: How can one determine whether the findings of an inquiry would be repeated if the inquiry were replicated with the same (or similar) subjects (respondents) in the same (or similar) context?

4. Neutrality: How can one establish the degree to which the findings of an inquiry are determined by the subjects (respondents) and conditions of the inquiry and not by the biases, motivations, interests, or perspectives of the inquirer?

Researchers grounded in both quantitative and qualitative methods can rely on these higher-level Canons of Science to ultimately arrive at well-reasoned conclusions. Felix Kaufmann [1985-1949], in his paper *The Logical Rules of Scientific Procedure*, addresses the requirement for a researcher to “. . . formulate the rules which the scientist wants to comply with to make the implicitly implied canons of inquiry specific.” (1942, p. 458) Strauss and Corbin (1998) address qualitative research and suggest that:

. . . the usual canons of good science should be retained, but require redefinition in order to fit the realities of qualitative research, and the complexities of social phenomenon that we seek to understand. (p. 250)

A large number of scholars have contributed specific criteria to satisfy the Canons of Science from a number of paradigm positions (see Denzin & Lincoln, 2000, p22; Lincoln & Guba, 2000, p. 166; Whittemore, Chase & Mandle, 2001, p. 529 for complete listings). Table 17 shows the Canons of Science and the generally accepted design quality concepts that are invoked for both the positivist and naturalist paradigms and the associated quantitative and qualitative research methods.

Canon of Science	Quantitative Research Methods and Positivist Paradigm	Qualitative Research Methods and Naturalist Paradigm
1. <i>Truth Value</i>	Internal Validity	Trustworthiness or Credibility
2. <i>Applicability</i>	External Validity or Generalizability	Transferability
3. <i>Consistency</i>	Reliability	Dependability or Auditability
4. <i>Neutrality</i>	Objectivity or External Reliability	Confirmability of Data

Table 17: Canons of Science and Design Quality Concepts

The Canons of Science and the Research Study

The design quality concepts that were invoked within the Canons of Science for the this research were as follows:

Internal Validity (Truth Value)

Ducasse (1951a, 1951b) reports that an important element of Whewell's concept of the Inductive process included the decomposition of facts where Whewell (1858) states:

When we inquire, what facts are to be made the materials of Science, perhaps the answer which we should most commonly receive would be, that they must be True Facts, as distinguished from any mere inferences or opinions of our own. (p. 51)

The internal validity of the framework was reinforced through the use of an outside expert where all of the objectively collected data was evaluated to ensure that personal bias had not entered the collection process.

“The concern over internal validity, for the case study element of the research, may be extended to the broader problem of making inferences. Basically, a case study involves an inference every time an event cannot be directly observed.” (Yin, 2003, p. 36) The researcher endeavored to show the plausibility of the research findings against the relationships contained in the research questions. Triangulation, the combination of research techniques, was included as a purposeful element of the research design. “The effectiveness of triangulation rests on the premise that the weaknesses in each single method will be compensated by the counter-balancing strengths of another.” (Jick, 1979, p. 604) *Data Triangulation* was achieved by collecting data from different sources over different time-scales by doing multiple case studies. *Theoretical Triangulation* was invoked by applying systems principles to the discipline of software engineering.

Method Triangulation was included through the use of both qualitative and quantitative methods. The use of multiple methods of triangulation ensured that the research was more robust and valid (White, 2000).

External Validity or Generalizability (Applicability)

External validity refers to the extent to which the research results may apply to situations beyond the immediate research. For generalizations related to inductive reasoning Hans Selye [1907-1982] wrote:

Those who object to inductive reasoning do not realize that what they actually deplore is the unwarranted confidence in a general law. To inspire confidence, a generalization must be based on as many observations as possible. However, once formulated on the basis of a given number of data, it is neither more nor less likely to be correct as a general law than as a guide permitting correct deductions in new particular instances. (1964, p. 283)

In order to ensure the external validity of the inductively developed framework a panel of experts was used to evaluate the both content and face validity of the framework elements.

For case studies a mode of generalization called analytic generalization (which is contrasted against the well know statistical generalization) was used. “In analytical generalization, the investigator is striving to generalize a particular set of results to some broader theory.” (Yin, 2003, p. 37) For this research it was the inductively developed systems-based framework for the analysis of software development project performance which was used as a template for comparing the empirical results of the case study data; following the Type ET Generalizability reported by Lee and Baskerville (2003). External validity was supported through the use of replication logic in the form of a multiple-case design ($n > 1$). “The use of multiple cases is desirable because it allows for cross-case analysis and the extension of theory.” (Benbasat, Goldstein & Mead, 1987, p. 373)

Reliability (Consistency)

The goal of reliability is “. . . whether the study is consistent, reasonable stable over time and across researchers and methods.” (Miles & Huberman, 1994, p. 278) The objective quality evidence for reliability is the ability of a future researcher to follow the same procedures described by the initial researcher, on the same case study, and arrive at the same findings and conclusions. To ensure replication and controls, both elements of the research, like any other empirical experiment must:

Make as many steps as operational as possible and to conduct research as if someone were always looking over your shoulder. (Yin, 2003, p. 38)

Reliability ensures the congruence between the research problem and the data, methods and analysis techniques used by the researcher. The detailed procedure in Chapter 4 was provided to ensure methodological reliability.

Objectivity (Neutrality)

Objectivity addresses “. . . the issue of whether independent researchers would discover the same phenomena or generate the same constructs in the same or similar settings.” (LeCompte & Goetz, 1982, p. 32) The external reliability of the inductive element of the study was enhanced by addressing four areas recommended by LeCompte & Goetz (1982). (1) The researcher’s role in the study was mitigated through the use of an outside expert and panel of experts. (2) “Every concept brought into the study or discovered in the research process was at first considered provisional. Each concept earns its way into the theory by repeatedly being present in interviews, documents, and observations in one form or another – or by being significantly absent.” (Corbin & Strauss, 1990, p. 7). In addition, the data chosen for the induction was reviewed by an outside expert

prior to its inclusion. (3) The analytic constructs of the framework were developed as part of a detailed procedure in Chapter 4. The organization of the empirical data and resulting stream of subcategories, categories, and constructs were linked to the theoretical foundation for the induction. (4) The methods of data collection and analysis were supported by precise identification and thorough description of the collection and analysis processes.

The case study element of the research also addressed objectivity in three ways. (1) By utilizing multiple sources of evidence during data collection, which provided for both converging lines of inquiry (that permit data triangulation and ensure internal validity) and multiple measures of the same phenomenon; (2) by establishing a chain of evidence during data collection; and (3) having key stakeholders review the draft case study report.

In summary, the four relevant quality concepts were invoked throughout the study. They started in design and were present in data collection, data analysis and the publication phases of the research study. The four quality indicators were measures of the research and its conformance to the Canons of Science.

SUMMARY

This chapter has described the high-level research and dissertation concept and provided a detailed description of the research paradigm in terms of both the researcher's view and the problem under study. The linkage between the researcher's view, the problem under study, and the Canons of Science (see Figure 14) is of significance because it frames the research study and all of the elements that influence the study. The chapter addresses each element to explicitly state the researcher's perspective. It has

provided rationale for the selection of the mixed-method design and reviewed the challenges presented by both the inductive and case study elements of the research and how the research complied with the Canons of Science.

The chapter has explicitly addressed the challenges to the inductive and case study elements of the research methodology and shown how each method satisfactorily complies with the Canons of Science. The four generally accepted criteria for high-quality research (Guba & Lincoln, 1985) were used to judge the adequacy of the research study. Each of the four criteria were examined without reference to either the qualitative/quantitative method or the positivist/naturalist paradigm. The design quality concepts invoked for the research satisfactorily complied with the Canons of Science.

The generalized methodology and paradigms described in this chapter provide the methodological support for the following chapter. The next chapter will provide an outline of the research design and the specific details of the methods, procedures, and techniques used in the two primary elements of the research.

CHAPTER IV: RESEARCH DESIGN, METHODS, AND PROCEDURE

This chapter discusses the assumptions and rationale that support the selection of the research method, lays out the research design and concludes with a discussion of the procedures and techniques of the two primary elements of the research.

THE RESEARCH DESIGN

Beginning with the formulation of the research purpose, as articulated in the research questions and propositions, the research plan moved through framework development, data requirements and structure, data collection and analysis, and publication. Other key aspects of the research plan included the role and influence of scholarly and professional literature and the Canons of Science and Research Paradigm. Figure 21 is the high-level research design, methodological elements, and study phases.

Qualitative Element of the Research Design

Phase 0 - Research questions and propositions

Definition of the principal research questions was the goal of this phase. The research addressed both the substance (what is the study about?) and form (*who, what, where, why, or how* questions?) during the development of the questions. Propositions directed the research focus to what was examined within the scope of the study. Stating specific propositions ensured that the research moved in the right direction and acted as a starting point for relevant evidence. The propositions served as a blueprint for the study and provided strong guidance in determining what data to collect and the strategies for analyzing the data.

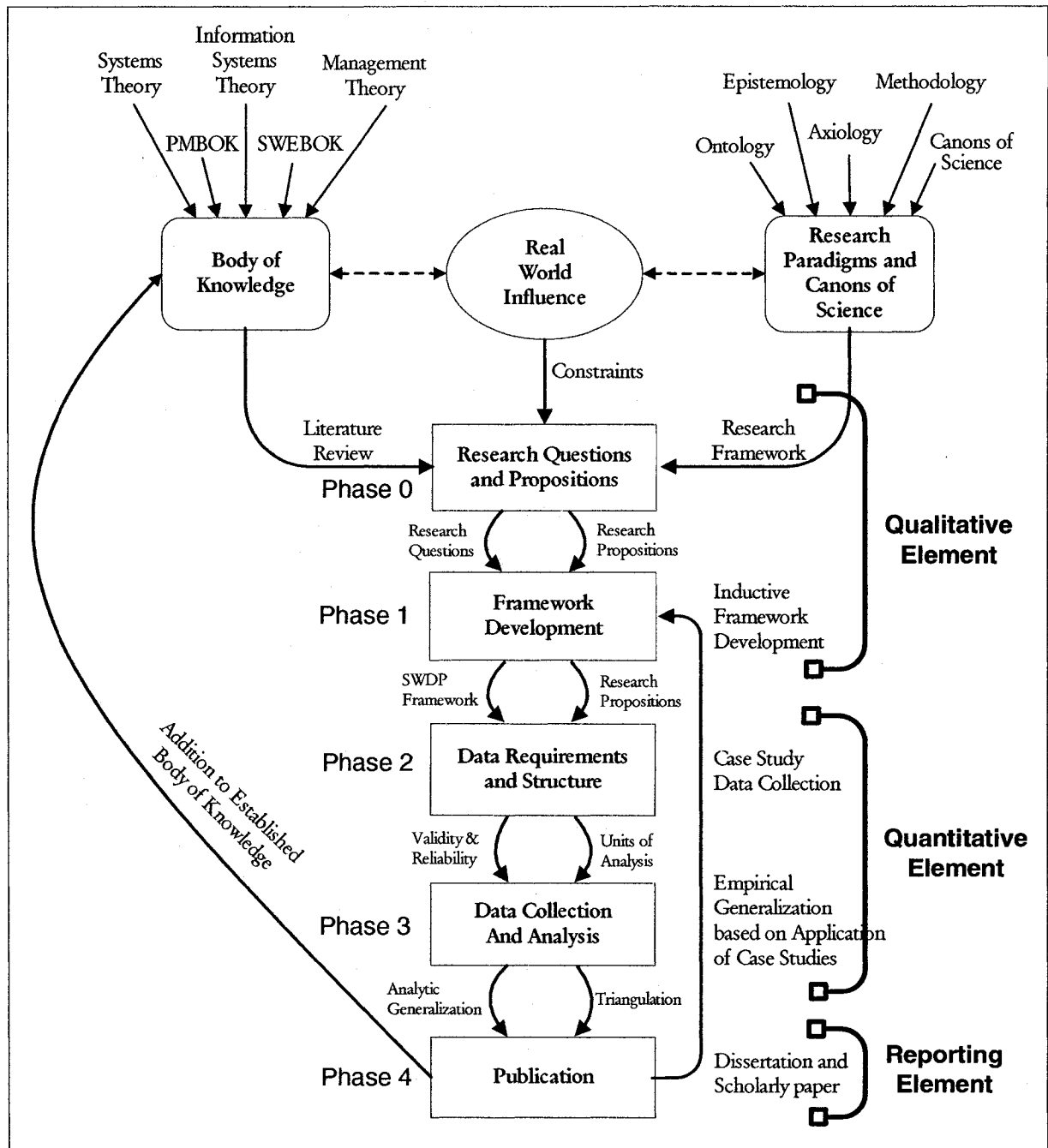


Figure 21: High-Level Research Design and Study Phases

Phase 1 - Framework Development

In this phase initial theory was developed from the literature. The initial idea or conception regarding a holistic, structured, systemic framework for software development projects was the object of the study. This phase was highly qualitative and

relied on inductive theory building to construct the software systems engineering framework. “For case studies, theory development as part of the design phase is essential, whether the ensuing case study’s purpose is to develop or test theory.” (Yin, 2003, p. 28) The framework was validated using the quantitative case study method of Yin (2003), described in the next phase of the research design.

Quantitative Element of the Research Design

Phase 2 - Data Requirements and Structure

The researcher selected two software development projects for use as validating case studies. The criteria utilized for the selection of each of the case studies was an important element of the research as the criteria have a direct impact on the ability to make generalizations based on the findings of the study. Once selected, the intrinsic characteristics of each software development projects were defined. The high-level characteristics of each project were captured and serve as a classification guide and measure of comparison for future research.

In order to avoid being overwhelmed with mountains of data, an analytic strategy that specified who and what was and was not studied was constructed using the guidelines developed by Miles and Huberman (1994). The analytic strategy included several specific methods for data collection and analysis related to software engineering (Seaman, 1999). Throughout, particular attention was paid to the data design to ensure that the measures of design quality in Table 14 were met throughout the study.

Phase 3 – Data Collection and Analysis

This phase of the research design was centered on analysis of the empirical data from the two case studies. Case studies use a mode of generalization called analytic generalization, which is contrasted against the well know statistical generalization (Yin,

2003). In analytic generalization the framework developed in Phase 1 was used as a template for comparing the empirical results of the case study data specified in Phase 2. Yin recommends an information systems research paper by Markus (1983) as an excellent example of a case study that was used to validate theory.

Two cases were used to permit cross-case analysis and the extension of theory (Benbasat, Goldstein & Mead, 1987). Triangulation, the combination of research techniques, was included as a purposeful element of the research design. The use of multiple methods of triangulation ensured that the research was more robust and valid (White, 2000). Methods included: (1) Data Triangulation, which was achieved by collecting data from different sources over different time-scales using multiple case studies; (2) Theoretical Triangulation, which was invoked by applying systems principles to the discipline of software engineering; and (3) Method Triangulation which was included through the use multiple techniques for gathering sources of evidence for the case studies (Zelditch, 1962).

A comprehensive set of rules, presented in the research procedure were applied throughout data collection and analysis. This tact provided an additional level of rigor to the technique and further mitigated many of the criticisms focused on case study research.

Reporting Element of the Research Design

Phase 4 - Publication

The final phase of the research design coherently published the research findings. This dissertation was the principal publication. A secondary publication, in the form of an article in a scholarly journal, will be produced in order to extend the research findings to a wider audience.

Summary of the Research Design

Although the representation of the research design in Figure 21 shows a linear progression of research study phases, it is also beneficial to look at the research from the perspective of the research purpose and research questions. In doing so, the research consisted of two primary elements: qualitative framework development using inductive theory building and quantitative framework validation using two case studies. A clear distinction between the two methodological elements was made because the activities, techniques, procedures, and methods used, and the outputs are different for each. Table 18 describes the major elements of the research.

Research Element	Data Collection Methods	Data Collection Reference	Data Analysis Methods	Data Analysis Reference	Expected Outputs	Relationship to Research Questions
Theoretical Framework Development	Literature review using <i>Discoverers' Induction</i> .	Snyder (1997a) Ducasse (1951a, 1951b) Miles & Huberman, (1994)	Inductive Theory Building using <i>Discoverers' Induction</i>	Carlile and Christensen (2005) Corbin & Strauss (1990) Glaser & Strauss (1967) Strauss & Corbin (1998)	Holistic, structured framework for software development projects	Supports Research Question #1
Framework Validation	Case study method	Yin (2003) Miles & Huberman (1994)	Case Study	Yin (2003)	Framework Validation	Supports Research Question #2

Table 18: Elements of the Research Design

In summary, the research design presented in this section is a compilation of the established body of knowledge on the subject. The design invoked the Canons of Science as measures of design quality, conformed with the rules for qualitative analysis (Munck, 1998), followed the procedures for qualitative data analysis (Miles and Huberman, 1994), and invoked the rigor of the empirical method for case study research (Yin, 2003). The use of these systematic methods and formal procedures were strategies to mitigate criticisms leveled at methods, procedures, and techniques, but it was up to the

researcher to do systematic and careful work. The two major sections that follow will discuss the methods and procedures used in each research element.

METHOD FOR THE THEORETICAL FRAMEWORK DEVELOPMENT

The holistic, structured, systemic framework for software development projects developed in this research element not only provided the conceptual basis for understanding the context surrounding software development projects, but supported the development of formal methodologies that can be used by software practitioners to improve software development project performance. The strength of the framework was based upon being grounded in the theoretical constructs derived from the application of systems theory to software development projects.

Development of the framework used an inductive method called *Discoverers' Induction*, with the categories, attributes, relationships, properties, and dimensions of the framework drawn directly from the inductive theory building method. In order to better understand this research element a review of the theoretical basis for and procedure used in building the framework are warranted.

Models, Frameworks, and Theory

A framework, in the context of this research is a type of model; a conceptual model that can be applied to carry out some specific purpose, function or task. Modeling theorist Peter Achinstein (1965) states that

Models may refer to anything from a physical construction in a display case to an abstract set of ideas . . . a consideration of them will illuminate the structure, interpretation, and development of scientific thinking. (p. 102)

A scientific model is defined to be (Bailer-Jones, 2003):

An interpretive description of a phenomenon (object or process) that facilitates perceptual as well as intellectual access to that phenomenon. 'Description' is intended as a term wide enough to admit various forms of external representations, propositional or non-propositional. A model is not, however, a description in the trivial sense of a mere phenomenological description of a phenomenon. It gives a description that is an interpretation in that the description goes beyond what 'meets the eye', e.g. by exploiting a theoretical background that is relevant to interpreting the phenomenon. (p. 61)

This is an important definition because “. . . scientific models are often contrasted with scientific theories.” (Nagel, 1961, pp. 96-97) However, “. . . theories are not about the empirical world in the same concrete sense as models . . . models, by their very constitution, are applied to concrete empirical phenomena, whereas theories are not.” (Bailer-Jones, 2003, pp. 61-62) The conceptual model, or framework, developed in this research was generated using systems principles, systems theory, systems thinking, and systems practice to explain the relationship to software development project performance. “The use of a framework allows us to express a greater number and a larger variety of *observational facts* and – *this is crucial* – *to explain these facts*.” (Maxwell, 1962, p. 136) Frameworks can include representations that range from localized observations to highly abstracted global generalizations. Figure 22 is the framework representation continuum that shows the relationship of frameworks to data, models, theories and paradigms.

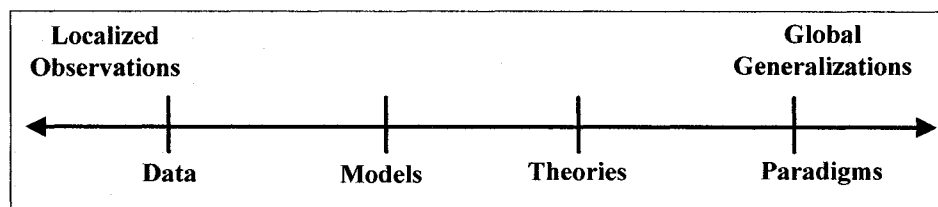


Figure 22: Framework Representation Continuum

Adapted from a Figure in Pemberton, M.A. (1993). "Modeling Theory and Composing Process Models," *College Composition and Communication*, Vol. 44, No. 1, p. 43.

The term *representation* points to characteristics of scientific models that *cannot* be captured in an account that exclusively relies on only propositions (Bailer-Jones, 2003). Giere (2004) states that “. . . scientific representation is to be understood as a two-place relationship between statements and the world. A focus on the activity of representing fits comfortably with a model-based understanding of scientific theories.” (pp. 743-744)

Figure 23 relates the roles of models with principles and generalized hypotheses and

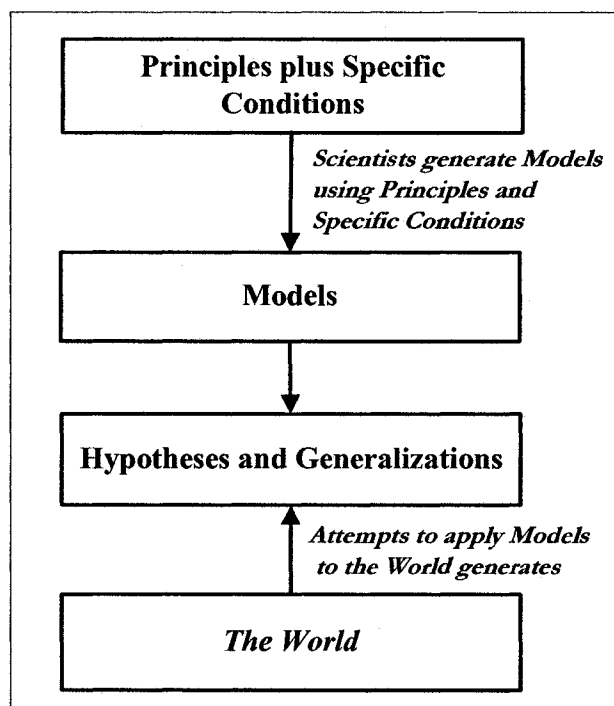


Figure 23: Model-based Understanding of Theories

Adapted from a Figure in Giere, R.N. (2004). "How Models are Used to Represent Reality," *Philosophy of Science*, Vol. 71, No. 5, p. 744.

shows: (1) how scientists generate models using principles and specific conditions; (2) how the application of models to the world generates hypotheses about the fit of the model to particular things in the world; and (3) how hypotheses may be generalized across previously designated classes of objects. The next section will discuss the role of the framework in the development of theory.

Theory Development and Theorizing

Runkel and Runkel (1984) remind us that “. . . theory belongs to the family of words that include guess, speculation, supposition, conjecture, proposition, hypothesis, conception, explanation, and model.” (pp. 129-130) Karl Weick (1995) states that a theory is a “. . . continuum rather than a dichotomy,” (p. 386) and differentiates between theory and theorizing as:

Theory work can take a variety of forms, because theory itself is a continuum, and because most verbally expressed theory leaves tacit some key portions of the originating insight. These considerations suggest that it is tough to judge whether something is a theory or not when only the product itself is examined. What one needs to know, instead, is more about the context in which the product lives. This is the process of theorizing. (p. 387)

Weick’s cautionary note to researchers includes his belief that most theories approximate rather than realize the conditions for strong theory. He goes on to state that most products that are labeled theory actually approximate theory and suggest that these approximations take one of four forms described by Robert K. Merton [1910-2003] in Table 19 (Merton, 1968, p. 140; Weick, 1995, pp. 385-386).

Product	Characteristics
General Orientations	Broad frameworks that specify the types of variables people should take into account, without any specification of relationships among these variables.
Analysis of Concepts	Concepts are specified, clarified, and defined but not interrelated.
<i>Post factum</i> Interpretation	Ad hoc hypotheses are derived from a single observation, with no effort to explore alternative explanations or new observations.
Empirical Generalization	An isolated proposition summarizes the relationship between two variables, but further interrelations are not attempted.

Table 19: Forms of Theory and Characteristics

Weick (1989) provides a broad statement about theories when he states that they:

... involve so many assumptions and such a mixture of accuracy and inaccuracy that virtually all conjectures and all selection criteria remain plausible and nothing gets rejected or highlighted. (p. 521)

He counsels those building theories to move toward theories of the middle range or toward theories that are nearly theories. Merton (1968) defines *theories of the middle range* as:

Theories that lie between the minor but necessary working hypotheses that evolve in abundance during day-to-day research and the all-inclusive systematic efforts to develop a unified theory that will explain all the observed uniformities of social behavior, social organization and social change. (p. 39)

Merton states that middle-range theory is principally used to guide empirical inquiry.

The abstractions contained in middle-range theories are close enough to the observed data that they may be incorporated in propositions that can be validated empirically. Weick (1989) believes that the rationale for moving toward middle-range theories is as follows:

Middle range theories are solutions to problems that contain a limited number of assumptions and considerable accuracy and detail in the problem specification. The scope of the problem is also of manageable size. To look for theories of the middle range is to prefigure problems in such a way that a number of opportunities to discover solutions is increased without becoming infinite. (p. 521)

Both Weick (1974) and Bourgeois (1979) address middle-range theorizing. Weick's efforts are directed at moving systems theory from the category of grand theory to the category of theories of the middle range while Bourgeois addresses methodological issues as how to organize the theory-building effort. Bourgeois (1979) suggests that middle-range theoretical work includes the activities in Table 20.

Step	Description
Partitioning of the Field	Clarification of the purpose, objectives, questions and propositions to be answered.
Method of Theory Construction	<ul style="list-style-type: none"> • Inductive Inference. Starts with observations of a set of phenomena, after which one arrives at general conclusions. • Deductive Inference: Starts with general knowledge and predicts a specific observation.
Review of Literature	Selective reading of the writings relevant to one's work, <i>which should include the classics</i> .
Construction of Theory	Generation of a theory through comparative analysis of empirical laws and substantive theories.
Extension of Theory	Generalization.
Metaphysical Elaboration	A receptacle for the occasional intuitions that surface into consciousness as one pursues the theory-building task.
Conclusion	Statements describing the theory.

Table 20: Bourgeois' Theory-Building Format

Freese (1980) proposes two independent strategies for theory construction in Table 21 that can be used to explain and predict the phenomena of real, complex, and usually contemporary social systems.

Features	Generalizing Theoretical Strategy	Pure Theoretical Strategy
Objective	To explain and to generalize about lawful phenomena in <i>open systems</i>	To predict the behavior of lawful phenomena in <i>closed systems</i> .
Structure	Systematic and contains ordinary language	Formal and contains no ordinary language.
Presentation	Nomothetic universal or statistical generalizations of non-limited spatio-temporal scope, having high information content, and describing some regularity that observations of the world should confirm.	Nomothetic statements expressed in universal or statistical form and having high information content, but they are not meant to be generalizations about the world of everyday experience, and describing some regularities that exist in a theoretically possible world but not in the actual world.
Method	Inductive abstraction.	Idealization.
Result	Consolidation of theories or data.	Cumulation of theory some of which could have engineering applications.
Theory Structure	Summarizing information, in an abstract and general form, that can be used to explain or predict particular empirical cases that fall within the scope of the theory.	Describing some idealized state of affairs in a closed system, with laws describing the invariances of the system, and then they are used for calculating changes in the system when other things are equal.
References	Blau (1970) Kelley & Thibault (1978)	White (1970) MacKenzie (1976)

Table 21: Strategies for Theory Construction and Features

Based on the review of the literature on framework models and theory development, the qualitative element used in this research was bounded by the following:

1. The holistic, structured, systemic framework for software development projects addresses open systems and was developed using a generalizing theoretical strategy and an inductive method.
2. The framework is an approximation of theory, residing on the theory continuum as a theory of the middle range, and as such is not expected to be an ultimate outcome of theory.

The next section will describe the theoretical basis for the inductive method, using both the classics of science and modern interpretations.

Theoretical Basis for Inductive Theory Building

W. Stanley Jevons [1835-1882] writes, in his most important published work, *The Principles of Science* (1877/1913):

Induction is the inference of general from particular facts . . . induction is, in fact, the inverse operation of deduction, and cannot be conceived to exist without the corresponding operation, so that the question of relative importance cannot arise. (p. 121)

And goes on to state the important characteristics of induction:

The truths to be ascertained are more general than the data from which they are drawn. The process by which they are reached is 'analytical,' and consists in separating the complex combinations in which natural phenomena are presented to us, and determining the relations of separate qualities. Given events obeying certain unknown laws, we have to discover the laws obeyed. Instead of the comparatively easy task of finding what effects will follow from a given law, the effects are now given and the law is required. We have to interpret the will by which the conditions of creation were laid down. (p. 212)

John Dewey [1859-1952] gave a concise high-level overview of the inductive method:

With respect to the formulation of the inductive procedures of ancient and modern science respectively there exists a 'verbal' similarity. Both start from scattered data (or particulars) and move toward institution of generalizations. But the similarity does not extend beyond the vague formula of 'going from particulars to generals.' For (1) particulars are conceived in radically different ways and (2) the process of 'going,' or the way in which generals are arrived at from particulars, is very different. (1938, p. 422)

Many of the early philosophers subscribed to the inductive method and include Francis Bacon [1561-1626], John Locke [1632-1704], David Hume [1711-1776], and John Stuart Mill [1806-1873]. John Venn [1834-1923] describes the general steps in induction:

Hence we may lay it down generally that a complete process of inductive discovery, -- if we suppose such a process to commence at the point at which an original investigator must be assumed to have started, and not to terminate until a sound and cautious investigator may be supposed to regard the conclusion as proved, -- must contain the three following steps:

- (1) There is first a stroke of insight or creative genius demanded in order to detect the property to be generalized, and possibly also to distinguish the class over which this property is to be generalized. . . .*
- (2) Then follows a more formal process, namely that of generalization . . . it is at this stage that we must claim a place for the so-called Methods of Inductive Enquiry, such as the Methods of Agreement, of Difference, and so forth. . . .*
- (3) Thirdly, there is a final verificatory stage. . . . Now one kind of verification, and, for scientific or logical purposes, the most important kind, consists of a deductive process. We confirm the inferred generalization, or we may even succeed in absolutely demonstrating it, by showing that it follows from a combination of various known laws. . . .*

(1907/1973, pp. 352-354)

The inductive method adopted for this research, *Discoverers' Induction*, adheres to Venn's generalized process.

Discoverers' Induction is an inductive method attributed to William Whewell [1794-1866], a polymath who “. . . ranks among the major figures in 19th-century philosophy of science.” (Laudan, 1981, p. 163) “Indeed, Whewell was the first author

who formulated the structure of science in the way in which it is conceived today.”

(Frank, 1957, p. 303) Venn states that “Whewell’s works are comparatively not much in vogue at the present day, but the aspect of Induction which he thus emphasized is one which we certainly ought to keep in view.” (1907/1973, p. 356) Ducasse comments on Whewell’s importance by stating that “Whewell is the first to formulate a comprehensive and systematic theory of induction throughout in terms of the so-called Newtonian method of Hypothesis – Deduction – Verification.” (1951b, p. 234)

Overview of the Inductive Theory Building Method

Whewell’s method of induction was concerned with the process by which science comes into being and has three distinct processes. The first two processes are distinctly inductive and third process involves verification of the inductively obtained hypotheses.

Whewell’s Process of Colligation

“Colligation is the mental operation of bringing together a number of empirical facts by *superinducing* upon them some idea or conception that unites the facts and renders them capable of being expressed by a general law.” (Snyder, 1997a, p. 585) The process of colligation is the purposeful action in which the researcher supplies something *to* the facts (in this case it is the holistic, structured, systemic view of software development projects), which causes them to be seen from a *new point of view*. Kaplan (1964) writes that “. . . a conception *belongs to* a particular person.” (p. 48) “One of Whewell’s important contributions is his recognition that *finding the proper conception with which to colligate the known facts* is the crucial – and often extremely difficult – step in scientific discovery.” (Snyder, 1997a, p. 586, *her emphasis*) Whewell (1849) remarks that “. . . there is a special process in the mind, in addition to the mere observation of facts, which is necessary.” (p. 40) This special process in *Discoverers’*

Induction is *inductive inference*, specifically, the inference from observations and background information to the hypothesis (Snyder, 1997a). Whewell has designated it a process because finding a property shared by known members of a class typically involves a number of inferences. This series of inferences from observations and contextual information is a *process* which Whewell called a *train of researches*. (1857) The *train of researches* may include enumerative, eliminative, causal, and analogical inferences.

This element of Whewell's method of induction clearly distinguishes him from the other 19th-century philosophers of induction. "Whewell believed with Kant [Immanuel Kant, 1724-1804] in the great importance of the *linguistic material* produced by our minds for the advance of science and contributed in this way a great deal to a better understanding of what the structure of science is." (Frank, 1957, p. 307) *Induction by new concepts* has had a significant impact in the advance of science and modern discovery (Frank, 1957). John Kemeny [1926-1992] commented that induction is clearly a much more useful thing than deduction and stated:

Induction tells us things we did not know before, whereas deduction only tells us things we knew already but did not realize we knew. (Kemeny, 1959, p. 113)

Whewell's Process of Generalization

The second process addresses generalization. When Whewell talked about generalization he was being fairly specific. In this case generalization requires the researcher to find a property in a group of facts (i.e. a property that is shared by the known members of the class) that can be projected onto the unknown members of the class as well. The use of enumerative, eliminative, causal, and analogical reasoning may

be used in reaching the conclusion that observed members of a class share a property and form the basis for generalization to all members of the class. Whewell's inclusion of the process of generalization clearly separates him from the later work of Mill and is the source of much debate (Strong, 1955; Walsh, 1962). "Whewell's inductivism, in contrast to Mill's, involves an inference or series of inferences from observations to a property or cause shared by observed members of the class, which is then generalized to all members, including the unobserved ones. This extra, inferential, element allows for the generalization of a property or cause instances of which may not have been observed, and which may even be unobservable." (Snyder, 1997b, pp. 194-195). John Dewey (1938) stated that *inductive inference* involves extension beyond the scope of already observed objects and the outstanding fact of scientific inductive inference is, namely, controlled reconstitution of the singulars which are the ground of generalizations. "This reconstitution is so effected as to determine what goes on in the way of interaction in a *singular* case. Inference from *one to all* is completely and exclusively determined by prior experimental operations through which the *one* has been determined to be an exemplary specimen of an order of interactions or of functional correlations of variations. This order, when it is ascertained, *is* the generalization." (Dewey, 1938, pp. 439-440) The inclusion of the concept of generalization to the unobservable, widely accepted in modern scientific research, was an essential factor in the selection of Whewell's method for this research.

Whewell's Process of Verification

The third and final process is where the inductively obtained hypothesis is validated by empirical consequences. Whewell's verification criteria included:

prediction, consilience, and coherence (Snyder, 1994). When he included *prediction* he was referring to “. . . the hypothesis’ ability to foretell phenomena which have yet to be observed; at least all phenomena of the same kind as those which the hypothesis was invented to explain.” (Whewell, 1858, p. 86) Whewell also stated that “. . . the prediction of results, even of the same kind as those which have been observed, in new cases, is the proof of real success in our inductive processes.” (Whewell, 1858, p. 87) When Whewell spoke of *consilience* he was referring to the hypothesis’ ability to explain and predict cases of a different kind from those which were contemplated in the initial formation of the hypothesis. When this occurs Whewell termed this a “*Consilience of Inductions*; that is, two laws obtained by independent inductions and concerning apparently heterogeneous classes of phenomena turn out to be, both of them, deducible from one and the same hypothesis.” (Ducasse, 1951b, pp. 229-230) The final criterion, *coherence*, deals with the adequacy of the hypothesis. The hypothesis should be sufficient to explain “. . . some view of the subject which *is* consistent with *all* the observed facts.” (Whewell, 1858, p. 85)

METHOD FOR THE FRAMEWORK VALIDATION

The goal of the second element of the research, and the final process in Whewell’s *Discoverers’ Induction*, was to validate the inductively developed holistic, structured, systemic framework for software development projects using actual real-world software development projects. Framework validation was a deductive act in which the researcher explored whether or not the same relationships existed between the framework attributes and outcomes by using a different set of evidence (i.e. the case studies) from which the framework was induced. This was the principal output of phases

2 and 3 in the research design and used a case study methodology to validate the framework. The criteria utilized for the selection of each of the case studies was an important element of the research as the criteria have a direct impact on the generalizations that may be drawn from the findings. Once selected each case was characterized using a formal project model which served as a guide for future researcher's who may study software development projects using the framework.

Theoretical Basis for the Use of Case Studies

Creswell (2003) reports that Case Study research is well suited for issues and questions:

... in which the researcher explores in depth a program, an event, an activity, a process, or one or more individuals. The case(s) are bounded by time and activity, and researchers collect detailed information using a variety of data collection procedures over a sustained period of time. (p. 15)

Leedy and Ormrod (2001) state:

A case study may be especially suitable for learning more about a little known or poorly understood situation. It may also be useful for investigating how an individual or program changes over time, perhaps as the result of certain circumstances or interventions. In either event, it is useful for generating or providing preliminary support for hypotheses. (p. 149)

Yin states that a case study is an empirical inquiry that (2003, p. 13-14):

- ✓ *Investigates a contemporary phenomenon within its real-life context especially when the boundaries between the phenomenon and context are not clearly evident.*
- ✓ *Copes with the technically distinctive situation in which there will be many more variables of interest than data points, and as one result relies on multiple sources of evidence, with data needing to converge in a triangulating fashion, and as another result.*
- ✓ *Benefits from the prior development of theoretical propositions to guide data collection and analysis.*

Case studies combine data collection methods such as archives, interviews, questionnaires, and observations. The data may be either qualitative (e.g., words) or quantitative (e.g., numbers), or as was the case, both. Case studies are used to provide description, validate theory or generate theory (Eisenhardt, 1989). However, the decision to use a case study approach was not clear cut. Table 22 reviews the terminology used in case study research against the traditional research phase as reported by Yin (2003) and Bonoma (1985).

Research Phase	Yin's Framework (2003)	Bonoma's Framework (1985)
Exploration	Description	Drift
Hypothesis generation	Exploration	Design
Hypothesis validation		
Confirmation	Explanation	Prediction
Disconfirmation	Explanation	Disconfirmation

Table 22: Terminologies used for Research Phases in Case Study Methods

Adapted from Benbasat, I., Goldstein, D.K. & Mead, M. (1987) "The Case Research Strategy in Studies of Information Systems," *MIS Quarterly*, Vol. 11, No. 3, p. 372.

A critical assumption in deciding to use a case study method for validation of the framework was that the boundaries of the case were not clearly evident at the outset of the research and that no experimental control or manipulation was going to be used. Specifically, the researcher had less *a priori* knowledge of what the variables of interest would be and how they were to be measured (Benbasat, Goldstein & Mead, 1987). The distinguishing characteristics of case studies were useful in helping to understand the strengths of the method. Table 23 presents the strengths of case studies against the most common characteristics.

Overview of the Case Study Method

The case study method permitted the researcher to gather extensive evidence from the object of the study. "Evidence may come from six sources: documents, archival records, interviews, direct observation, participant-observation, and physical artifacts."

(Yin, 2003, p. 83) Because the researcher was looking into the past both direct observation and participant-observation were eliminated as potential data collection methods. The evidence from the cases were collected, analyzed, and interpreted. The generalization method for case studies was analytic generalization.

Strength of Case Study Approach	Associated Case Study Characteristic
The ability to investigate a contemporary phenomenon within its real-life context permits the researcher to include the rich human element that surrounds the problem.	Phenomenon is examined in a natural setting.
Case studies are able to use multiple data sources. While no individual source is deemed "best" the use of various sources is a highly complementary practice enabling the researcher to use <i>triangulation</i> as a method for ensuring the quality of the design (i.e., internal validity).	Data are collected by multiple means that include: (1) documentation, (2) archival records, (3) interviews, (4) direct observations, (5) participant-observation, and (6) physical artifacts. (Yin, 2003, p. 85)
Permits the researcher to include multiple views in the analysis.	One or few entities (person, group, or organization) may be included in the study.
The complex nature of the unit of analysis (i.e., a software project) is studied from a variety of perspectives invoking the <i>Principle of Complementarity</i> .	The complexity of the unit of analysis is studied intensively.
The case is viewed in the natural setting and does not require control over behavioral events.	No experimental controls or manipulation are involved.
The researcher does not have to design experiments using control and independent variables.	The investigator may not specify the set of independent and dependent variable in advance.
The method is flexible enough to allow for redesign.	Changes in site selection and data collection methods could take place as the investigator develops new hypotheses.
How and why questions are more explanatory and deal with operational links that need to be traced over time. This is in sharp contrast to the more traditional frequency or incidence of objectivist approaches.	Useful for the study of <i>how</i> and <i>why</i> questions.

Table 23: Strengths of Case Study Approach

Analytic generalization involved generalizing to a theory or in this case a framework—not to a population. The case study evidence was used as the basis for the validation of the framework developed in the previous inductive method. The real-world behaviors discovered in the case studies rendered judgment with respect to the framework's ability to predict performance behaviors based on the frameworks constructs and measurement objects.

Organization of Details about the Case

Formal analysis required an analytic strategy, in this case, one that was broad enough to address the conduct of analysis at the level of the whole case. The case-based analytic strategy relied on the theoretical propositions and research questions that framed the overall research study, forcing them to guide and shape the data collection plan. The analytic strategy had three sections: (1) formulation, (2) quantitative and qualitative analysis, and (3) interpretation. The analytic strategy was the guide for the remaining processes in the case study method.

Collecting the Evidence

The researcher collected evidence from each of the software development projects or cases. This used the 1st section of the analytic strategy, formulation, where the quantitative and qualitative approach used in the analysis was developed. For this research the evidence included documents, archival records, interviews, and physical artifacts. The collection techniques for most of the evidence were very straightforward. However, the use of interviews, in the form of a questionnaire was more problematic, and was fully addressed in Sub-Step 13-2.

Analyzing and Interpreting the Evidence

The researcher organized, analyzed, and interpreted the collected evidence. This process used the 2nd section of the analytic strategy, where quantitative and qualitative analysis served as a guide for the researcher during analysis of the evidence. Seaman recommended the use of coding as a valid method for software engineering studies because coding is able to extract values for quantitative analysis from primary qualitative data (often collected from documentation, records, interviews, and questionnaires) in

order to perform some type of quantitative or statistical analysis (Seaman, 1999). The techniques for open and axial coding axial, from the grounded theory method, were also utilized in this process. The decision to use a software tool to assist with the coding of the data was a function of the large volume of data associated with each of the case studies. The overall goal of this process was to derive meaning from the case study evidence in order to reflect any relationships that may emerge.

Reporting the Case Studies

The researcher was required to bring the results and findings to closure in a report. For this research the dissertation was the principal publication. A secondary publication, in the form of an article in a scholarly journal, will be produced in order to extend the research findings to a wider audience.

THE DETAILED RESEARCH PROCEDURE

The detailed research procedure implemented the research design and supporting methods. As Philosopher Ernest Nagel [1901-1985] stated:

Every branch of inquiry aiming at general laws concerning empirical subject matter must employ a procedure that, if it is not strictly controlled experimentation, has the essential logical functions of experiment in inquiry. This procedure (we shall call it 'controlled investigation') does not require, as does experimentation, either the reproduction at will of the phenomena under study or the overt manipulation of variables, but it closely resembles experimentation in other respects. (1961, p. 452)

The structure for the research design included three high level research elements and five phases. The detailed procedure included two new terms; step and milestone. A step is a specific and unique technique or procedure, taken in conducting the research. A step is the 3rd and lowest level of the research design and supports a phase. A milestone marks a

point in time when either a specific product was delivered or an important decision was made.

Introduction to the Qualitative Procedure

The first nine steps in the procedure developed the framework using Whewell's

Discoverers' Induction. The method for *Discoverers' Induction* followed Venn's general process of induction and was accomplished in a series of three well-defined processes.

Table 24 is the overall structure for the qualitative research element; Whewell's

Discoverers' Induction. It is interesting to note that (1) Christensen's and Raynor's (2003) recent explanation of where theory comes from, (2) Bourgeois' (1979) theory building format in Table 20, and (3) the metatriangulation procedure of Lewis and Grimes (1999) all conform very well to Whewell's three processes.

The researcher augmented Whewell's inductive method with modern qualitative data collection and analysis techniques which facilitated the use of his method as the basis for the research procedure. This followed the pragmatic practice of combining techniques to obtain desired results recommended by Creswell (2003).

The following sections will discuss the detailed steps taken and milestone delivery points during the qualitative element of the research. Phase 0 was not included because the associated step and milestone were fully discussed in Chapter 1.

Procedure for Phase 1: Literature Database for the Induction

The goal of the 1st phase was the assembly, synthesis and verification of empirical facts for the induction. This started when the researcher observed the phenomena under study and carefully described what had been observed. Whewell called this the *Clarification of the Elements of Knowledge by Analysis*, which focused the research

Structure	Definition of the Element
Qualitative Element	Inductive development and verification of a structured, systemic framework for software development.
Phase 0	Research Questions and Propositions. Definition of the principal research questions is the goal of this phase
Step 0	Research Questions: Definition of the principal research questions is the goal of this phase
<i>Milestone 0</i>	<i>Product 0: The formal structure for the inquiry which includes the research purpose, objectives and research questions.</i>
Phase 1	Literature Database for the Induction: The assembly, synthesis and verification of empirical facts for the induction.
Step 1	Selection of the Idea: The proposal of a scientific problem in the form that includes the research purpose, objective and questions.
Step 2	Observation and Collection of Facts: This includes the development of a formal data collection framework, data reduction, and data display. Data collection rules are enforced.
Step 3	Verification of Real-World Facts: A one-time <i>expert review</i> to verify that the literature review in Chapter 2 has provided an appropriate range of ideas, concepts, and theories.
<i>Milestone 1</i>	<i>Product 1: A database of synthesized literature sources for the induction.</i>
Phase 2	Inductive Development of the Framework: The holistic, structured, systemic framework for software development projects developed in this research element.
Step 4	Decomposition of Facts: The empirical facts contained in the synthesized literature review are broken into their basic elements.
Step 5	Classification of Facts: The classification of the collected data in an attempt to simplify and organize the data into information groupings.
Step 6	Construction of the Conception: The development of a theoretical framework regarding the conception and real-world software development project outcomes.
<i>Milestone 2</i>	<i>Product 2: A structured, systemic framework for software development.</i>
Phase 3	Verification of the Framework: The structured, systemic framework is verified to ensure that it contains the requisite procedures and features, and <i>looks like</i> it measures what it was intended to measure.
Step 7	Internal Procedural Verification: A formal feedback loop which permit the theoretical proposition or framework to be verified and/or reintroduced to the process.
Step 8	Internal Feature Verification: The framework is checked for essential features.
Step 9	External Verification: A formal content and face validation of the completed framework is accomplished using a panel of experts.
<i>Milestone 3</i>	<i>Product. A formal verification that the structured, systemic framework accurately represents the real-world phenomena.</i>

Table 24: Structure for the Qualitative Element

effort by establishing boundaries that both constrained and enabled the inductive method.

The principal process was *colligation* – “. . . whereby known facts are connected into a law by the *superinduction* upon them of a conception.” (Snyder, 1999, p. 542) This was a mental operation that focused on an idea or conception supplied by the researcher. This was accomplished in three distinct steps.

Step 1: Selection of the Idea

Whewell (1858) defined a conception as:

The special modification of these ideas which are exemplified in particular facts, we have termed Conceptions; as a circle, a square number, an accelerating force, a neutral combination of elements, a genus. (p. 31)

By explication of concepts he meant:

Their clear development from Fundamental Ideas in the discoverer's mind, as well as their precise expression in the form of Definitions or Axioms when that can be done. (Whewell 1858, p. 49)

This was the process where the researcher, acting as the discoverer, brought an idea to bear upon the formation of knowledge. For Whewell, who was also an Anglican Priest, the source of conceptions was from the fundamental ideas that God had implanted in our minds. For this research the source of the idea or conception was a function of the academic training and real-world experience of the researcher.

Whewell acknowledged that this step was not assisted by a formal method but that the discoverer must ensure that the idea is clear, appropriate and consistent. The idea may be stated as a proposal of a scientific problem in the form of a statement concerning some set of known facts. This step consisted of a "... suggestion of a conception not before apparent which is superinduced upon the facts." (Whewell, 1858, p. 110) This complied with Freese's (1980) notion that theory construction "... typically begins with empirically grounded, systematic discourse expressed in an ordinary language." (p. 191)

Step 2: Observation and Collection of Facts

This step encompassed the literature review and the reduction of information presented in the scholarly journals. "Reviewing relevant literature enhances traditional induction by helping theorist's link emerging theory to extant work recognizing the influence of their own theoretical inclinations." (Lewis & Grimes, 1999, p. 678) The

content and structure of this initial stage of the research created a formal boundary for the research which was clearly stated. The schema for the literature review, the scholarly journals included in the review, and the resulting synthesis were one side of the boundary; the side that ensured that the researcher was exposed to a range of ideas, concepts and theories. The researcher's conceptual lens or worldview formed the other side of the boundary; the side that acted as a filter affecting the importance placed on the observations made by the researcher and the decisions to include or exclude particular elements of the observations. This resulted in "... facts that are both theory-laden and value-laden." (Guba & Lincoln, 1994, p. 105) The researcher was tasked with ensuring that the underlying assumptions and boundaries of the research were made explicit as the outputs of this step were the principal factual information/data sources for the first element of the research.

During this step the empirical data were documented and measured in both words and numbers using formal methods and techniques that were developed which address the collection and analysis of qualitative data. Of particular importance was the conceptual framework for the collection of data (Miles & Huberman, 1994). This *construct* specified who and what was and was not studied and developed the formal relationships that bounded the collection of data. Two concurrent flows of activity occurred in this step: data reduction and data display. Seaman (1999) described several qualitative methods for data collection and analysis and how they might be incorporated into empirical studies of software engineering. Gerardo Munck's (1998) *Canons of Research Design in Qualitative Analysis* contained a summary of the issues that pertained to qualitative and small-N research. Munck offered an extended discussion of the work

by King, Keohane and Verba (1994), a work that has been cited as “. . . one of the best and most important works in social science methodologies.” (Munck, 1998, p. 18; Gerring, 2001, p. 11) The principal element of Munck’s work that was applied to this research was the concept of a research cycle and the methodological rules for qualitative analysis. Munck included specific questions to ensure that the data collected in this step were replicable, reliable and valid. This mitigated many of the criticisms involving data collection in a qualitative research setting, thereby ensuring the validity of the data, which distinguished between the internal validity (truth value) and external validity (generalization) described in the methodology.

Step 3: Verification of Real-World Facts

This step was a one-time feedback loop to verify that the literature review captured all of the relevant information. The information synthesized in the literature review was the source of empirical data for the colligation; and provided an appropriate range of ideas, concepts, and theories. The observation and collection of empirical facts “. . . has a direct affect on the validity of the inductively predicated allegory which depends primarily on the quality of the data base from which the inductive inferences were derived.” (Sutherland, 1973, p. 168) The use of an expert, outside of the researcher, was intended to decrease research risk by ensuring that the information selected by the researcher was adequate enough to provide a firm foundation for the induction. An *expert* is defined as “. . . a person who has background in the subject area and is recognized by his or her peers or those conducting the study as qualified to answer questions. Questions are usually posed to the experts because they cannot be answered

by other means.” (Meyer & Booker, 2001, p. 3) The means for gathering expert judgment usually involve three factors (Meyer & Booker, 2001, p. 7):

1. *selecting experts according to particular criteria,*
2. *designing elicitation methods, and*
3. *specifying the mode in which the expert is to respond*

The procedure for the verification of real-world facts formally addressed each of these factors.

The selection of the expert was governed by both the professional qualifications and availability of the expert. The professional qualifications for the expert reviewer are listed in Table 25.

Qualification	Qualification Criteria
Education	Earned Ph.D. in engineering management, systems engineering, software engineering, or related discipline.
Experience	Greater than 20 years experience with both commercial and government systems and software development methodologies.
Reputation	A recognized expert in software or systems engineering.
Publications	A widely published researcher, author, and speaker.

Table 25: Qualifications for Expert Reviewer

By satisfying the qualifications in this profile the expert added additional validity to the research study.

“Elicitation is the process of gathering the expert judgment through specially designed methods of verbal or written communication.” (Meyer & Booker, 2001, p. 9) In this case the elicitation method was a modified *Delphi* situation in which the expert, isolated from the researcher, provided judgments about the adequacy of the literature for the induction. The mode in which the expert was to respond was specified in the verification guidelines for the review contained in Appendix B. The researcher anticipated that the expert would recommend additional literature sources that would add depth and breadth to the study. Appendix B included a section where the outside expert

recommended additional literature sources. The additional literature sources submitted by the expert became research data with recommended sources being added to the database of empirical facts used for the induction. The additional understanding gathered from these sources proved useful in the development of the framework.

The product at the end of the 1st phase was a database of synthesized literature sources used for the induction.

Procedure for Phase 2: Inductive Development of the Framework

The goal of the 2nd phase was the inductive development of the structured, systemic framework for software development. Whewell called this the *Colligation of Facts by Means of a Conception*. “Whewell’s doctrine of the Colligation of Facts constitutes the most important and most original part of his contribution to the theory of induction.” (Ducasse, 1951b, pp. 217-218) Whewell (1858) defines the term *colligation of facts* as:

To every case in which, by an act of the intellect, we establish a precise connection among the phenomena which are presented to our senses. (p. 60)

Whewell (1858) uses this definition to define Induction as follows:

Induction is a term applied to describe the ‘process’ of a true Colligation of Facts by means of an exact and appropriate conception. An ‘Induction’ is also employed to denote the ‘proposition’ which results from this process. (p. 70)

The importance of this definition cannot be overemphasized. Whewell’s central theme emphasized that in every Induction, there is a conception supplied by the human mind that is *superinduced* upon the facts. Ducasse (1951b) stated that an inductive formula might be something like the following:

These particulars, and all known particulars of the same kind, are exactly expressed by adopting the Conceptions and Statement of the following Proposition. (p. 220)

Ducasse stated that “. . . the all-important requisite for performance of successful inductions is the possession of a fertile, sagacious, ingenious, and honest mind, certain rules and methods of procedure useful in various degrees may be formulated in connection with the colligation of facts mentioned.” (1951b, pp. 221-222). Whewell’s colligation of facts by means of a conception was accomplished in three steps.

Step 4: Decomposition of Facts

Whewell (1858) stated that:

What facts are to be made the materials of Science, perhaps the answer which we should most commonly receive would be, that they must be True Facts, as distinguished from any mere inferences or opinions of our own. (p. 51)

Whewell was following the empiricist tradition in which “. . . a distinction is made between *hard* and *soft* data, according to whether they are purely observational or contain an inferential element.” (Kaplan, 1964, p. 131) In a literature too vast to summarize here, theorists have argued that observation is already cognition and that we cannot describe a fact without implying more than the fact. As a result, Clyde H. Coombs [1912-1988] proposed that the term *data* be used for observations already interpreted in some way. The diagram in Figure 24 depicts the scope of Coombs’ theory of data (1964). Figure 24 shows how the researcher’s interpretation of observables and classification of data lead to logical inferences but has additional import when considered with the following statement (Coombs, 1964):

The scientist enters each of these three phases in a creative way in the sense that alternatives are open to him and his decisions will determine in a significant way the results that will be obtained from the analysis. Each successive phase puts more limiting boundaries on what the results might

be. At the beginning, before phase 1, there are perhaps, no limits on the potential conclusions; but each phase then constrains the universe of possible inferences that can be ultimately drawn from the analysis. (p. 5)

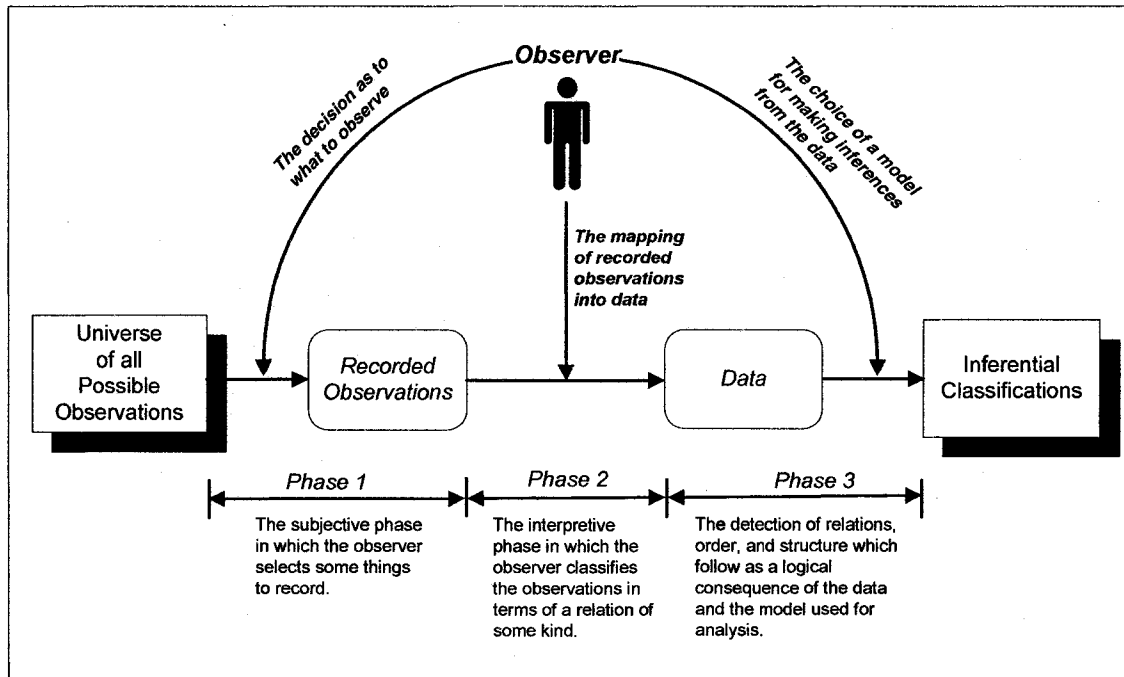


Figure 24: Flow Diagram of Observable to Inference

It is important to note that the researcher depicted in Figure 24 addresses each of the three phases in the following ways.

1. Phase 1 – the decision as to what to observe
2. Phase 2 – the mapping of recorded observations into data
3. Phase 3 – the choice of a model for making inferences from the data

In summary, Coombs' central thesis was that *data* are recorded observations that have already been subjected to analysis.

During this step Whewell stated that the discoverer must strive to decompose the complex facts found in the real-world into their elementary facts. This is where the empirical facts contained in the synthesized literature review were broken into their basic

elements; where information was transformed to data, and data into categories, and categories into properties and dimensions.

To support this step a new word was introduced to the research lexicon; *coding*. “Coding is analysisThis part of analysis involves how you differentiate and combine the data you have retrieved and the reflections you make about this information. *Codes* are tags or labels for assigning units of meaning to the descriptive or inferential information compiled during a study. Codes usually are attached to *chunks* of varying size – words, phrases, sentences, or whole paragraphs, connected or unconnected to a specific setting.” (Miles & Huberman, 1994, p. 56) The formal decomposition technique labeled as *open coding* (specified in the grounded theory method) was useful in this step and is presented in Table 26.

Decomposition Technique	Open Coding
Goal	To discover, name, and categorize phenomena according to their properties and dimensions. (Strauss & Corbin, 1998, p. 206)
Description	The data are scrutinized for commonalities that reflect categories, or themes, within the data. After the data are categorized, they are further examined for properties – specific attributes or subcategories – that characterize each category. In general, open coding is a process of reducing the data to a small set of themes that appear to describe the phenomenon under investigation. (Leedy & Ormrod, 2001, pp. 154-155)
Variations	<ul style="list-style-type: none"> • Line-by-line analysis • Analysis of a whole sentence or paragraph • Peruse the entire document (Strauss & Corbin, 1998, pp. 119-120)

Table 26: Characteristics of Open Coding

Figure 25 is the schema for the decomposition of facts that shows how the body of knowledge was reduced first by the study purpose and second by the literature review. The synthesis conducted in the literature review resulted in a number of information threads that populated the document database with an appropriate range of factual ideas,

concepts, and theories which acted as the empirical data for the colligation. Figure 25 depicts the hierarchical nature of the facts and how they were decomposed into elementary properties and dimensions. This step was enhanced through the use of a code-based theory building tool discussed in the final section of this chapter.

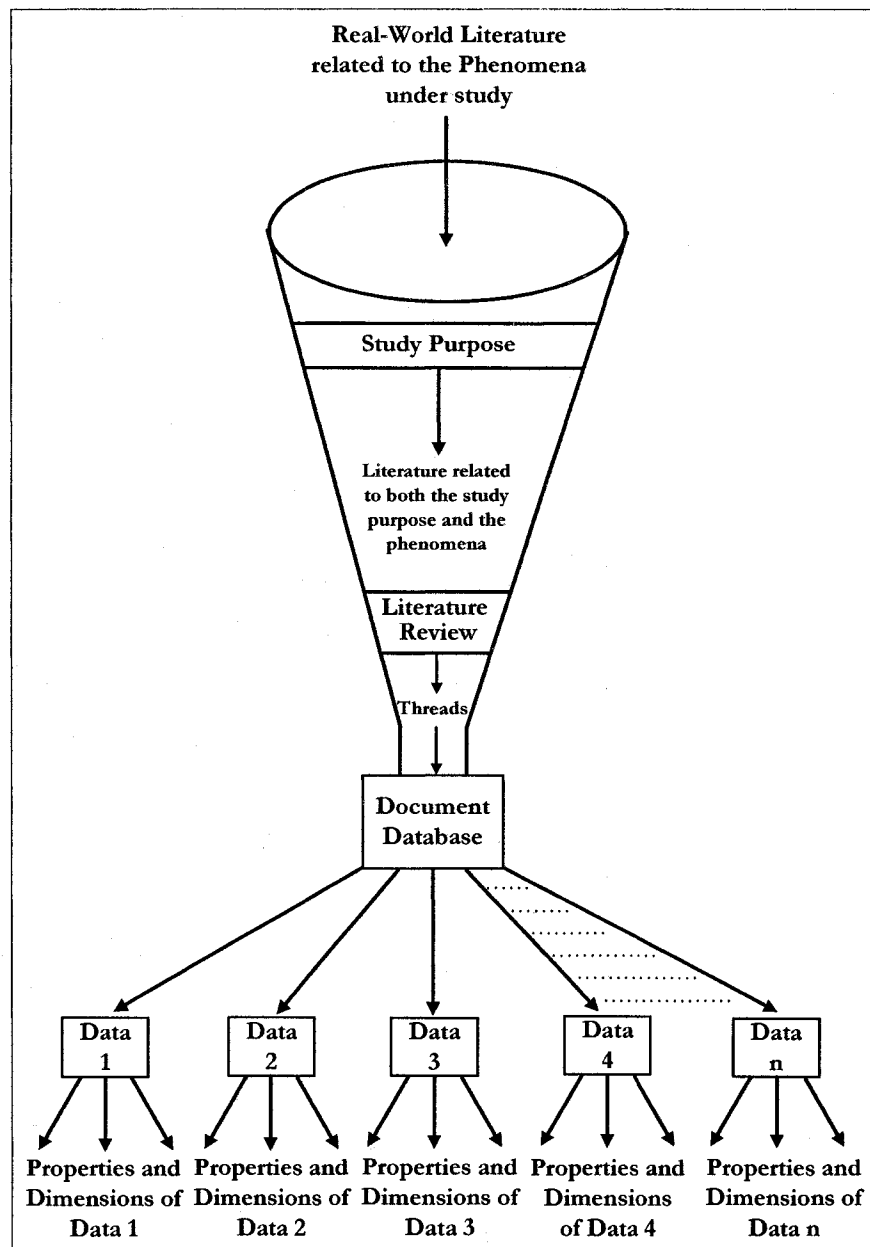


Figure 25: Schema for the Decomposition of Facts

Step 5: Classification of Facts

The empirical data of the observed phenomena, which Whewell called facts of quantity or facts of resemblance, were classified into relevant categories. Jevons (1877/1913) commented that “. . . classification is not really distinct from the process of perfect induction, whereby we endeavor to ascertain the connexions [*sic*] existing between properties of the objects under treatment.” (1877/1913, p. 675) He went on to state that it was impossible to lay down specific rules or procedures but offered the following logical rule:

Having given certain objects, group them in every way in which they can be grouped, and then observe in which method of grouping the correlation of properties is most conspicuously manifested. (Jevons, 1877/1913, p. 690)

The initial classification schema was defined by the natural attributes of the phenomena. This classification schema was used in an attempt to simplify and organize the data properties and dimensions into information groupings that proposed possible relationships between the observed phenomena and the idea or conception that served as the basis for the framework under development. In most cases the information groupings were descriptive typologies. Mintzberg (1979) places his trust in typologies over taxonomies and states:

To generate those configurations, I have more faith in typologies than taxonomies, if I understand correctly how these terms are used. In other words, while I believe we need empirical data to generate our categories – systematic data reinforced by a good deal of anecdote – I do not expect them to come from mechanical data reduction techniques. It is pattern recognition that we are after, in the form of those creative leaps. (p. 588)

Mintzberg's assertion conforms to Glaser's and Strauss' (1967) notion that “. . . in generating theory it is not the fact upon which we stand, but the conceptual category that was generated from it.” (Glaser & Strauss, 1967, p. 23) Another way of looking at

categories was to see them as “. . . clusters of interrelated rules and that rules are in turn the product of goal-directed inductive mechanisms.” (Holland, Holyoak, Nisbett & Thagard, 1986, p. 179) “The rules refer to categories, concepts, and schemas.” (Holland et al, 1986, p. 93). Eisenhardt (1989) recommends using a systematic series of analyses to help manage the researcher’s limited information-processing capability in breaking down, interpreting and conceptualizing large amounts of data. The formal classification technique for *axial coding* (specified in the grounded theory method) were useful in this step, and are presented in Table 27.

Classification Technique	Axial Coding
Goal	To systematically develop and relate categories. (Strauss & Corbin, 1998, p. 142)
Description	Interconnections are made among categories and subcategories. Hence the focus is on determining more about each category in terms of: <ul style="list-style-type: none"> • The condition that gave rise to it. • The context in which it is embedded. • The strategies that people use to manage it or carry it out. • The consequences of those strategies. The researcher moves back and forth among data collection, open coding, and axial coding, continually refining the categories and their interconnections. (Leedy & Ormrod, 2001, p. 155)
Variations	<ul style="list-style-type: none"> • Use of mini-frameworks and conceptual diagrams to show the relationships between concepts. (Strauss & Corbin, 1997, p. 141)
Errata	Termed <i>axial</i> because coding occurs around the axis of a category, linking categories at the level of properties and dimensions. (Strauss & Corbin, 1997, p. 123)

Table 27: Characteristics of Axial Coding

The classification of facts were based on a systematic set of relationships. The systematic relationship, in words, is as follows (Strauss & Corbin, 1998):

- **Properties:** Characteristics of a category, the delineation of which defines and gives it meaning.

- Dimensions: The range along which general properties of a category vary, giving specification to a category and variation to the theory.
- Subcategories: Concepts that pertain to a category, giving it further clarification and specification.
- Categories: Concepts that stand for phenomena.
- Concepts: The building blocks of theory.
- Phenomena: Central ideas in the data presented as concepts.

These relationships, presented diagrammatically, are shown in Figure 26.

For this research properties and dimensions referred principally to those of processes and not to those of a person, group or organization; as the properties and dimensions of a process were more relevant to studies aiming at theoretical conceptualization (Glaser, 1978). Because the researcher was moving back and forth between open coding in step 4, and axial coding in step 5, much of this work occurred in parallel, which allowed the researcher to complete work on a single or small group of documents prior to starting another. Lewis and Grimes (1999) state that regardless of how *parallel* the researcher attempts to keep the inductive efforts, insights from previous analyses will exert some influence on later analyses. They go on to recommend an *itinerary* or a planned order of analyses as a method to heighten the awareness of the influences of previous analyses and better enable them to balance contrasting images. Once again, this step was enhanced through the use of a code-based theory building tool discussed in the final section of this chapter.

In summary, the properties and categories discovered in the empirical data were the bricks and mortar of the emerging concepts. For, as they became interrelated, they formed the structure that became the theoretical framework. Once again, Munck's method was used to ensure that the data collected in this step were "... replicable,

reliable, valid, without bias, and within the measurement tolerance and certainty.”

(Munck, 1998, p 22) This helped mitigate criticism surrounding data

analysis/classification in the qualitative element of the research.

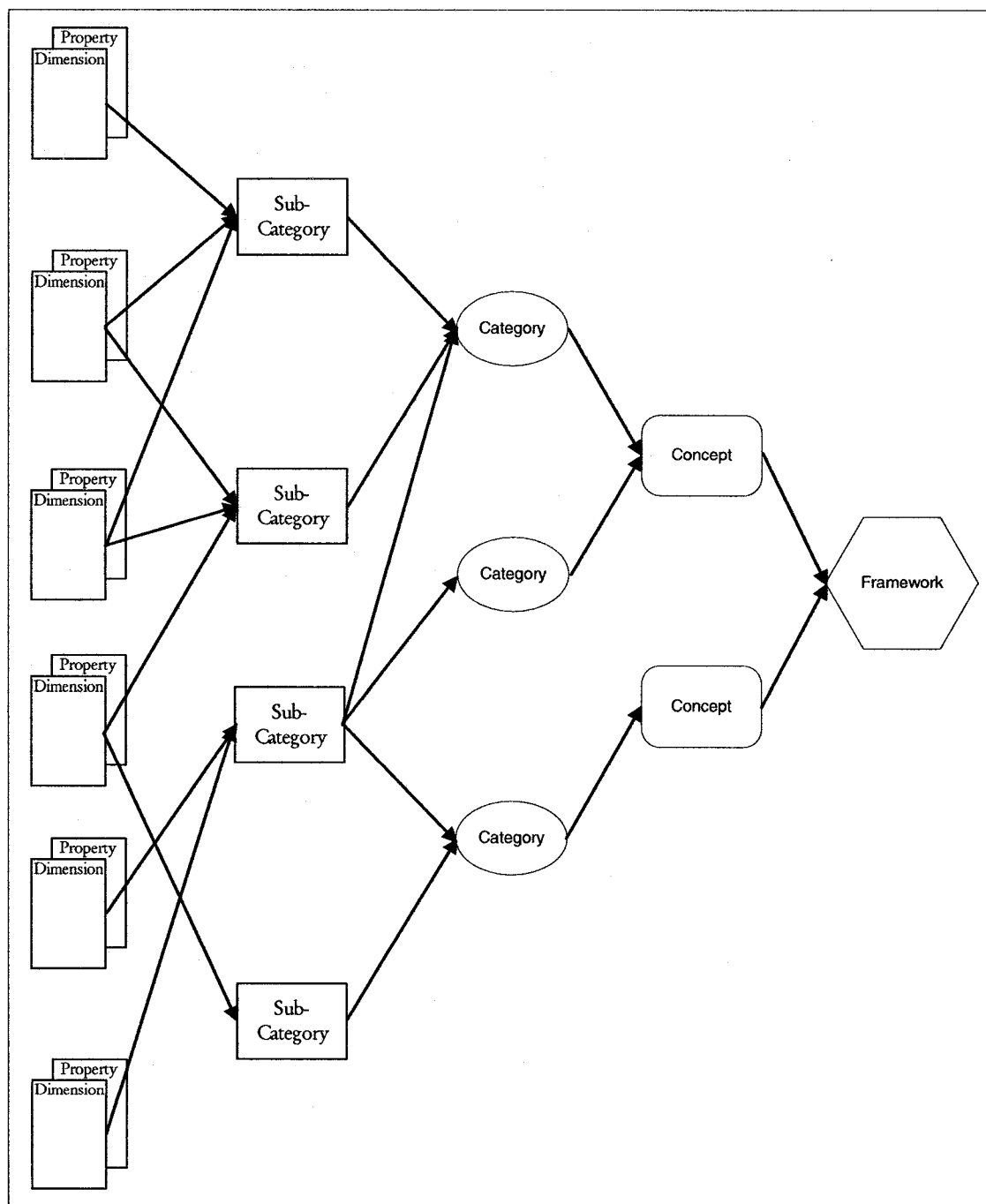


Figure 26: General Classification Schema

Step 6: Construction of the Conception

Whewell proposed two distinct methods for construction of the conception which are a function of whether or not the elementary facts are *facts of quantity* or *facts of resemblance*. “Whewell’s 19th-century methods for the construction of conception, later adopted by Mill,” (Venn, 1907/1973, pp. 403-435) are shown in Table 28.

Facts of Quantity	Facts of Resemblance
The Method of Curves	The Law of Continuity
The Method of Means	The Method of Gradation
The Method of Least Squares	The Method of Natural Classification
The Method of Residues	

Table 28: Whewell’s Methods for the Construction of Conception

Whewell gave no additional guidance as to method. However, the researcher must carefully relate the salient facts, ultimately exploring the relationship between the information groupings previously developed and the outcomes observed. A formal technique called *selective coding* (specified in the grounded theory method), presented in Table 29, was useful in this effort.

Classification Technique	Selective Coding (Strauss & Corbin, 1997, p. 148-161)
Goal	The process of integrating and refining categories.
Description	Categories are reviewed to identify the central category that represents the main theme of the research. A new conceptual idea, in the form of a new category may be created which subsumes the other categories. The criteria include: <ul style="list-style-type: none"> • All other categories can be related to it. • It must appear frequently in the data. • The relation is logical and does not force data. • The concept can explain variation in the data. The central category has analytic power because it can pull the other categories together to form an explanatory whole. Can be represented in an explanatory statement such as: “under these conditions,” “then,” and “when this set of events occurs.”
Facilitating Techniques	<ul style="list-style-type: none"> • <i>Writing a storyline.</i> Descriptive sentences that explain what is going on. • <i>Diagrams.</i> The logic helps present the integrative story. • <i>Validating the Schema.</i> The concept is able to demonstrate (1) a <i>range of variability</i> that accounts for all data and (2) is <i>theoretically saturated</i> because no new dimensions or properties emerge from the data.

Table 29: Characteristics of Selective Coding

There were an infinite number of conceptualizations that could describe the classified facts. This required the researcher to recognize and make explicit what differences in attributes and their magnitudes correlated most strongly with the patterns in the outcomes of interest (Carlile and Christensen, 2005). In an effort to reduce the number of possible conceptions Mullins (1974) constructed a system for cumulating and evaluating these facts. Mullins' analysis reveals, in Table 30, that four basic properties summarize all types of relations that have been proposed for relating concepts within a theory.

Properties	Definition
Association	Two concepts are joined, and this juxtaposition is asserted in a proposition.
Asymmetry	An assertion of the relation in one sentence is not equivalent to asserting that relation in the opposite order.
Quantification	Quantification has two elements: <ul style="list-style-type: none"> • Sign: For concepts which are divided into dichotomies the sign (+ or -) indicates which category of one concept varies with which concept of another. • Effect: This is the size of the effect of one concept on another, in either verbal or numerical form.
Interdependence	The dependence of one relation, for some of its properties or for the value of those properties, on other relations.

Table 30: Properties for Relating Concepts within a Theory

The literature-intensive inductive inference revealed a number of concepts, each with varying degrees of validity and reliability. The researcher determined which had the greatest worth. Mullins includes a procedure by which the researcher may reduce the number of relational statements among the concepts in order to produce a theory which can be logically and empirically evaluated. The three essential steps are (Mullins, 1974):

1. The combination of properties from different statements to give a more comprehensive statement, or build separate models to be verified against data if specific properties contradict each other.
2. Develop an estimate of the effect of each concept on each other.

3. Creation of a matrix which uses the concepts in the set of relations as the rows and columns in the table.

The interpretation of the empirical data in tabular form provided the researcher with an aid in producing a series of verifiable propositions or concepts. Reducing the concepts included in the framework required the researcher to address the dual criteria of *comprehensiveness* and *parsimony* (Whetten, 1989). Comprehensiveness was concerned with including all the relevant factors in the framework while parsimony addressed deleting factors that added little additional value. In the early stage of development a large number of factors were included in the analysis. "Sensitivity to the competing virtues of parsimony and comprehensiveness is the hallmark of a good theorist." (Whetten, 1989, p. 490). Eisenhardt (1989) states that the principal activity of this iterative step is to compare systematically the emergent frame with the evidence in order to assess how well or poorly it fits with the data. She recommends a two step process that includes the procedural elements in Table 31.

Procedural Step	Elements of the Procedure
Sharpening the constructs	<ol style="list-style-type: none"> 1. Refining the definition of the construct. 2. Building evidence which measures the construct. <p>Both of these happen through the constant comparison between data and constructs where the evidence from diverse sources accumulates and converges on a single well-defined construct.</p>
Verifying the emergent relationships	<ol style="list-style-type: none"> 1. The proposed relationship is compared to the evidentiary data. The relationship may be confirmed, revised, disconfirmed, or thrown out for insufficient supporting evidence. 2. When confirmed the construct and the supporting data often provide the foundation for understanding the <i>why</i> of what is happening. <p>These steps are crucial in establishing internal validity.</p>

Table 31: Construct Shaping Procedure

Although this step used the code-based theory building tool discussed in the final section of this chapter, it is important to note that this step in the inductive process was

creative, intellectual work, what Mintzberg (1979) calls “. . . detective work, the tracking down of patterns, consistencies.” (p. 584). Mintzberg goes on to note that “. . . there is no one-to-one correspondence between data and theory. The data do not generate the theory – only researchers do that.” (1979, p. 584). This was where the researcher took the *creative leap*, and developed the proposition or framework. Mintzberg’s *creative leap* has been called many things. Hans Selye [1907-1982] called it “. . . an *intuitive flash*, the *hunch*, which though inspired by the previous steps cannot be deduced from them by the application of formal logic.” (Selye, 1964, p. 267) The product at the end of the 2nd phase was a structured, systemic framework for software development.

Procedure for Phase 3: Verification of the Framework

The goal of the 3rd phase was verification that the structured, systemic framework contained the requisite procedures and features, and measured what it was intended to measure. Whewell concluded his *Discoverers’ Induction* process by verifying the hypothesis. This phase included three procedural steps which culminated in the release of the framework for formal validation using case studies in the quantitative element of the research.

Step 7: Internal Procedural Verification

This step permitted the theoretical framework to be verified, as part of the inductive process. As stated earlier, Whewell’s verification criteria were: “. . . prediction, consilience, and coherence.” (Snyder, 1994, p. 797) The specific characteristics of the verification criteria were as follows.

1. Prediction.

“Quite simply, the use of the model [framework] is to generate predictions or to make truth statements about the model [framework] in operation.” (Dubin, 1978, p. 163)

An operational framework is characterized by its components, units, laws of interaction, boundaries, and systems states and these establish the range over which the framework may explain past or predict future behaviors. Hempel and Oppenheim (1948) opine “. . . that an explanation is not fully adequate unless its *explanans* [the explanatory premises] if taken account of in time, could have served as a basis for predicting the phenomenon under consideration.” (Hempel & Oppenheim, 1948, p. 138) They go on to note that Reichenbach (1944) has established the logical similarity of explanation and prediction, and the fact that one is directed toward past occurrences, the other toward future ones. So, logically, there is no difference between explanation and prediction.

Operationally, the best measure for the framework may be relevance. A *relevant* framework is one that is useful. A useful framework will predict relationships, without causal assumptions. “Prediction seeks to establish an X-Y relationship . . . saying simply that where X appears, Y will also appear.” (Gerring, 2001, p. 125) The measure of goodness for prediction is a function of two criteria: co-variation and priority. Co-variation is the correlation between X and Y. The higher the co-variation between X and Y the better the prediction. Priority refers to the temporal distance separating X and Y. The closer the time interval between X and Y the higher the priority.

2. Consilience.

Consilience, or the unity of knowledge, is a term coined by Whewell (1840) and recently revived by Wilson (1998) as an attempt to bridge the culture gap between the sciences and the humanities. Whewell stated that “. . . the evidence in favour [*sic*] of our induction is of a much higher and more forcible character when it enables us to explain and determine [i.e., predict] cases of a kind different from those which were

contemplated in the formation of our hypothesis. The instances in which this has occurred, indeed, impress us with a conviction that the truth of our hypothesis is certain.”

(Whewell, 1858, pp. 87-88) He noted that “. . . consilience is the means by which we effect the successive generalization that constitutes the advancement of science.”

(Whewell, 1847, II, p. 74) This is the foundation for the concept of generalization used in all of modern science. In this case the consilience of the framework was based on analytic generalization. Type ET generalizability, described in Figure 19, was considered to be well developed and addressed the generalizability of empirical descriptions to theory. The framework was judged on its ability to logically apply the empirical descriptions in the systems-based literature to a framework that addressed software development project performance.

3. Coherence.

Whewell’s third test of a theory’s truth was coherence. Whewell claimed “. . . the system becomes more coherent as it is further extended. The elements which we require for explaining a new class of facts are already contained in our system . . . In false theories, the contrary is the case.” (Whewell, 1858, p. 91) In this case, coherence occurs when the framework is able to be applied to a new class of phenomena without modification of the framework. Whewell saw coherence as a special type of consilience that happens over time; remarking that “. . . consilience and coherence are, in fact, hardly different.” (Whewell, 1858, p. 95) Because this researcher did not have the luxury of evaluating the framework over any meaningful period of time simplicity was utilized as a measure for coherence.

Kaplan (1964) writes that “. . . coherence is a conservative principle, which ruthlessly suppresses as rebellion any movement of thought which might make for a scientific revolution. The unyielding insistence that every new theory must fit those theories already established is characteristic of closed systems of thought, not of science. . . . Nevertheless, the point remains that theories can not be validated as though they were wholly self-contained.” (Kaplan, 1964, p. 314) The most widely applied modern norms of coherence are internal to the theory [framework] itself; simplicity and esthetics.

While a great deal of discourse has occurred on the distinction between descriptive and inductive simplicity, this was avoided in favor of a more general explanation. “A framework’s simplicity may be evaluated as a function of the manageability of both the equations and the text. A simple framework can be recognized as such, even if the evaluator can not say precisely why. Esthetic considerations closely follow simplicity. While the notion that a framework can be *beautiful* in the same sense as art can be argued, there is no doubt that simplicity and symmetry of design have significant roles in the evaluation of a framework.” (Kaplan, 1964, pp. 314-319)

In summary, the completed framework was verified against the criteria contained in Table 32.

Criteria	Measure	Characteristics
Prediction	Co-variation	The higher the co-variation between the framework (X) and the prediction (Y) the better the prediction.
	Priority	The closer the time interval between the framework (X) and the prediction (Y) the higher the priority.
Consilience	Analytic Generalization	The use of Type ET generalization to describe empirical findings (data in the literature) to theory (the inductively developed systems-based framework)
Coherence	Simplicity	Manageability of the equations and/or text in the framework.
	Esthetics	Symmetry of design.

Table 32: Framework Verification Criteria

If the framework had unresolved issues in these areas it could be re-introduced to the construction of the conception in Step 6 via the verification feedback loop in Figure 27. This ensured that the relationships between the information groupings

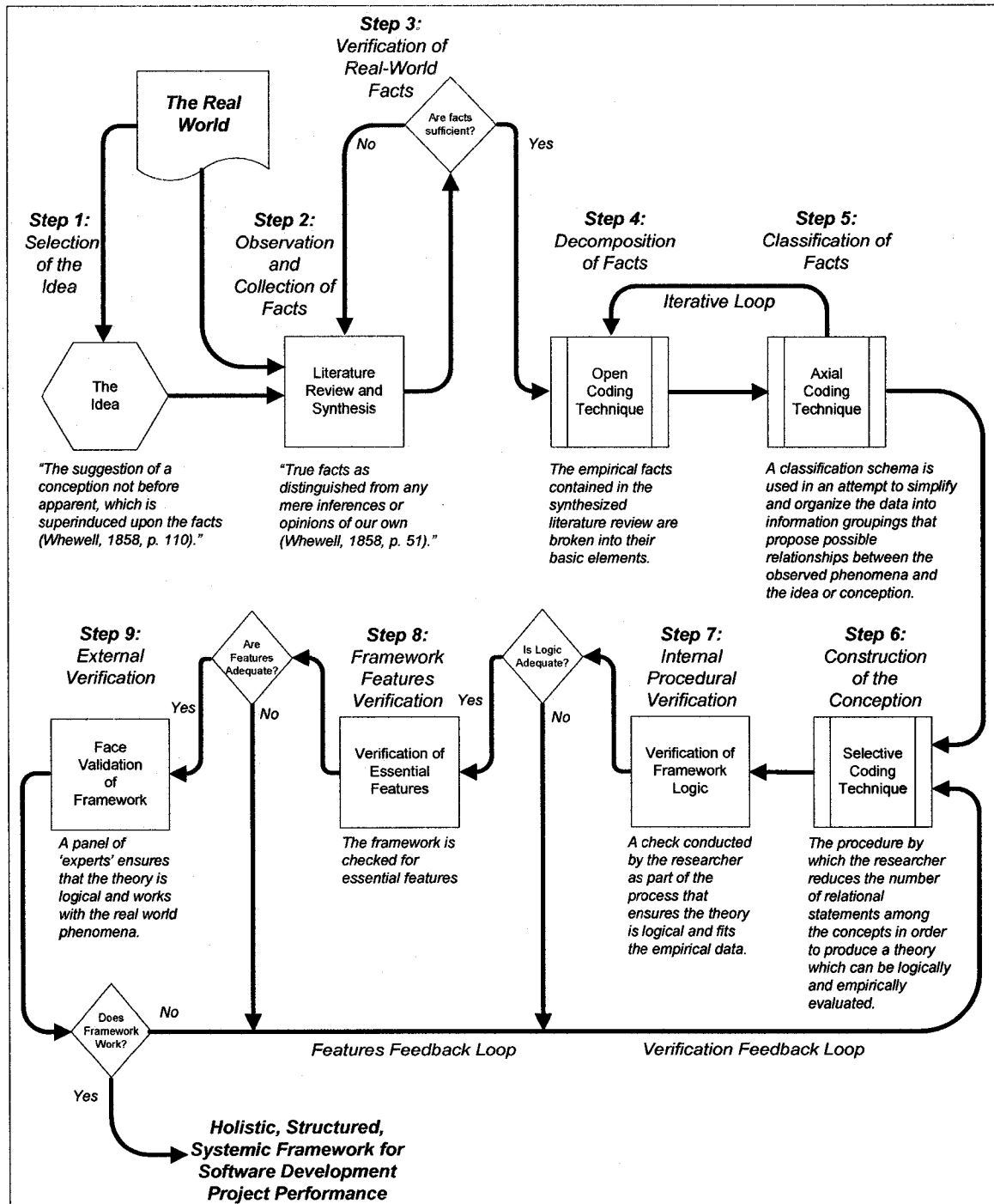


Figure 27: Steps in the Augmented Discoverers' Induction Procedure

(i.e. properties and categories) and the conception could be re-evaluated. Because “. . . inducing theory from qualitative data is adaptive and highly iterative;” (Carroll & Swatman, 2000, p. 236) this step had tremendous significance in the inductive process. Eisenhardt (1989) writes that “. . . while an investigator may focus on one part of the process at a time, the process itself involves constant iteration backward and forward between steps.” (1989, p. 546). The iterative and verification feedback loops in Figure 27 provided *learning* to the processes which permitted the framework to emerge through the introduction and verification of various categories related to the conception.

Step 8: Internal Features Verification

Once the framework successfully passed the internal procedural verification it was checked for essential features. Because the framework was a conceptual or theoretical model designed to be used in the real-world it included the key features of modern theories and elemental constructs derived from systems science. The features feedback loop in Figure 27 was included to re-cycle the framework if it was missing these essential features. Six key features were included in the framework features verification:

1. Units of the Framework

Units refer to the things from which the theoretical framework was developed. Kaplan (1964) is very clear on the meaning of things.

By and large, then, the important terms of any science are significant because of their semantics, not their syntax; they are not notational, but reach out to the world which gives the science its subject-matter. The meaning of such terms results from a process of conceptualization of the subject-matter. In this process the things studied are ‘classified’ and ‘analyzed:’ several things are grouped together and particular things are assigned to the several groups to which they belong The concept of ‘paranoid,’ for example, puts into a single class a certain set of persons,

and is itself analyzed into such patterns as delusions of persecution, auditory hallucinations, impairment of ego-functions, or the like. Each of these patterns in turn is a classification, grouping together a set of actions, verbal or otherwise as the case may be, and without regard to the actors performing them. (p. 50)

It is the properties of Kaplan's *things*, or *chunks of information* grouped from the empirical data, that were important. The selected characteristics of the *chunks of information* are what the theoretical framework was about. The *chunks of information* are the operational units of the theory. Because the units of theory may be either attributes (properties or dimensions) or variables an important distinction between the two must be made. The two may be differentiated as follows (Dubin, 1978, p. 42).

- *Attribute: distinguished by the quality of being present.*
- *Variable: the property of a thing that may be present in degree.*

The hierarchies of concepts, categories, and subcategories had, as their primal elements, the *chunks of information*. Therefore, by extension, the subcategory, category, or concept are units of theory. This was important because theory is concerned with modeling the processes and outcomes of particular units interacting in systems.

2. Rules for Interactions among the Units of the Framework

The linkages between units of a model are labeled as *laws of interactions*. "A lawful statement expresses a linkage or connection between two or more units." (Dubin, 1978, p. 90) The structure of a scientific law is such that it is composed of two analytically distinct parts – units that are connected or linked by a law of interaction. It is the connecting phrase in a sentence that is the *law of interaction* which is linking the subject (a unit) with the object (a unit). Once the basic definition was established it became important to understand where causality fit. Dubin (1975) warns that a law, as a

statement of relationship, does not necessarily also act as a statement of causality. He goes on to state that the basis for this contention is “. . . that modern systems theory, when applied to the analysis of social systems does very well without the blessing or aid of causal notions.” (Dubin, 1975, p. 107) This is supported by a statement from von Bertalanffy (1968)

We may state as characteristic of modern science that this scheme of isolable units acting in one-way causality has proved to be insufficient. Hence the appearance, in all fields of science, of notions like wholeness, holistic, organismic, gestalt, etc., which all signify that, in the last resort, we must think in terms of systems of elements in mutual interaction. (p. 45)

Gibbs (1972) also warns that “. . . causal language in theory construction introduces seemingly insoluble problems and irresolvable issues.” (Gibbs, 1972, p. 815) In summary, the interaction was only a statement of relationship, not causality. That stated, there were three general and one special category within which interactions were expressed. Table 33 is a list of the interaction categories (Dubin, 1978).

Interaction Category	Definition
Categoric	Values of a unit are associated with values of another unit.
Sequential	Always employs a time dimension.
Determinant	Associates determinate values of one unit with determinant values of another unit. An example is Boyle's law which states that under constant temperature the volume of a gas is inversely proportional to the pressure bearing upon the gas.
Negative	Specifies generalized non-relationship among units. For instance, there is systematic relationship between the values of unit A and those of unit B. This negative law of interaction is often called the null hypothesis.

Table 33: Interaction Categories

There was one final dimension used to describe the law of interaction. This is the efficiency of a law, which described the range of variability in the values of one unit when they are related by a law to the values of another unit. The four general levels of

efficiency of a law of interaction, in increasing order, are listed in Table 34. The laws of interactions added order to the framework by explicitly delineating patterns. “The more complex the set of relationships under consideration, the more useful it is to graphically depict them.” (Whetten, 1989, p. 491)

Efficiency of Interaction	Definition
Presence-absence	This is the lowest level of efficiency and is one that states that given a positive or negative value of unit A, there will a corresponding positive or negative value of unit B.
Directionality	This proclaims the directionality of a relation between the values of unit A and the values of unit B.
Co-variation	This expresses co-variation between two or more units.
Rate-of-change	This expresses the rate of change in the value of unit A and the associated rate of change in the values of unit B.

Table 34: Levels of Efficiency of a Law of Interaction

3. Boundaries of the Framework

The boundaries of the framework explicitly stated where the framework was expected to be effective. This is called the domain and is defined as “. . . the territory over which we can make truth statements about the framework, and therefore, about the values of the units composing the framework.” (Dubin, 1978, pp. 134-135) “There is an inverse relationship between the number of boundary-determining criteria employed in a model and the size of the domain owned by the model.” (Dubin, 1978, p. 134) This must be related to the section on Theory Development and Theorizing and the stated intention to work with *theories of the middle range* (Merton, 1968). “The sense or meaning that can be given to the term *theories of the middle range* is that they are models having not too few and not too many boundary-determining criteria.” (Dubin, 1978, p. 135) At the start of the research the boundary conditions for the framework were stated in broad terms and addressed the following:

- a theoretical strategy upon which the framework is based
- a formal method for constructing the framework
- a position on the theoretical continuum
- the real-world domain within which it will be applied

4. Utility of the Framework

This feature was concerned with the utility of the framework. Utility addressed the question *what makes this framework useful?* This is what Maxwell (1962) calls the conditions of adequacy of the framework. In the most basic sense the framework's role is to report, explain and predict the facts concerning the phenomena under consideration. Maxwell provides an elegant statement when he states: "A framework allows us to express a greater number and a larger variety of *observational facts* – *and this is crucial* – *to explain these facts.*" (Maxwell, 1962, p. 136) Bacharach (1989) tabulates the characteristics of *utility* in Table 35.

Characteristics	Utility
Variables	<i>Variable scope:</i> Variables must sufficiently although parsimoniously tap the domain of the constructs in question.
Constructs	<i>Construct scope:</i> Constructs must sufficiently although parsimoniously tap the domain of the phenomenon in question.
Relationships	<i>Explanatory potential:</i> Establishes the substantive meaning of constructs, variables, and their linkages. <i>Predictive adequacy:</i> Validates substantive meaning by comparing it to empirical evidence.

Table 35: Characteristics used in Evaluating Utility

5. Pragmatic Factors and the Framework

The final feature was included to ensure that the framework was *useful*. Usefulness answers the question *why is this framework more useful than another?* The question addressed the concerns that arise when more than one framework exists. Bacharach (1989), Maxwell (1962), and Whetten (1989) answer the question by citing a number of pragmatic factors that affect frameworks in Table 36.

Pragmatic Factors	Elements
Logic underlying the Framework	<p>The framework must convince others that the propositions make sense.</p> <ul style="list-style-type: none"> • Useful guide for research. • Provides a framework for interpreting patterns, or discrepancies in empirical observations. • Explains <i>how</i>, <i>when</i>, and <i>why</i> certain relationships exist in the empirical data.
Simplicity	Ease of comprehension, communication, computations and other inferential manipulations
Aesthetic Considerations	Idiosyncratic tastes and a language relevant for users of the framework

Table 36: Pragmatic Factors Affecting Frameworks

Step 9: External Verification

This was designed as a formal check of the completed framework prior to validation through the use of case studies. Ahire & Devaraj (2001) conducted an empirical comparison of construct validation approaches and recommend two validity checks at the completion of the development phase for measurement instruments (frameworks). Table 37 provides the details for post-development validity checks for measurement instruments. The external verification was an important step and was accomplished using a panel of experts who conducted the post-development verification using both content and face validation criteria to judge the framework. By occurring prior to the formal validation of the framework it allowed the researcher to incorporate the comments of outsiders prior to the case study validation. The use of outside experts increased the validity of the inductive process, the stability of the framework, and the external validity and transferability of the research.

Validity Check	Definition
Content Validity	<ul style="list-style-type: none"> • “The degree to which an empirical measurement reflects a specific domain of content (Carmines & Zeller, 1979, p. 20)” • “The <i>representativeness</i> or <i>sampling adequacy</i> of the content – the substance, the matter, the topic – of a measuring instrument (Kerlinger & Lee, 2000, p. 667)” “Content validation, then, is basically judgmental. The items of a test must be studied, each item being weighed for its presumed representativeness of the universe. This means that each item must be judged for its presumed relevance to the property being measured, which is no easy task. Usually other <i>competent</i> judges should judge the content of the items. The universe of the content must, if possible, be clearly defined; that is, the judges must be furnished with specific directions for making judgments, as well as with the specification of what they are judging (Kerlinger & Lee, 2000, p. 668).”
Face Validity	<ul style="list-style-type: none"> • “Concerns the extent to which an instrument <i>looks like</i> it measures what it is intended to measure (Nunnally, 1967, p. 99).” “Face validity concerns judgments about an instrument after it is constructed. . . . Face validity can be considered one aspect of content validity, which concerns an inspection of the final product to make sure nothing went wrong in transforming plans into a completed instrument (Nunnally, 1967, p. 99).” • “Face validity is not validity in the technical sense. It refers to what the test appears to measure. Trained or untrained individuals would look at the test and decide whether or not the test measures what it was supposed to measure. There is no quantification of the judgment or any index of agreement that is computed between judges (Kerlinger & Lee, 2000, p. 668).”

Table 37: Post-Development Validity Checks for Measurement Instruments

As was the case with the expert reviewer in Step 3 an *expert* is defined as “a person who has background in the subject area and is recognized by his or her peers or those conducting the study as qualified to answer questions. The three factors that must be addressed when gathering expert judgment are (Meyer & Booker, 2001, p. 7).

1. *Selecting experts according to particular criteria,*
2. *Designing elicitation methods, and*
3. *Specifying the mode in which the expert is to respond*

The selection of the panel of experts was governed by their professional qualifications.

The criteria for the expert panelists are listed in Table 38.

Qualification	Qualification Criteria
Education	Ph.D. candidate or earned Ph.D. in engineering management, systems engineering, software engineering, or related discipline.
Experience	Greater than 10 years experience with both commercial and government systems and software development methodologies.

Table 38: Qualifications for Expert Panelists

By meeting the qualifications in this profile the panelist's added additional validity to the research study.

The elicitation method for the panel of experts was a modified *Delphi* situation in which each panel expert, isolated from one another and the researcher provided judgments based on the verification criteria for the framework. The judgments were made against the completed framework from Step 8. The guidelines for the panel and the mode of response were specified in Appendix C. The completed verification forms from the panel of experts became research data with recommendations serving as sources of change for the framework. The researcher anticipated that the panelist's would recommend modifications to the framework that added clarity to the study. The additional understanding gathered from the recommendations of the panelists ensured the plausibility of the framework. Because the panel of experts was of limited size, formal statistical measures (Cohen, 1960; Lawshe, 1975) required to correlate the judgments of the panel were not required.

The product at the end of the 3rd phase was formal verification that the structured, systemic framework accurately represented the real-world phenomena.

Summary of the Qualitative Procedure

At this point the reader may think that the researcher has high-jacked the grounded theory method and disguised it as Whewell's Method of *Discoverers*'

Induction. While it is true that the open, axial, and selective coding techniques of the grounded theory method were useful for the decomposition and classification of facts and to aid in the construction of the conception, there are two significant differences between Grounded Theory and *Discoverers' Induction*.

1. The most important difference was centered upon the *idea*. In *Discoverers' Induction* the researcher brought the *idea* to bear upon the facts. It is important to note that the source of the conception from the mind of the researcher was based upon academic training and real-world experience. Both of these sources served to create ideas in the mind that correspond closely enough with reality that true theories about the real-world were developed using these ideas as their conceptions. In grounded theory "... a researcher does not begin a project with a preconceived theory in mind. Rather, the researcher begins with an area of study and allows the theory to emerge from the data." (Strauss & Corbin, 1998, p. 12) "A grounded theory study is least likely to begin from a particular theoretical framework." (Leedy & Ormrod, 2001, p. 154)

2. Another important difference is that in *Discoverers' Induction* the extant literature was the source of the facts for decomposition and classification and served as the source for the colligation of facts. In grounded theory the data from the extant literature is considered *after* the emergence of substantive theory.

Figure 28 is an alternative view of the generalized inductive process described in the last section and serves as an excellent summary of the first nine steps in the procedure. The theory building triangle is an adaptation of emerging work by Carlile and Christensen (2005) that accurately describes the steps used in Whewell's *Discoverers' Induction* using what Carlile and Christensen call descriptive theory.

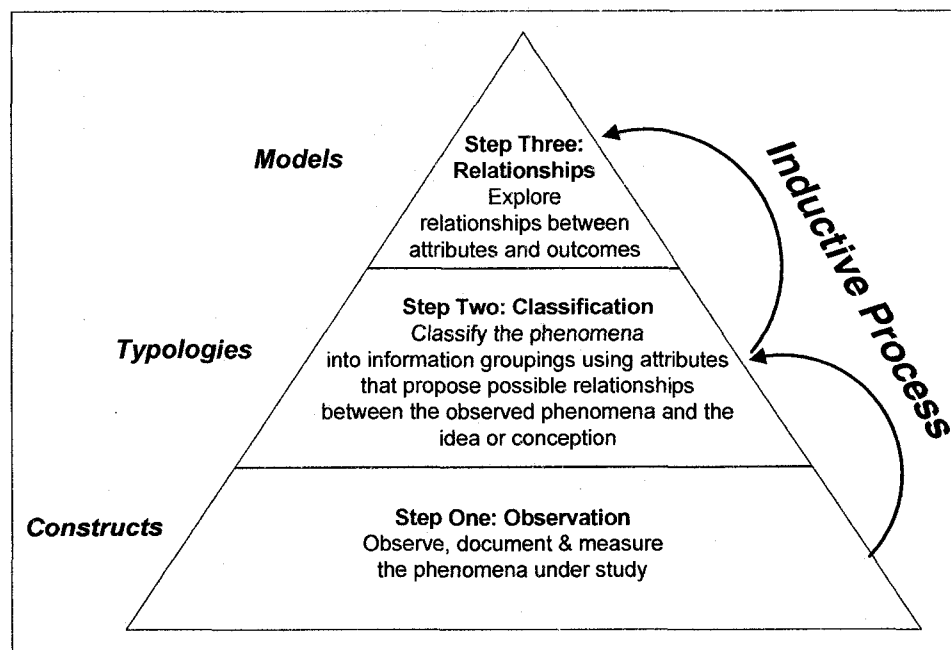


Figure 28: The Theory Building Triangle

Adapted from Carlile, P.R. & Christensen, C.M. (2005). "The Cycles of Theory Building in Management Research," Unpublished Manuscript, p. 5.

The process in Figure 28 may be repeated as a series of inferences from observations and contextual information conforming to Whewell's *train of researches*. It is interesting to note that the arrows stepping upward in the depiction of the inductive process in Figure 28 conforms to Bacon's expression *ascending* for the process of inductions and to the more common metaphor of "... *rising to first principles*." (Venn, 1907/1973, p. 364) The theory building triangle will be called upon again to describe the transition from theory building to framework validation in what Carlile and Christensen describe as the transition from descriptive to normative theory.

In summary, the use of formal techniques from the Grounded Theory Method to support the *Discoverers' Induction Method* was well within the purview of the researcher and followed the pragmatic practice of combining techniques to obtain desired results recommended by Creswell (1998). It is important to note the temporal relationships depicted in Figure 27 and listed in Table 24, included three important feedback loops that

ensured that the process *learned* during development of the framework. These important controls ensured that the emerging theory complied with the logic and features of good theory.

Introduction to the Quantitative and Reporting Procedures

The final seven steps validated the framework and reported the findings of the research study. The validation was conducted by using two real-world case studies and the framework. This element of the research was centered on analysis of the empirical data from the case studies and comparison with the inductively derived framework developed in the first element. Table 39 is the overall structure for the quantitative and reporting elements of the research.

Structure	Definition of the Element
Quantitative Element	Validation of the inductively developed holistic, structured, systemic framework for software development projects using case studies.
Phase 4	Validation Case Studies: Selection and characterization of the case studies used to validate the framework.
Step 10	Selection of the Case Studies: The researcher must compile, review, and select two case studies from of number of candidate software development projects.
Step 11	Characterization of the Case Studies: A multi-dimensional project typology is used to characterize the software development projects. which will contribute significantly to the generalizability of the research and provide clear avenues for future validation of the framework
<i>Milestone 4</i>	<i>Product: The selected and characterized case studies.</i>
Phase 5	Data Collection and Analysis: Analysis of the empirical data from the case study
Step 12	Developing the Analytic Strategy. Development of an analytic strategy that is broad enough to address the conduct of analysis at the level of the whole case
Step 13	Collecting the Evidence: The researcher must collect evidence from each of the software development projects or cases.
Step 14	Analyzing the Evidence: In this step the researcher must analyze the evidence collected in the previous step.
Step 15	Interpreting the Evidence: In this step the researcher must interpret the evidence collected in the previous step.
<i>Milestone 5</i>	<i>Product: Interpretation of the case studies and implications for the framework.</i>
Reporting Element	In this element the findings are reported in the dissertation and the data is preserved for use in an article in a scholarly journal.
Phase 6	Publication: Publication of the research findings.
Step 16	Reporting the Case Study: The researcher must bring the results and findings to closure in a report.
<i>Milestone 6</i>	<i>Product: Completed case studies and dissertation results.</i>

Table 39: Structure for the Quantitative and Reporting Elements

The following sections will discuss the phases and detailed steps taken during the quantitative and reporting elements of the research.

Procedure for Phase 4: Validation Case Studies

The goal of the 4th phase of the research was the selection and structure of the data required to validate the framework developed in Step 9. This phase was supported by two independent steps that selected and characterized the case studies.

Step 10: Selection of the Study Cases

In the 10th step the researcher compiled and reviewed of number of candidate software development projects for inclusion in the case study. While “. . . multiple-case designs allow for cross-case analysis and the extension of theory,” (Benbasat, Goldstein & Mead, 1987, p. 373) the time span for completion of the dissertation was an important consideration, therefore only two cases were selected. In addition to this constraint, the criteria utilized for the selection of each of the case studies was an important element of the research as the individual criteria had a direct impact on the ability to make generalizations based on the findings of the study. In order to fully describe the large field of diverse software development projects the “. . . selection criteria included categories that were mutually-exclusive, exhaustive, and comparable.” (Gerring, 2001, p. 120) Table 40 lists the general criteria and the guidelines for each.

General Criteria	Guideline
Mutual-exclusivity	Are the categories mutually-exclusive, or do they overlap? Can relevant phenomena be sorted into one or another category without difficulty?
Exhaustiveness	Do the categories account for all the phenomena of a given type?
Comparability	Are the dimensions of the classification comparable? Are they logically-compatible parts of a larger whole?

Table 40: General Selection Criteria for Case Studies

The use of the criteria in Table 40 permitted the researcher to make logical selections of two cases with a reasonable degree of certainty that they would not overlap or describe the same domain. This directly supported the method of generalization used in case studies; analytic generalization. Analytic generalization involved generalizing to a theory or in this case a framework—not to a population—and was based on validating framework-driven behaviors with evidence collected in a variety of settings in the case studies. In order to ensure the variety of the case studies two project criteria were selected: project type and project duration.

- **Project Type:** The software development project is delivering software to either a commercial firm, local, state or federal government or a consortium of commercial and government entities.
- **Project Duration:** The software development project took less than one year or greater than one year and less than 2 years, and so forth; from start to finish, to complete delivery of the software.

Table 41 shows how each of the selected project criteria satisfied the general selection criteria guidelines, thereby ensuring that the case study selections did not represent similar domains.

General Criteria	How it Satisfies the Guideline
Mutual-exclusivity	Are the categories mutually-exclusive, or do they overlap? Can relevant phenomena be sorted into one or another category without difficulty? <u>Yes:</u> (a) A software development project may only be a commercial or government project or a consortium but not both and (b) a software development project may only have one duration.
Exhaustiveness	Do the categories account for all the phenomena of a given type? <u>Yes,</u> both criteria are totally inclusive.
Comparability	Are the dimensions of the classification comparable? Are they logically-compatible parts of a larger whole? <u>Yes.</u> Each criterion belongs to the superset of: (a) all software development projects and (b) all software development projects that have been completed.

Table 41: Software Development Project Selection Criteria

Plotting the software development projects on the grid in Figure 29 clearly portrayed the limitations of the study and served as a means for future researchers to use the framework and extend its applicability beyond the research conducted in this study.

Project Type	Federal Government					
	State Government					
	Local Government					
	Government-Commercial Consortium					
	Commercial					
		< 1 Year	1-2 Years	2-3 Years	3-4 Years	> 4 Years
		Project Duration				

Figure 29: Software Development Project Grid

Step 11: Characterization of the Study Cases

After selection, a multi-dimensional project typology was used to characterize the software development projects. The NCTPO *Pentagon Model* was selected to serve as a guide for the software development projects included as Case Studies. The rationale for selection of this model was as follows.

The first action in the selection process was to conduct a review of the software project management literature in search of a standard typology. A survey of the major software project management texts (Bennatan, 2000; Futrell, Shafer & Shafer 2000; Gilb, 1988; Royce, 1998; Schwalbe, 2002; Thayer, 1997) and the latest version of the

SWEBOK (Abran & Moore, 2004) made no mention of project type, typology or taxonomy. However, a large number of software project characteristics were considered when developing estimates for the creation of software. Perhaps the most comprehensive empirical analysis of the characteristics that affect software development effort was conducted by Barry Boehm. Boehm developed 22 characteristics to be used as factors when developing detailed estimates of effort for software development projects (Boehm, 1981; Boehm, Abts, Brown, Chulani, Clark, Horowitz, Madachy, Reifer & Steece, 2000). All of the factors and effort multipliers were directly related to the characteristics of the project and/or the product to be developed.

The second action in the selection process was to search the general project management literature in search of a standard project typology. The search uncovered early work by Shenhar and Dvir (1996) and a project typology called the *UCP Model*, which used uncertainty, complexity, and pace as project measures. These measures proved to be a dominant construct for understanding technical projects. Shenhar and Dvir built upon their earlier research and proposed a more refined four-dimensional model called the NCTP Diamond Model (Shenhar, Dvir, Milosevic, Mulenburg, Patanakul, Reilly, Ryan, Sage, Sauser, Srivannaboon, Stefanovic & Thamhain, 2005). Analysis of this model found that the NCTP *Diamond Model* addressed 17 of the 22 software project characteristics identified by Boehm in the 4 Dimensions of the NCTP *Diamond Model*. The 5 characteristics that are not accounted for are shown in Table 42.

The third action in the selection process was to account for the characteristics missing from the NCTP *Diamond Model* in Table 42. The missing characteristics have to do with staff skills (ACAP, PCAP) or organizational competencies (PCON, TEAM and

PMAT). In order to account for these a fifth dimension for Organizational Maturity was added to the four-dimensional NCTP *Diamond Model*. The 5th dimension was conglomerated by using the PMAT, which is a representation of the team's Capability Maturity Model® (CMM) rating.

Boehm's Project Estimating Factors	Factor Definition
Analyst Capability (ACAP)	Rates the analysis and design ability, efficiency and thoroughness, and the ability to communicate and cooperate of the analyst team.
Programmer Capability (PCAP)	Evaluates the capability of the programmers as a team.
Personnel Continuity (PCON)	Rates the Project's annual personnel turnover
Team Cohesion (TEAM)	Accounts for the sources of a project's turbulence
Process Maturity (PMAT)	The project's CMM level at the start of the project.

Table 42: Characteristics Missing from NCTP Diamond Model

The final action was the description of the new multi-dimensional project typology; the NCTPO *Pentagon Model* – that included measures for Novelty, Complexity, Technology, Pace, and Organizational Maturity. The five project measures are plotted on the axes of a relational graph known as a Kiviat graph. (Kolence, 1973; Kolence & Kiviat, 1973) Each of the characteristics has five measures which determined the placement on the axes of the NCTPO *Pentagon Model*. Table 43 lists the characteristics for Novelty, Complexity, Technology, Pace, and Organizational Maturity and their measures. The transparency of the selection criteria for the software development projects used as case studies was ensured by using the NCTPO *Pentagon Model* which contributed significantly to the generalizability of the research and provided clear avenues for future validation of the framework.

Software Development Project Characteristics	Characteristic Measure
Novelty where software is described as:	<ol style="list-style-type: none"> 1. Maintenance: Work on existing software 2. Improvement: Revision to existing software. 3. Upgrade: A new generation of an existing software product. 4. Replacement: A replacement for existing software. 5. Breakthrough: New-to-the-world software.
Complexity where the software is being developed as:	<ol style="list-style-type: none"> 1. Program: Program performing a single function. 2. Subsystem: Module performing multiple functions in a single functional area. 3. System: Collection of subsystems with multiple functions. 4. Array: Widely dispersed collection of subsystems with a common mission. 5. Super-System: A collection of independent systems.
Technology being used is:	<ol style="list-style-type: none"> 1. Low-Tech: No new technology is used. 2. Medium-Tech: Some new technology. 3. High-Tech: All or mostly new, but existing technologies. 4. Emerging Tech: Technology is in development but not yet released. 5. Super High-Tech: Necessary technologies do not exist at project initiation.
Pace of software delivery is:	<ol style="list-style-type: none"> 1. None: Not critical. 2. Routine: Based on well-developed release schedule that must be met. 3. Fast-competitive: Time to market is important for the business. 4. Time-critical: Completion time is crucial for success-window of opportunity. 5. Blitz: Crisis project - immediate solution is necessary.
Organizational Maturity or CMM® or CMMI® rating at start of project was/is:	<ol style="list-style-type: none"> 1. Ad hoc 2. Repeatable 3. Defined 4. Managed 5. Optimizing

Table 43: NCTPO Pentagon Model Characteristics and Measures

A sample representation of the NCTPO *Pentagon Model* is shown in Figure 30. The products at the end of the 4th phase were the selected and characterized cases.

Procedure for Phase 5: Case Study Validation

In this phase the case study data were collected and analyzed and a judgment with respect to the applicability of the framework was made.

Step 12: Developing the Analytic Strategy

Formal analysis required an analytic strategy, in this case, one that was broad enough to address the conduct of analysis at the level of the whole case. The case-based analytic strategy relied on the 2nd question in the research study, forcing it to guide and shape the data collection plan. The analytic strategy had three sections: (1) formulation,

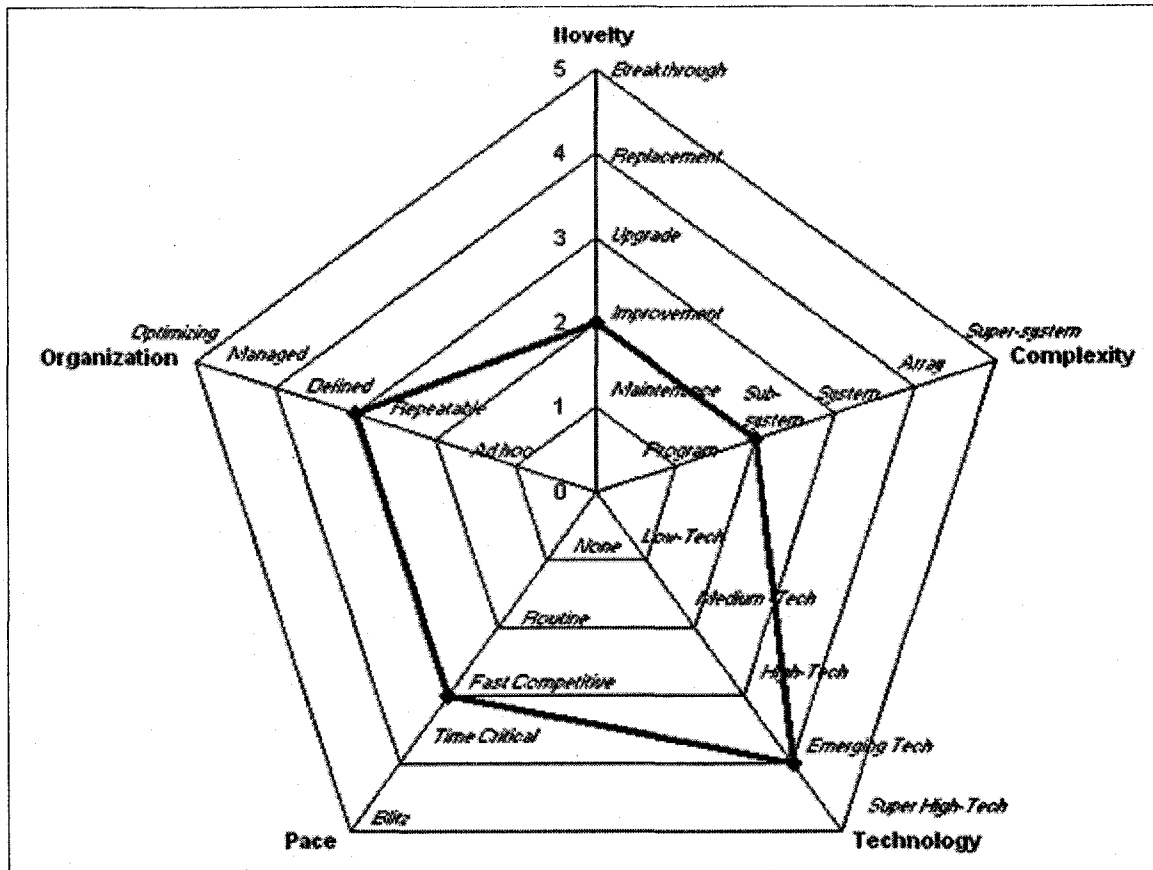


Figure 30: NCTPO *Pentagon Model*

(2) quantitative and qualitative analysis, and (3) interpretation. The uniqueness of each analytic strategy is heavily influenced by the following factors:

- Problem definition: the contextual domain of each case was the starting point and was the central focus of the research questions and propositions.
- Case Study boundaries: defining the bounds of the specific case to be studied allowed the researcher to frame the case for analysis from a number of viewpoints. Multiple views allowed the researcher to gain significant insight into the case.

- Stakeholders: The stakeholders included entities or people that had a perceived interest in the case under study. The inclusion of stakeholders, when possible, ensured that the analysis included the views of all involved with the case.
- Data Collection: The availability of data and the data reduction strategy were significant factors influencing the analytic strategy.
- Analysis Forms and Techniques: Access to data analysis software, techniques, and expertise all affected the analytic strategy.
- Researcher: The background and qualifications of the researcher were of significant import. The researcher developed a number of new skills, techniques, and functional knowledge.
- Resource Constraints: During the formulation of the analytic strategy the researcher set the requirements for resources to include the manpower, material, money, methods, time, and information required for the research.

The analytic strategy developed in this step served as the frame for the next three steps of the case study method and acted as the protocol for the study. “The protocol is a major way of increasing the *reliability* of case study research and is intended to guide the investigator in carrying out the data collection.” (Yin, 2003, p. 67)

Step 13: Collecting the Evidence

In the 13th step the researcher collected evidence from each of the software development projects or cases. This was addressed by the 1st section of the analytic strategy, formulation, where the quantitative and qualitative approach used in the analysis was developed. Formulation involved two actions: (1) setting boundaries to define the aspects of the cases that were studied within the limits of time and means, and (2) the

creation of a frame to help uncover, confirm, and qualify the basic processes and constructs that served as the foundation for the study. This step was subjective because determinations about what data to collect and the data reduction schema evolved during the data collection process. However, by using one of Miles & Huberman's theoretically-driven sampling strategies (1994, pp. 27-30), claims for *generalizability* were able to approximate the quantitative ideal. "Evidence from the case studies may come from six sources: documents, archival records, interviews, direct observation, participant-observation, and physical artifacts." (Yin, 2003, p. 83) Table 44 was used to guide the data collection and reduction schema.

Types of Evidence	Data Collection Methods and References	Example Sources of Evidence
Documents	Miles & Huberman (1994) Leedy & Ormrod (2001) Creswell (2003)	<ul style="list-style-type: none"> • Letters, memoranda, and other communiqués • Agendas, announcements and minutes of meetings, and other written reports of events • Administrative documents: proposals, progress reports, and other internal records • Formal studies or evaluations of the same "site" under study • Newspaper clippings and other articles appearing in the mass media or in community newsletters
Archival Records	Miles & Huberman (1994) Leedy & Ormrod (2001) Creswell (2003)	<ul style="list-style-type: none"> • Service records, such as those showing the number of clients served over a given period of time • Organizational records, such as organizational charts and budgets over a period of time • Map and charts of the geographical characteristics or layouts of a place • Lists of names and other relevant items • Survey data, such as census records or data previously collected about a site • Personal records, such as diaries, calendars, and telephone listings
Interviews	Merton, Fiske, & Kendall (1990) Patton (1987) Rubin & Rubin (1995) Berdie, Anderson & Niebuhr (1986)	<ul style="list-style-type: none"> • Open-ended interview • Focused interview • Semi-structured interview • Questionnaires

Table 44: Evidence and Data Collection Methods for Case Studies

Because completed software development projects were selected as case studies, neither direct observation, participant-observation, nor physical artifacts were included as potential data collection methods.

Using Multiple Sources of Evidence

Using multiple sources of evidence, as described in Table 44, permitted the researcher to address a broader range of issues than those found in any one data collection method. However, the most important advantage was that multiple sources of evidence formed *converging lines of inquiry* in a process called triangulation. “The triangulation metaphor is from navigation and military strategy that use multiple reference points to locate an object’s exact position.” (Smith, 1975, p. 273) Denzin (1971) states that “. . . triangulation forces the observer to combine multiple data sources, research methods, and theoretical schemes in the inspection and analysis of behavioral specimens.” (Denzin, 1971, p. 177) White (2000) and Denzin (1971) propose three methods of triangulation:

1. Data Triangulation. Achieved by collecting data from different sources over different time-scales. In this case multiple case studies which occurred over different periods of time were used.
2. Method Triangulation. This was applied through the use of multiple methods in what Zelditch (1962) calls between-method triangulation. In this case evidence for the case studies was collected with more than one collection technique (questionnaires and documentation).

3. Theoretical Triangulation. This was invoked by applying the theory of one academic discipline to the research situation within another discipline. In this case systems principles were applied to the discipline of software engineering.

The first triangulation method encouraged the researcher to collect information from multiple sources in order to corroborate a particular fact or phenomenon. When *real* triangulation took place the fact or phenomenon was supported by more than one source of evidence. Using data triangulation as part of the procedure for Step 13 helped establish one of the key measures of case study design quality from Table 14; *construct validity*.

Sub-Step 13-1: Collecting Documents, Records and Artifacts

The collection of documents, archival records, and physical artifacts from the software development projects selected for case studies was a straightforward process. All documents, records, and artifacts were stored in the case study database created in sub-step 13-3. However, the 4th type of evidence collected in case study research, questionnaires, was unique and warranted further examination.

Sub-Step 13-2: Collecting Evidence using Questionnaires

Case Study Questions

“Within the behavioral and social sciences, *psychometrics* has evolved as the subspecialty concerned with measuring psychological and social phenomena. Typically, the measurement procedure used is the questionnaire, and the variables of interest are part of a broader theoretical framework.” (DeVellis, 2003, p. 3) This was the situation in this research. The questionnaire and its particulars were elements of the larger theoretical framework. The use of the questionnaire as part of the case study was required in order

to collect evidence for the framework validation, from project participants, that was not present in the literature, records, or physical artifacts.

“More than with the other research strategies . . . case studies require an inquiring mind during data collection, not just before or after the activity. The ability to pose and ask good questions is therefore a prerequisite for case study investigators.” (Yin, 2003, p. 59) “Questions in a case study protocol can have five levels.” (Yin, 2003, p. 74)

Level	Characteristics
Level 1	Questions asked of specific interviewees.
Level 2	Questions asked of the individual case (these are questions in the case study protocol to be answered by the investigator during a single case, even when the single case is part of a larger, multiple-case study.
Level 3	Questions asked of the pattern of findings across multiple cases.
Level 4	Questions asked of an entire study – for example, calling on information beyond the case study evidence and including other literature or published data that may have been reviewed.
Level 5	Normative questions about policy recommendations and conclusions, going beyond the narrow scope of the study.

Table 45: Levels of Questions in a Case Study Protocol

Yin recommends “. . . you should only articulate Level 1 and Level 2 questions for data collection purposes.” (Yin, 2003, p. 74) It is important to note the principal difference between the Level 1 and Level 2 questions. Level 1 questions are those asked *in the field* while Level 2 questions originate with and are answered by the researcher.

Questions could have been asked in either a face-to-face interview or in a focused questionnaire. The questionnaire, when used in this manner, is an extension of the standardized interview and focuses on supplementing, reinforcing, and filling-in gaps that may not have been fully answered or are missing from the collection of documents, archival records, and physical artifacts collected in Sub-Step 13-1. This was important because “. . . specific information that may become relevant to a case study is not easily predictable . . . judgments may lead to the immediate need to search for additional evidence.” (Yin, 2003, p. 59)

Patton (1987) addresses three approaches for asking questions in order to collect qualitative data through in-depth, open-ended interviews:

1. The informal conversational interview. This technique relies entirely on the spontaneous generation of questions in the natural flow of an interview.
2. The standardized open-ended interview. This technique consists of a set of questions carefully worded and arranged for the purpose of taking each respondent through the same sequence and asking each respondent the same questions with essentially the same words.
3. The general interview guide approach. This technique uses a list of questions or issues that are to be explored in the course of the interview. An interview guide is prepared to make sure that essentially the same information is obtained from a number of people by covering the same material. The interview guide provides topics or subject areas about which the interviewer is free to explore, probe, and ask questions that will elucidate and illuminate that particular subject.

The 2nd approach, the standardized open-ended interview, was implemented through the use of a questionnaire. The questionnaire had the following advantages. (1) It saved the researcher travel expenses, and for a very low cost, expanded the questions asked to a wider group of potential interview subjects, and (2) Distance became an advantage because “. . . people can respond with assurance that their responses will be anonymous, and so they may be more truthful than they would be in a personal interview.” (Leedy & Ormrod, 2001, p. 197) The questionnaire had disadvantages as well. The questionnaire may overly constrain the dialog and may introduce some of the researcher’s own personal

bias into the questions. “Questionnaire construction is a tricky business,” (Leedy & Ormrod, 2001, p. 202) and is the subject of the following section.

Questionnaire Design Concepts

“The use of questionnaires in research is based on one basic, underlying assumption: Each individual question will work. This means that the respondent will be both willing and able to give truthful answers.” (Berdie, Anderson & Niebuhr, 1986, p. 1) The researcher acknowledged this underlying assumption and formally addressed reliability, validity, and representativeness in the purposeful design of the questionnaire.

1. Respondents. The questionnaire considered the context that surrounded the respondent and the ability to answer the question. Will the respondent have to search for historical data to answer the question? Will the respondent take the time to do this? Will the respondent have to rely on memory alone? In order to mitigate these issues “. . . the use of questionnaires should be limited to asking for information that is not directly available from other sources.” (Berdie, Anderson & Niebuhr, 1986, p. 2)
2. Reliability. Questionnaire reliability required the researcher to ensure that “. . . the questionnaire item consistently conveys the same meaning to all people in the population being surveyed.” (Berdie, Anderson & Niebuhr, 1986, p. 3) Therefore, questionnaire items used simple, clear, unambiguous language to ask questions that required no unwarranted assumptions, and did not give clues about preferred or desirable responses.
3. Validity. Questionnaire validity required the researcher to construct a valid question that “. . . stimulates accurate, relevant data.” (Berdie, Anderson &

Niebuhr, 1986, p. 3) The questionnaire considered the context that surrounded the respondent and the ability to answer the question. Will the respondent have to search for historical data to answer the question? Will the respondent take the time to do this? Will the respondent have to rely on memory alone?

4. Representativeness. Questionnaire representativeness addressed the generalizability of the results. The value of the results was a function of how well the actual responses matched the representative group the questionnaire was sampling.

Each of the questionnaire design concepts was addressed in the construction of the questionnaire.

Questionnaire Construction

The construction of the actual questionnaire addressed both the formal design concepts of the previous section and a number of practical matters concerning the respondent. Respondent related factors included the following items (Berdie, Anderson & Niebuhr, 1986):

1. Begin with a few interesting, non-threatening, interesting questions because questions that are *threatening* or *dull* may reduce the subject's likelihood of completing the questionnaire.
2. Group items into logically coherent sections. Questions that deal with a specific topic or those with the same response options should go together.
3. Make smooth transitions between sections, avoiding the appearance of a series of unrelated questions.

4. Do not put important questions at the end of the questionnaire.
5. Number questionnaire items so the respondent will not become confused.
6. Put the study title in bold type on the first page of the questionnaire.
7. Include brief but clear instructions for completing the form. Construct the questions so they do not require extensive instructions or examples. Print all instructions in boldface or italics.
8. Arrange the questionnaire so that the place where respondents mark their answers is close to the questionnaire, which encourages fewer mistakes.

The final element of construction dealt with the creation, structure, and responses for questions and was addressed as four questions.

What type of question do I ask? The questionnaire included four types: (1) dichotomous questions (those with only two answers), (2) open-ended questions (respondents are asked to express answers in their own words), (3) multiple choice questions (where the respondents chose from more than two options), and (4) ranking questions (those that ask the respondent to rank given response options according to some specified criteria).

What questions should I ask? The obvious answer was those that achieved the goals of the research. The less obvious issue was the effect the question would have on the respondent. Offensive, socially and politically sensitive questions were avoided.

How do I ask a question? Questions communicated something specific required for the research. The questionnaire used language that was familiar and appropriate for the targeted respondents (Moser & Kalton, 1971). The

questionnaire avoided *loaded questions* that suggested a response (Phillips, 1941).

Each question clearly indicated whether it required a factual answer or an opinionative answer. Questions asked for only one piece of information per question as a *double* question may have confused the respondent and made the question impossible to answer.

What response options may I use? The response options in the questionnaire affected the respondent's answers. The questionnaire was thoroughly tested to ensure that: (1) one response category was listed for every conceivable answer, (2) responses options were mutually exclusive, and (3) balanced scales were used in response options by including a equal number on each side of a middle position.

Table 46 is a convenient checklist used in constructing the questionnaire (Leedy & Ormrod, 2001).

Questionnaire Procedure

A small number of questionnaires were used to gather focused information from individuals that worked on each of the software development projects selected as case studies. The questionnaires were used to supplement and support the documents, archival records, and physical artifacts from the software development projects collected in sub-step 13-1. To ensure anonymity of participants, questionnaire respondents were referred to by number only. The questionnaires were strictly voluntary and each respondent was provided with background material that explained the purpose of the research and intent of the research study. The procedure used an asynchronous computer-based method to distribute questionnaires to the respondents. The questionnaire was placed on a web-

Guideline	Description
1. Keep it short.	The questionnaire should solicit only that information essential to the research. Use two tests: (1) What will I do with the information I am requesting? (2) Is it absolutely essential to have this information to solve part of the research?
2. Simple, clear language.	Use simple, clear, unambiguous language. Construct questions that communicate exactly what you want to know. Avoid terms that the respondents may not understand.
3. Check assumptions.	Review the question to ensure that underlying assumptions are not ambiguous.
4. Give no clues.	Word questions in ways that do not give clues about preferred or more desirable responses.
5. Check for consistency.	When a question addresses something that make elicit a "politically or socially correct" answer a countercheck question may be incorporated. Insert a questions at some distance from the first question which verifies the consistency with which the respondent has answered the first question.
6. Response processing.	Determine, in advance, how you will process the responses.
7. Keep the task simple	Make the questionnaire as simple to read and respond to as possible. Place high-value on the respondent's time.
8. Provide clear direction.	Communicate exactly how you want people to respond. Do not assume they are familiar with your scales.
9. Give rationale for questions.	Give people a reason o want to respond. At a minimum each question should have a purpose, and in one way or another, you should make that purpose clear.
10. Make the questionnaire attractive.	The questionnaire should be of professional caliber; have clean lines, crystal-clear typing, and, perhaps two or more colors.
11. Conduct a pilot test.	Conduct a trial run of the questionnaire to address problem prior to the use in the research.
12. Scrutinize the final product.	Scrutinize the questionnaire, item-by-item, in a quality test for precision of expression, objectivity, relevance, and probability of favorable reception.

Table 46: Guidelines for Questionnaire Development

server and accessed by the respondents as time permitted. Each respondent had an opportunity to review the materials prior to publication. The dissertation research, both oral and written, used aggregated questionnaire data and was presented in such a way that others would not be aware of how a particular participant responded in the questionnaire. The researcher was committed to the use of non-attributable reporting methods as an integral part of the research procedure. The questionnaire results were stored in the case study database created in sub-step 13-3.

Sub-Step 13-3: Creation of a Case Study Database

Yin (2003) states that it is absolutely essential to separate the case study materials into two distinctly independent collections.

1. **Collected Evidence.** All documents, archival records, physical artifacts, questionnaires, and researcher's notes. This should be in the form of a formal, presentable database that can be viewed by other investigators.
2. **Investigator's Conclusions.** The formal conclusions from the case studies. These include the analysis and interpretation of the case study evidence contained in the formal database as well as the final report of the findings.

The researcher created a formal case study database in the NVivo qualitative analysis software to ensure that the raw data from the case study database was available for independent inspection. The creation of a reviewable and objective database helped to establish one of the key measures of case study design quality from Table 14; *reliability*.

Sub-Step 13-4: Maintenance of the Chain of Evidence

The final action associated with the collection of case study evidence addressed the ability of an external observer to follow the derivation of evidence, ranging from initial questions to the ultimate conclusions of the case study. "Such a principle is based on a notion similar to that used in forensic investigations." (Yin, 2003, p. 105) It is important to note that the entire research sequence was reversible and could be traced in either direction. For example, an external observer would be able to trace the logic from conclusions back to the initial research question or from the question to conclusions, as in Figure 31. The ability to clearly and unequivocally demonstrate the chain of evidence, with clear cross-referencing between collected evidence, methodological procedures, and

conclusions helped establish one of the key measures of case study design quality from Table 14; *construct validity*.

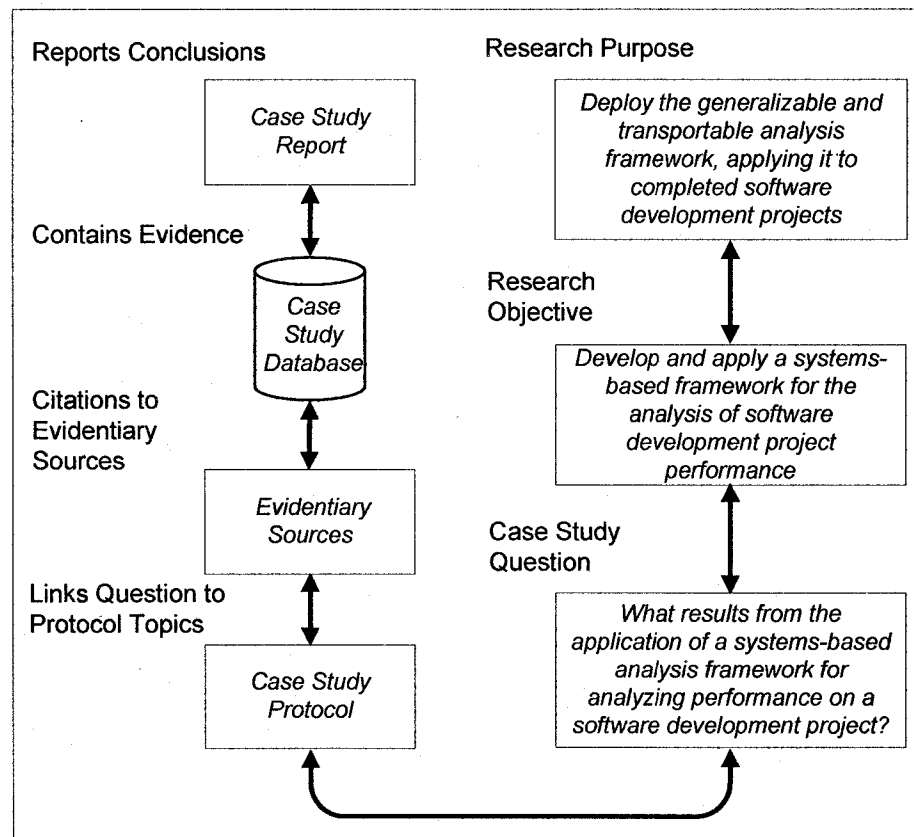


Figure 31: The Case Study Chain of Evidence

Step 14: Analysis of Case Study Evidence

In the 14th step the researcher analyzed the evidence collected in the previous step.

This is addressed by the 2nd section of the analytic strategy: quantitative and qualitative analysis. Yin (2003) provides the following warning to case study researchers:

The analysis of case study evidence is one of the least developed and most difficult aspects of doing case studies. . . . Unlike statistical analysis, there are few fixed formulas or cookbook recipes to guide the novice (one of the few texts providing useful advice is Miles and Huberman, 1994). (pp. 109-110)

The 2nd section of the analytic strategy was used to guide the researcher in analyzing the evidence. In this step both quantitative and qualitative analysis methods were applied to

the decision alternatives presented within the problem domain. Seaman recommends the use of coding as a valid method for software engineering studies because coding can “. . . extract values for quantitative variables from qualitative data (often collected from observations or interviews) in order to perform some type of quantitative or statistical analysis” (Seaman, 1999, p. 563)

The documents, archival records, and questionnaire data were transcribed and input to the case study database in the NVivo qualitative analysis software. The evidence in the case study database was subjected to the open-coding technique (see Table 26) described in Step 4. As each applicable item was coded it was coded against a specific node in the hierarchical tree nodes in NVivo. The NVivo hierarchical tree node was constructed from the verified software development project framework completed in Step 9. The completed framework’s hierarchical structure of measurement objects and constructs was the structure for the hierarchical tree-node against which all case study data was coded. As the tree nodes were populated with coded data from the documents, archival records, and questionnaires, the key meaning of the case study data became clearer, and served as measures for the validity of the framework.

Step 15: Interpretation of Case Study Evidence

In the 15th step the researcher interpreted the evidence collected in the previous step. Because the cases were, according to Lijphart’s (1971) typology, theory confirming/infirming cases “. . . the case study is a test of the proposition, which may turn out to be confirmed or infirmed by it.” (Lijphart, 1971, p. 692). For this research the cases were confirming or infirming the framework’s ability to predict software development project performance.

This was addressed by the 3rd section of the analytic strategy: interpretation. The researcher formally interpreted the analysis results and their implication for the framework. This step was subjective but was constrained by the structure for the inquiry, the framework and the general deductive method. The interpretive section of the general analytic strategy was selected from among three interpretive strategies recommended by Yin (2003). The interpretive strategy presented in Table 47 was used for this research.

Interpretive Strategy	Description of Strategy
Theoretical propositions	The research study purpose Research element objective Research question two will be used to guide the interpretation.

Table 47: Interpretive Strategy for the Case Study

The interpretive strategy in Table 47 underlies the specific analytic technique used to interpret the study. “The analytic technique may be selected from among six techniques.” (Yin, 2003, p. 116-137) The technique used to interpret the case studies in this research used a special type of pattern-matching logic called explanation building. In the general pattern-matching technique the researcher compared an empirically based pattern with a predicted one where the links were related to either the dependent or the independent variables of the study, or both. In the explanation building derivative the researcher explained the phenomenon by stipulating a presumed set of causal links about it. “In most studies, the links may be complex and difficult to measure in any precise manner.” (Yin, 2003, p. 120) Yin warns that because explanation building case studies are reported in narrative form they lack precision. Therefore, “. . . the better case studies are the ones in which the explanations have reflected some theoretically significant propositions.” (2003, p. 120) The procedure for explanation building was iterative and included six sub-steps (Yin, 2003):

Sub-Step 15-1: Making an Initial Statement

The researcher made an initial statement about the framework's ability to predict software development project performance. This was a subjective analysis based on the FSE Framework's systems-based theoretical construction.

Sub-Step 15-2: Compare Findings of 1st Case

The researcher compared the findings of the 1st case against the research purpose, objectives, and questions as instantiated in the framework. This involved matching the empirical data from the 1st case against the measurement objects in the FSE Framework. The case study evidence arrayed against the FSE Framework provided the basis for a sound objective analysis.

Sub-Step 15-3: Revise the Initial Statement

The researcher revised the initial subjective statement about the framework's ability to predict software development project performance. The revised statement was based upon an objective analysis of the predictive ability of the FSE Framework based upon empirical case study evidence from the 1st case that was presented against the FSE Framework.

Sub-Step 15-4: Making an Follow-On Statement

The researcher made a *follow-on* statement about the framework's continued ability to predict software development project performance. This was a slightly less subjective analysis based upon a single use of the systems-based FSE Framework.

Sub-Step 15-5: Compare Findings of 2nd Case

The researcher compared the empirical evidence from the 2nd case against the measurement objects in the FSE Framework. The case study evidence arrayed against

the FSE Framework provided the basis for a 2nd objective analysis of the framework's ability to predict software development project performance.

Sub-Step 15-6: Revise the "Follow-On" Statement

The researcher revised the subjective *follow-on* statement about the framework's ability to predict software development project performance. The revised statement was now based upon two objective analyses of the predictive ability of the FSE Framework based upon empirical case study evidence from the 1st and 2nd cases.

The result of the six step iterative explanation building process was cross-case analysis, not simply the analysis of each individual case. Yin warns that the researcher must be extremely careful as the iterative process progresses for the "... investigator may slowly begin to drift away from the original topic of interest. Constant reference to the original purpose of the inquiry and the possible alternative explanations may help to reduce this potential problem." (Yin, 2003, p. 122) The use of an analytic strategy (step 12), a case study database (sub-step 13-3), and strict adherence to the chain-of-evidence (sub-step 13-4) were additional safeguards provided for the researcher.

The final products for the 5th phase were the interpretation of the case studies and implications for the framework.

Procedure for Phase 6: Reporting the Findings

The goal of the 6th phase was to report the findings in the dissertation and the preservation of the evidentiary data for use in a follow-on article in a scholarly journal.

Step 16: Reporting the Case Study

In the 16th step the researcher brought the results and findings to closure in the dissertation. The report for each case was prepared as a separate Appendix within the dissertation. Because the interpretation step used explanation building as the primary

technique, the report was in narrative style. Patton (1987) provides an overview of the case study report as follows.

The case study is a readable, descriptive picture of a person or program that makes accessible to the reader all the information necessary to understand that person or program. The case study is presented either chronologically or thematically (sometimes both). The case study presents a holistic portrayal of a person or program. (p. 149)

Yin (2003) addresses four specific concerns related to the composition and reporting requirements for case studies.

1. Targeting Case Study Reports

The researcher considered the likely or preferred audience as the starting point when composing the case study. This research has as its principal audience, the dissertation committee and academic colleagues. The wider secondary audience will be the software engineering community with a focus on software development project managers. Table 48 lists the needs of the primary and secondary audiences for this research.

Audience	Needs of the Audience
Dissertation Committee	Mastery of the methodology Inclusion of theoretical issues Careful, thorough, and rigorous research
Academic colleagues	Connection between the case study and previous theoretical research
Software Engineers	Ability to use the findings to improve software development project performance

Table 48: Needs of the Case Study Audience

2. Case Study Reports as Part of Larger Multi-Method Studies

This research has included the inductively developed holistic, structured, systemic framework for software development projects as a principal element. The framework

served as a systems-based lens through which the two software development projects were viewed. The case studies were used to analyze and validate the framework.

3. Illustrative Structures for Case Study Compositions

The organization of the report, beyond the format dictated by the dissertation and journal article, were of import when preparing a case study. Yin (2003) suggests six illustrative structures that can be used with a variety of formats. This research used a theory-building structure in which the content followed a theory-building logic that produced a compelling statement about the utility of the holistic, structured, systemic framework for software development.

4. Procedures in Doing a Case Study Report

Aside from the method steps and techniques used in selecting, collecting and analyzing the case study evidence, there were three important procedures that pertained specifically to case studies that the researcher considered. (1) The first procedure encouraged the case study researcher to start writing as soon as possible. The chapters on the literature review, methodology, research design and procedure, and references were written early in the process and permitted the researcher to spend more time in analysis. (2) The second procedure concerned the identities of the case and the participants. Were they to be identified or remain anonymous? The researcher made this decision early-on and received approval from the Institutional Review Board which was presented in Appendix A. (3) The third procedure concerned what constitutes an exemplary case study. Yin (2003) provides five general characteristics of exemplary case studies which are listed in Table 49. In addition to the characteristics in Table 49 a general guide for dissertation evaluation was utilized as part of the overall review (Lovitts, 2005).

Characteristic	Explanation
Must Be Significant	<ul style="list-style-type: none"> • The individual case or cases are unusual and of general interest • The underlying issues are nationally important, either in theoretical terms or in policy or practical terms • Both of the preceding conditions have been met
Must Be Complete	<ul style="list-style-type: none"> • The distinction between the phenomenon being studied and its context are given explicit attention • The study demonstrates that the investigator expended exhaustive effort in collecting relevant evidence • The case study did not have artificial time or resource constraints which led to premature reporting
Must Consider Alternative Perspectives	The investigator must seek those alternatives that most seriously challenge the design of the case study
Must Display Sufficient Evidence	<ul style="list-style-type: none"> • The researcher judiciously and effectively presents the most relevant evidence so that the reader can reach an independent judgment regarding the merits of the analysis • Evidence is presented neutrally, with both supporting and challenging data • Present enough evidence to gain the reader's confidence • Indication that the researcher attended to the validity of the evidence
Must Be Composed in an Engaging Manner	A clear writing style that constantly entices the reader to continue reading

Table 49: Characteristics of an Exemplary Case Study

The final products for the 6th phase were the completed case studies and their implications for the framework.

Summary of the Quantitative and Reporting Procedures

The theory building triangle from the previous section (see Figure 28) depicted the inductive process used to develop the theoretical framework in the qualitative element of the research. Carlile and Christensen (2005) also used the triangle to describe the validation of theory in what they call a *transition from descriptive theory to normative theory*. The normative theory triangle in Figure 32 shows how the researcher cycled deductively from the top to the bottom to confirm the framework developed in the inductive processes. Once again, it is interesting to note that this conforms to Bacon's expression of *descending* which he gave to the process of deduction and the more common expression of "... coming down to particulars." (Venn, 1907/1973, p. 364)

The overall transition from induction to deduction (in what Carlile and Christensen call descriptive theory to normative theory) can be shown by depicting the theory building triangle and the normative theory triangle together.

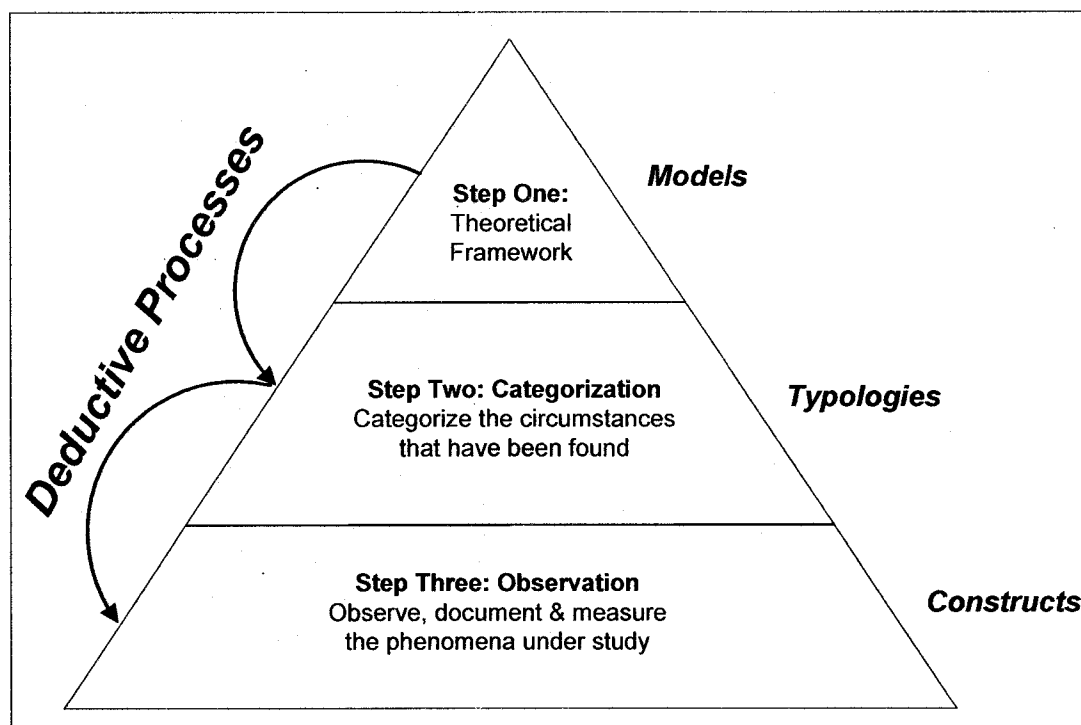


Figure 32: Normative Theory Triangle

Adapted from Carlile, P.R. & Christensen, C.M. (2005). "The Cycles of Theory Building in Management Research," Unpublished Manuscript, p. 6.

Figure 33 shows the process through which the inductively developed framework was validated by conducting careful, field-based case study research, making the transition from descriptive to normative theory. This conforms to Jevons' (1877/1913) statement on deduction and induction.

As generally stated, deduction consists in passing from more general to less general truths; induction is the contrary process from less to more general truths. We may however describe the difference in another manner. In deduction we are engaged in developing the consequences of a law. We learn the meaning, contents, results or inferences, which attach to any given proposition. Induction is the exactly inverse process. (p. 11)

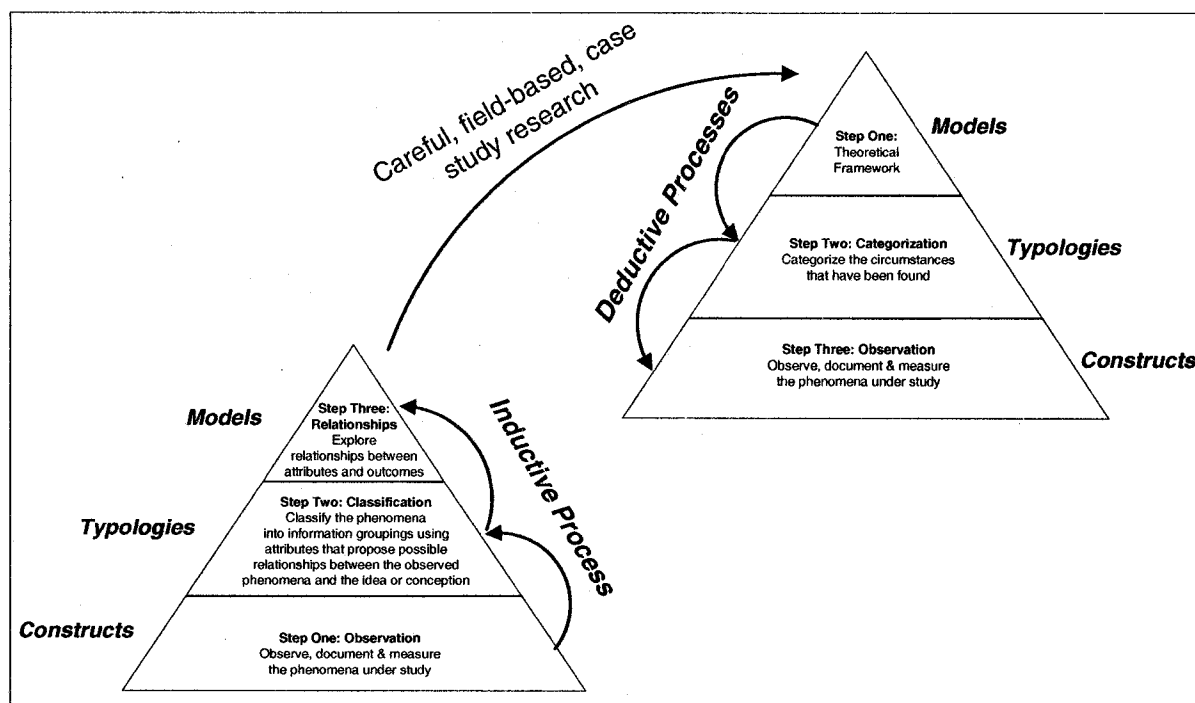


Figure 33: The Transition from Descriptive Theory to Normative Theory

Adapted from Carlile, P.R. & Christensen, C.M. (2005). "The Cycles of Theory Building in Management Research," Unpublished Manuscript, p. 6.

DATA ANALYSIS SOFTWARE

The analysis of data that occurred during the Generalization Phase of the *Discoverers' Induction* Method was assisted by a computer program for qualitative data analysis. A wide variety of computer programs specifically designed to assist researcher's doing qualitative analysis were available to assist qualitative researchers. Weitzman & Miles (1995) developed a software sourcebook that includes thorough reviews of 24 of computer programs in five families of software applications. The section on the family of Code-Based Theory-Builders analyzes five programs that can accomplish the tasks of (1) coding and retrieving data, and (2) building and validating theory.

- Code and Retrieve Data: “Apply keywords to meaningful segments or *chunks*, such as lines, sentences or paragraphs of text. They help you set up the kinds of *chunks* you want to use, to develop a list of codes, to attach the right codes to the right chunks – and then to search for and retrieve all the chunks to which one or several codes have been applied.” (Weitzman & Miles, 1995, p. 148)
- Build and Validate Theory: “They help you apply categories and outlines, extending your coding scheme; to annotate your data, write memos, and link these to your codes; to formulate and test propositions or hypotheses; and sometimes see graphical representations among your concepts.” (Weitzman & Miles, 1995, p. 204)

Of the five (5) programs reviewed in this category, Weitzman and Miles found that the Non-numerical Unstructured Data Indexing, Searching and Theorizing (NUD*IST) tool was “. . . one of the best thought-out programs around.” (1995, p. 238) The newest version of NUD*IST is called NVivo7 and was the analysis tool selected for use in this research. The marketing material from QSR states that:

*NVivo7 combines cutting edge innovation with the best features of QSR's pragmatic, robust workhorse N6 (formerly NUD*IST) and flexible, fluid analysis tool NVivo 2. The result is a powerful yet user-friendly program that accommodates the widest range of research methods. NVivo7 allows you to combine subtle analysis with linking, shaping, searching and modeling. Regardless of the type of data or the language it is in, NVivo7 is also ideal for team projects and research involving multiple methods. If your challenge involves handling very rich text-based information, where deep levels of analysis on both small and large volumes of data are required, then NVivo7 is your answer.*

NVivo7, from QSR International was used to document and track the data decomposition, organization, and analysis processes. NVivo7 helped the researcher manage the data, while creating and developing ideas and theories as the researcher gained understanding of the phenomena. The NVivo7 software was an invaluable tool that the researcher used when implementing the qualitative procedure for coding data. The importance of the research method over the software was stressed in the authoritative

guide to the use of the software (Richards, 1999) and proved to be extremely helpful in understanding how the software supported the inductive process. Another excellent guide (Gibbs, 2002) has a particularly well-written section the role of grounded theory principles in the analysis of qualitative data.

SUMMARY

This chapter has provided an outline of the research design and the specific details for the methods, procedures, and techniques used in the two primary elements of the research. The specific procedures and techniques in this chapter provided the formal steps used to obtain the research results described in the following chapter.

The detailed steps used in the development of the framework were of particular import. *Discoverers' Induction* is a Victorian Era (Snyder, 2003) method that was supplemented with modern techniques for decomposing and classifying empirical facts during the construction of the conception. The detailed steps used in the inductive element of the research (see Table 24 and Figure 27) were established to increase reliability by ensuring replication and controls were in place for future researchers. The internal validity, reliability and objectivity of the research were supported by the use of a software database (NVivo 7). The NVivo 7 software database served as a repository for the induction and each case study and provided a proper chain of evidence for the research.

The next chapter will explicitly state the data sources and data collection methods used in the research as well as any unique procedures and techniques used in the development and validation of the structured systemic framework for software development project performance.

CHAPTER V: RESEARCH RESULTS

This chapter presents the results of the research and is subdivided into two major sections. The first section is the *Inductive Development of the Framework*. This section explains the results of the framework construction. The second section is the *Case Study Validation of the Framework*. This section addresses the results of the application of the framework to two real-world software development projects.

INDUCTIVE DEVELOPMENT OF THE FRAMEWORK

The framework was constructed inductively. The construction followed the steps in the Augmented *Discoverers' Induction* Procedure presented in Chapter 3 and depicted in Figure 27. This section will include a high-level review of the key points in the inductive procedure and present the construction results; the detailed framework.

Collection and Verification of Facts

The formal induction began with the completion of the literature review in Chapter 2. The journal articles synthesized from the scholarly literature provided the empirical facts required for the inductive development of the literature based, structured, systemic framework. In addition, the synthesized literature ensured that the researcher was exposed to a range of ideas, concepts and theories from the extant literature. This enhanced the induction by formally linking the emerging framework to the extant work. Because the information synthesized in the literature review was the primary source of empirical data for the colligation; it had a direct affect on the validity of the induction. The researcher employed an outside expert to verify that the literature review had captured all of the relevant information in order to directly address content validity. The expert followed the verification guidelines in Appendix B and provided additional

literature sources that added depth and breadth to the empirical facts included in the induction. The recommendations of the expert are contained in Appendix D.

The researcher revised the original literature review to include the additional scholarly articles recommended by the outside expert. The 34 scholarly articles in Figure 11 served as the database for the induction. Each article was input to the NVivo toolset to ensure direct linkage and traceability between the emerging framework concepts, categories, and subcategories and the empirical facts in the 34 articles.

Decomposition of Facts

The facts in the 34 articles were subjected to the *open coding technique*. A line-by-line analysis of each of the 34 documents was conducted. Documents were scrutinized for commonalities that reflected categories, or themes, within the data. Codes were attached to *chunks of information* which consisted of phrases, sentences, or whole paragraphs. This step was greatly enhanced through the use of the NVivo code-based theory building tool. The 34 articles were decomposed into 481 *chunks of information* called open-coded nodes; each of which was directly linked to the source document.

Classification of Information Groupings

Relationships between the 481 open-coded nodes were developed using the *axial coding technique*. The goal was to simplify and organize the 481 open-coded nodes into *information groupings* that proposed possible relationships between the data and the idea or conception about the relationship between systems theory and software development project performance. The *chunks of information* contained in the *information groupings* were organized into themes or subcategories that had significance for the construction of the framework. The open coding technique reduced the large data set to a smaller set of themes intended to describe the software development framework. The 481 open-coded

nodes were placed in *information groupings* that looked very much like a descriptive typology. This element of the research was iterative and found the researcher moving back and forth among the open coded nodes and the axial coding *information groupings*, continually refining the information structure or typology. “Cycles of decontextualization/recontextualization and exploration of relationships between categories in the hierarchical network of codes led both to splitting of categories and the pruning and consolidation of others.” (Araujo, 1995, p. 103) As before, NVivo provided invaluable assistance in the management and interpretation of the 481 open-coded nodes. NVivo provided the ability to take the open-coded nodes and attach them to a hierarchical structure of *tree-nodes*. It is important to note that the final interrelated data structure of NVivo tree-nodes contained all of the information used to build the theory underlying the framework. In this case each of the 481 nodes was assigned to a subcategory. 33 subcategories were created to relate the 481 information chunks and are depicted in Figure 34.

Information Reduction and Relationships

This was the most intellectually challenging step in the development of the framework. The reduction of the 33 subcategories into categories and concepts for inclusion in the underlying theory for the framework required the researcher to address both *comprehensiveness* and *parsimony* of information. Knowing that there were many relevant factors in the subcategories but that many others added little additional value were key points in selecting the essential data to be carried forward and included in the categories, concepts, and theory. Essential data were those that (1) could be related to other categories, (2) appeared frequently, (3) had relations that were logical and not forced to fit with other data, and (4) could succinctly explain and support the relevant

elements of the category or concept. The criteria invoked to control subcategory and category creation required the demonstration of a range of variability that accounted for all data but that showed theoretical saturation because no new dimensions or properties emerged from the information contained within the definition established for the subcategory or category.

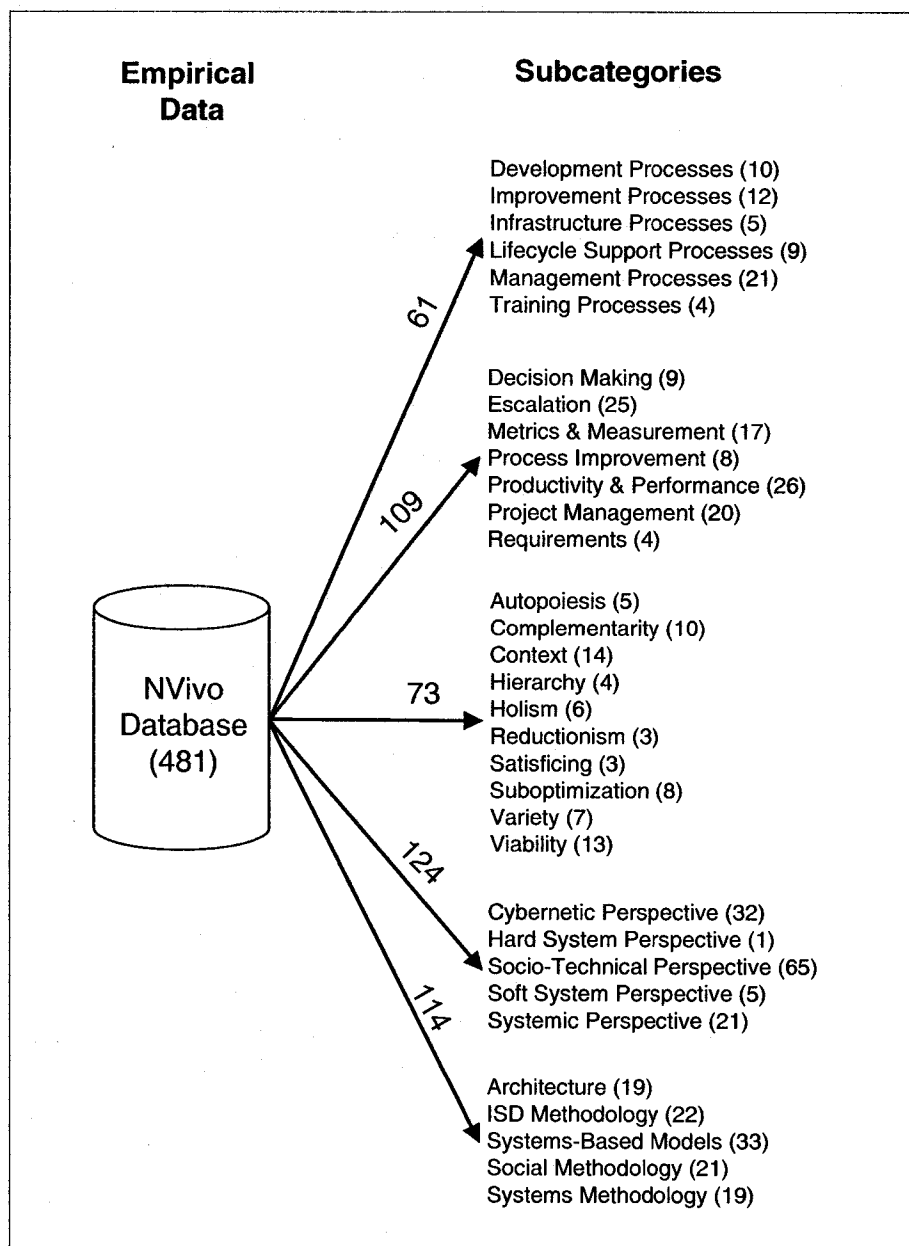


Figure 34: Subcategories

Although the NVivo code-based theory building tool was helpful, it is important to note that this step in the inductive process was creative, intellectual work. The researcher found that the automated tool was a poor substitute for the human mind and the effort and intelligence required to track down and recognize patterns and consistencies in the data. The researcher settled on a systematic set of hierarchical relationships with five elements: open coded nodes (the empirical *information chunks*), subcategories (the *information groupings*), categories, concepts, and the theory for the framework. Five categories emerged from the 33 subcategories.

1. Software Development Project Processes: The category that contained all subcategory information related to the six ISO/IEC software development functions and the supporting software development processes.
2. Software Development Project Characteristics: The category that contained all subcategory information related to the characteristics and empirically reported data from software development projects.
3. Systems Principles: The category that contained all subcategory information related to systems principles.
4. Systems Perspectives: The category that related all subcategory information related to systems perspectives.
5. Framework Form: The category that related all subcategory information related to the systems-based models and methodologies.

Three concepts emerged from the five data categories:

- Foundation: The concept that provided a basis for the theoretical framework that was founded upon recognized systems principles.
- Structure: The concept that provided a solution structure for the theoretical framework based upon the proven perspectives of a synthesis of systemic methodologies and systems-based models.
- Elements: The concept that provided the essential functions and processes for the theoretical framework that are characteristic of software development projects.

The hierarchical relationship between subcategories, categories, and concepts is presented in Figure 35.

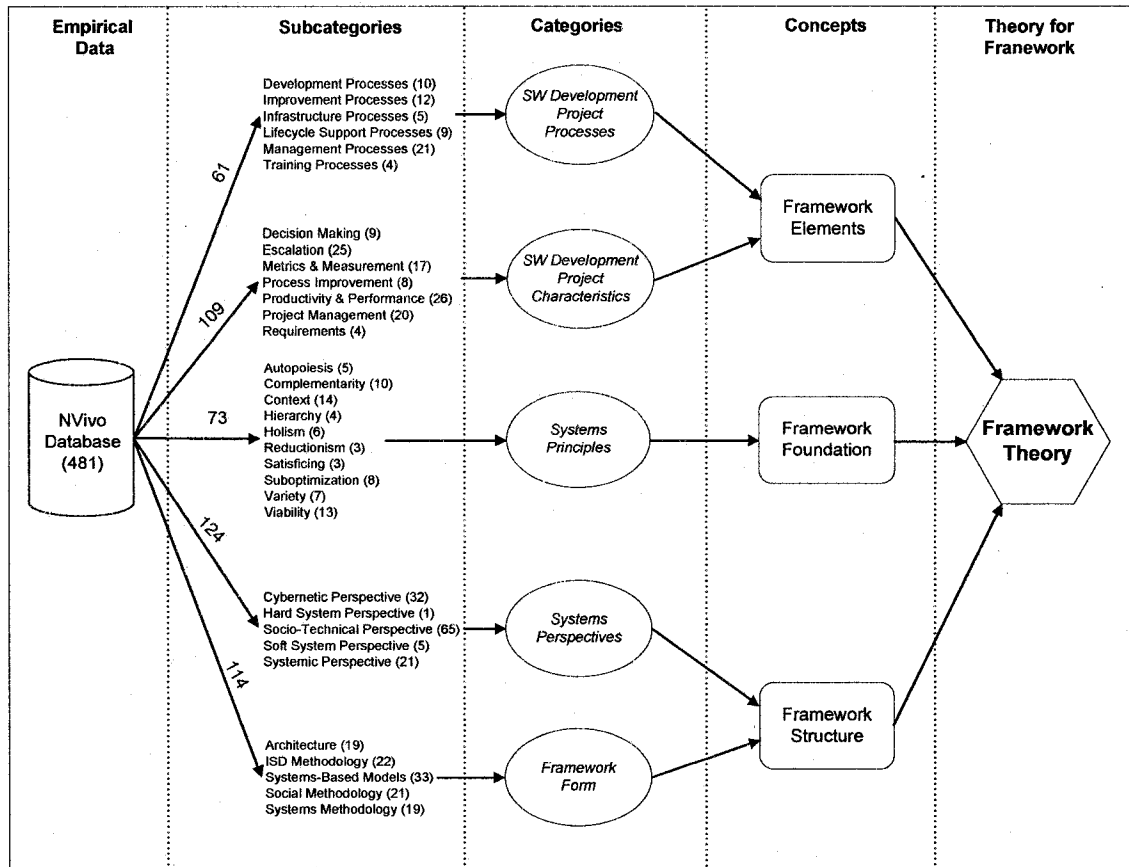


Figure 35: Tree Node Hierarchy

Synthesis of Concepts

The development of the underlying theory for the framework required the researcher to select essential data from the concepts. Figure 36 shows how the five categories and three concepts were logically assembled to influence the underlying theory for the framework. The essential empirical data contained within the three concepts were used to produce the framework. The hierarchical structure contained in the tree-nodes of NVivo provided assurance that the 481 *chunks of information* in the 33 *information groupings* were linked to the original empirical data in the 34 scholarly articles from the

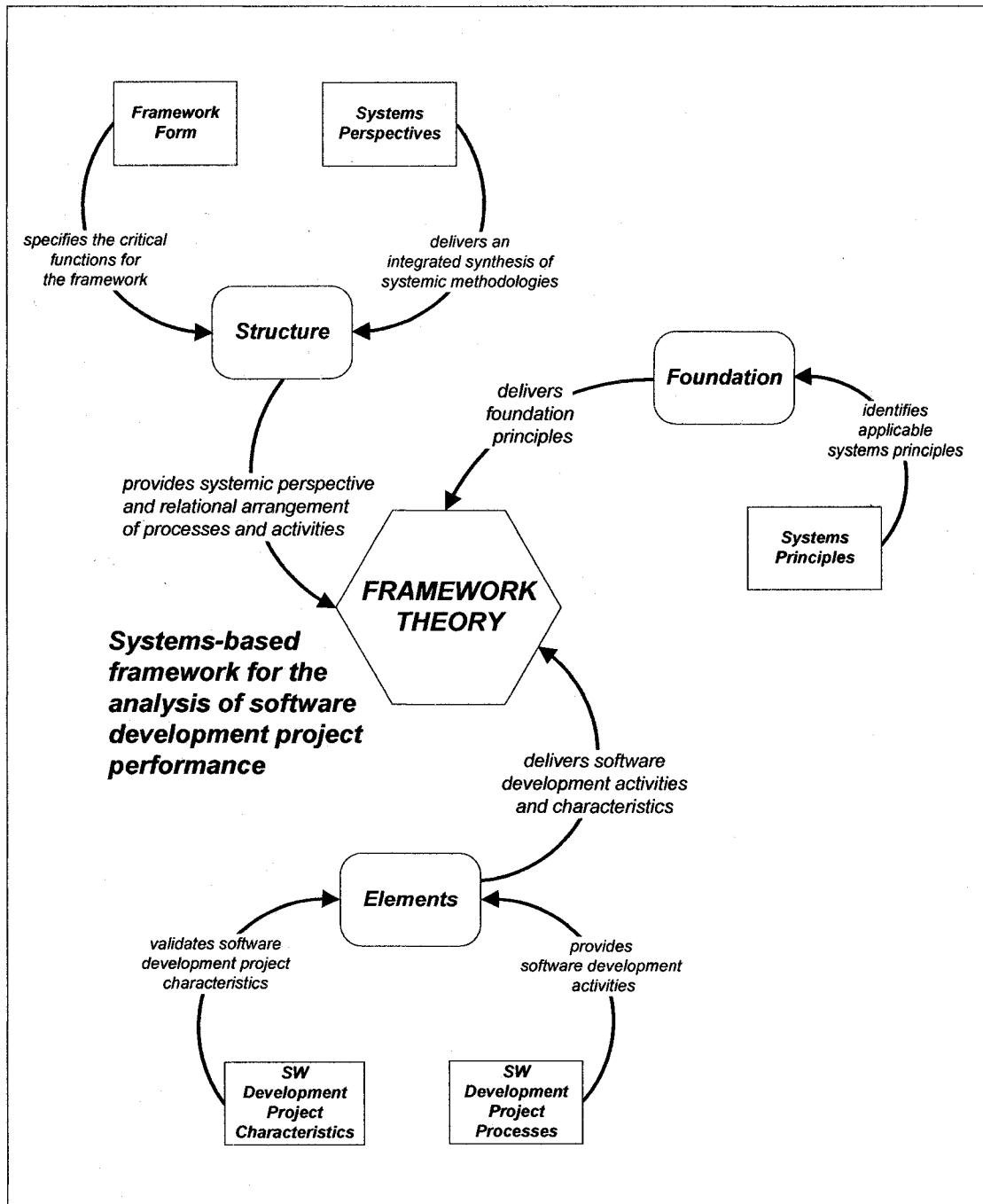


Figure 36: How Categories and Concepts Influenced the Framework

synthesized literature review. The principal concepts that are the theoretical basis for the actual framework, their hierarchical category and concept, and the source of the empirical data, are presented in Table 50.

Theoretical Concept	Hierarchical Category	Hierarchical Concept	Empirical Data Source
Major software development functions 1. Functions 2. Processes	SW Dev Project Processes	Framework Elements	1. ISO/IEC Standard 12207 (1995) 2. CMMI SM (2002)
Software development project characteristics for 1. Decision Making 2. Escalation - Separation of responsibility for evaluation (External Audit Process) 3. Metrics & Measurement	SW Dev Project Characteristics	Framework Elements	1. Keil & Robey (1999) 2. ISO/IEC Standard 12207 (1995) 3. Jones (2004)
Systems Principles 1. Complementarity 2. Context 3. Suboptimization 4. Variety 5. Viability	Systems Principles	Framework Foundation	1. Bohr (1928, 1937) 2. Ackoff (1979a, 1979b) and Keating, Kauffmann & Dryer (2001) 3. Hitch (1953) 4. Ashby (1956) and Conant & Ashby (1970) 5. Beer (1984), Espejo (2004) and Yolles (2004)
Systems Perspectives 1. Cybernetic Perspective 2. Socio-Technical Perspective	Systems Perspectives	Framework Structure	1. Beer (1984), Espejo (2004), Yolles (2004) 2. Shani, Grant, Krishnan & Thompson (1992); Jacobs, Keating & Fernandez (2000); and Shani & Sena (1994), Wallace, Keil & Rai (2004)
Systems-Based Models and Methods 1. Systems-Based Models (VSM and Three Lens) 2. Social Methodology 3. Systems Methodology	Framework Form	Framework Structure	1. Beer (1999), Espejo (2004), Yolles (2004) 2. Boloix & Robillard (1995) 3. Walton (2004)

Table 50: Sources for the Theoretical Concepts

Figure 37 shows how the essential concepts from Table 50 came together to form the theoretical basis for the framework.

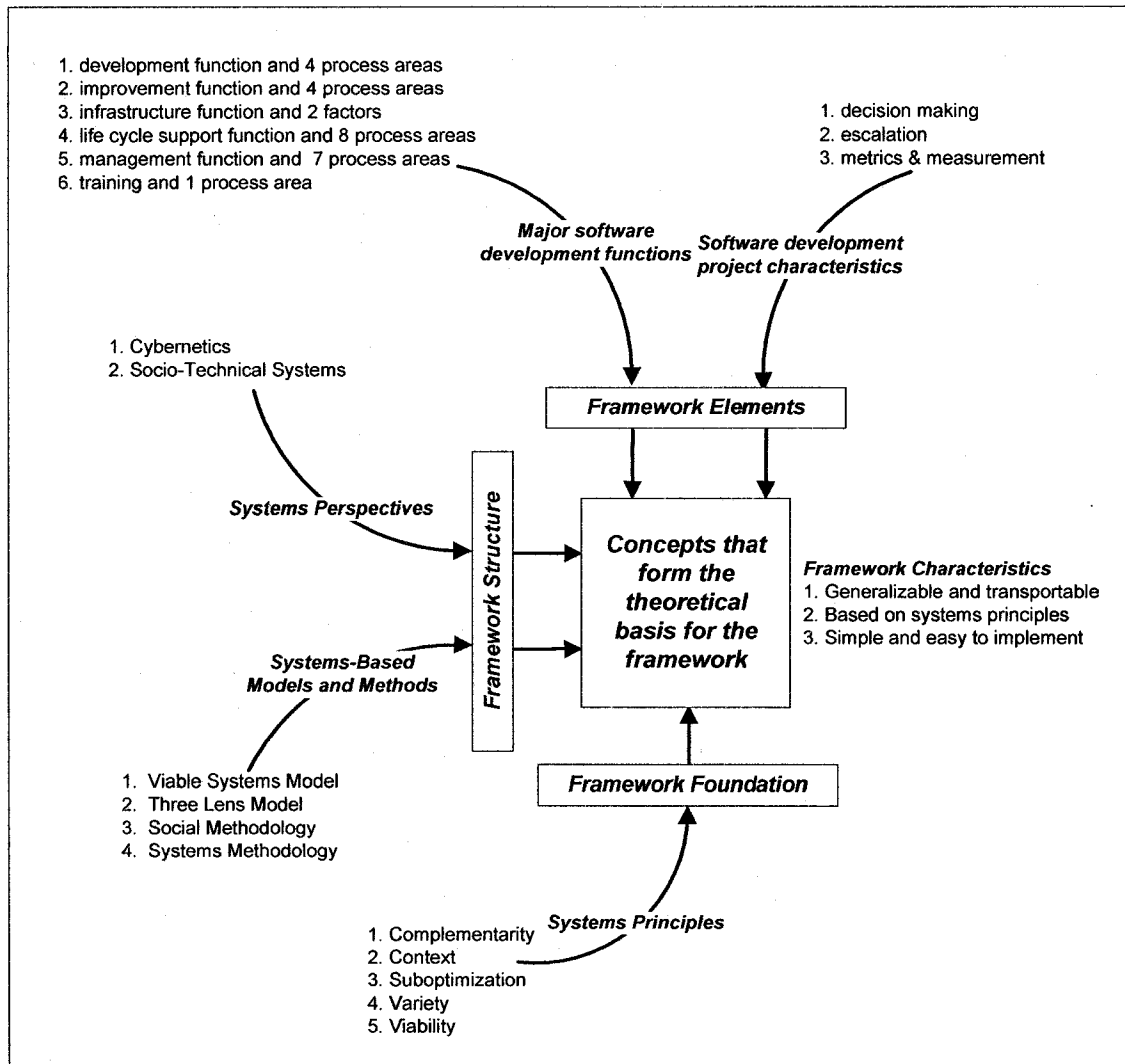


Figure 37: Theoretical Concepts Underlying the Framework

High-Level Structure of the Framework

Construction of the framework from the theoretical concepts was constrained by the framework features established in Chapter 4 (see Step 8). The governing features were a compilation of the boundary conditions, utilitarian characteristics (Table 35), and pragmatic factors (Table 36) established for the framework:

1. *Generalizable and transportable.* The framework must be able to be used on the full-range of worldwide software development projects and remain unconstrained by the rapid evolution of development mechanics (languages, analysis and design methods, applications, etc.).

2. *Analysis is based upon systems principles.* The framework analysis must be based upon perspectives and solutions founded on systems principles.
3. *Boundary conditions.* The framework is an approximation of theory on the theory continuum as a theory of the middle range.
4. *Explanatory potential.* The framework must establish the substantive meaning of the constructs, measurement objects and their relationships.
5. *Predictive adequacy.* The framework validates its substantive meaning by comparison with empirical evidence.
6. *Simple and easy to implement.* The framework must be easily understood by software professionals and be presented in a form and language that requires little or no formal training.

The shape and elements of the framework were a direct result of the application of the underlying theoretical concepts within the six governance factors. Figure 38 is a high-level view of the constructs and measurement objects.

It is important to note that all of the underlying theoretical concepts have been transformed to the theory (framework) and construct elements. A construct is a concept; but has the added meaning of having been deliberately and consciously invented by the researcher from his own imagination, to represent something that does not exist as an isolated, observable dimension of behavior (Nunnally, 1967; Kerlinger & Lee, 2000). The 14 constructs in Figure 38 bridge the gap between the theory (framework) and measurable empirical phenomena (Costner, 1969). In turn, each construct was supported by objects that have attributes which have criteria subject to measurement (i.e. yielding a measure). A measure is defined as “. . . an observed score gathered through self-report, interview, observation, or some other means.” (Edwards & Bagozzi, 2000, p. 156) The measures were important because they were the linkage between observable, real-world, empirical facts and the unobservable constructs in the theoretical framework.

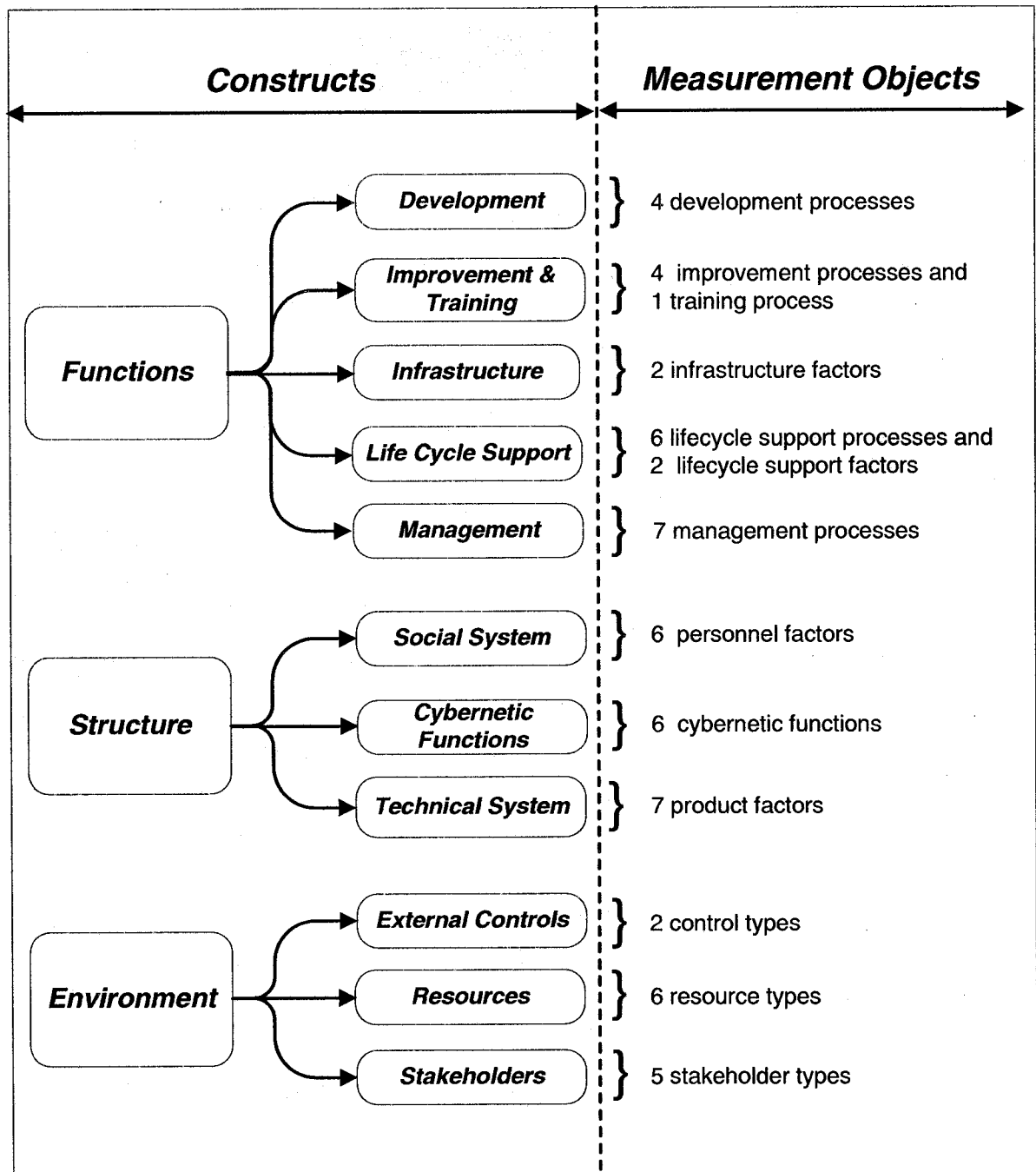


Figure 38: High-Level Framework and Construct Elements

The individual measures supporting the constructs were designed by the researcher as reflexive measures; that is, they represent reflections or manifestations of the unobservable construct and react to variation in the construct. The direct reflective model specifies the relationship between the construct and its measures, factor loading,

and measurement error (Edwards & Bagozzi, 2000). The causal nature of the relationship between constructs and measures has been the focus of continuing debate, although the literature suggests an emerging consensus based on four conditions for causality (Edwards & Bagozzi, 2000):

1. Distinct entities. The construct and measure must be distinct. This requirement is satisfied definitionally.
2. Association. The construct and measure must co-vary. This is a thorny issue because the researcher has direct access to the measure but not to the construct (Nunnally, 1967). Therefore, the researcher must rely on the use of co-variances among the multiple measures of the construct to infer the true relationship. This condition will be the subject of continuous improvement efforts as the framework becomes more extensible.
3. Temporal Precedence. This addresses whether change in the construct precedes, accompanies, or follows the change in the measure. Because the researcher has direct access to the measure but not to the construct (Nunnally, 1967) temporal precedence cannot be determined directly.
4. Elimination of Rival Causal Explanations. "Ruling out rival causal explanations is a daunting task that cannot be reduced to universal prescriptions." (Edwards & Bagozzi, 2000, p. 160) This condition for causality will continue to be difficult to completely satisfy because of the complex of software development activities that may induce a relationship between the cause and effect. However, the principle causes have been

addressed in this research and will continue to be the subject of continuous improvement efforts as the framework becomes more extensible.

The relationship between the reflexive measures and the constructs in the framework satisfied three of the four conditions for causal directivity. At this point a technique capable of measuring the constructs and measurement objects was required.

“Measurement instruments that are collections of items combined into a composite score, and intended to reveal levels of theoretical variables not readily observable by direct means, are often referred to as scales. We develop scales when we want to measure phenomena that we believe exist because of our theoretical understanding of the world, but that we cannot assess directly. (DeVellis, 2003, p. 9)

The development of the framework structure proceeded to the next step, scale selection.

Scale Selection for Use in the Framework

A scale is defined as “. . . a theoretical variable in a model, and *scaling* or measurement is the attachment to empirical events of values of the variable in a model.” (Cliff, 1993, p. 89) Because measurement “. . . concerns the assignment of numbers to objects to represent amounts or degrees of a property possessed by all of the objects,” (Torgerson, 1958, p. 19) the type of scale was an important element. “The type of scale achieved depends upon the character of the basic empirical operations performed. These operations are limited ordinarily by the nature of the thing being scaled and by our choice of procedures.” (Stevens, 1946, p. 677) Uni-dimensional scale selection falls into one or another of the four scale types (Torgerson, 1958) shown in Figure 39. The axiomatic basis for each scale type has been stated by Coombs, Raiffa, & Thrall (1954). Because the measurement objects in this study had no natural origin or empirically defined distance, the ordinal scale was selected for all scales used in the framework.

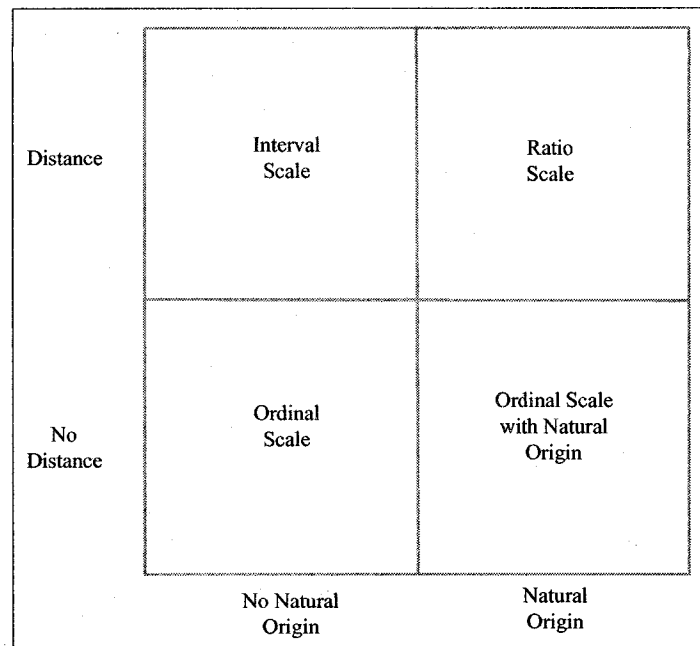


Figure 39: The Four Types of Scales

The measurement concept underlying ordinal scales is based on *order* and does not assume that the scores and other measures have the status of equal-interval scales. The use of ordinal scales enabled the researcher to apply a rationally conservative approach to the measurement attributes, ensuring that they did not act as though they had properties that they did not have. The decision to use ordinal scales was made based on the fact that most measurements in social and behavioral science are ordinal (Cliff & Keats, 2003). Ordinal scales are those “. . . in which (1) a set of objects is ordered from most to least with respect to an attribute, (2) there is no indication of how much in an absolute sense any of the objects possess the attribute, and (3) there is no indication of how far apart the objects are with respect to the attribute.” (Nunnally, 1967, p. 12) The numbers attached to the ordinal scale only provide a shorthand notation for designating the relative positions of the objects in the scale. For instance, the scale in Table 51 provides a rank-order that makes the following statement:

Measure	Descriptor	Attribute Description
S ₁	Very Low	Measure criteria 1.
S ₂	Low	Measure criteria 2.
S ₃	Nominal	Measure criteria 3.
S ₄	High	Measure criteria 4.
S ₅	Very High	Measure criteria 5

Table 51: Sample Ordinal Scale

$$S_1 < S_2 < S_3 < S_4 < S_5$$

This statement is the same as the accompanying descriptive words.

$$\text{Very Low} < \text{Low} < \text{Nominal} < \text{High} < \text{Very High}$$

The transitivity postulate holds and we are also able to state that $S_1 < S_5$ and so on. When this type of scale is used as the basis for a questionnaire it is referred to as “. . . multiple choice and polytomous scoring.” (Cliff & Keats, 2003, p. 44) The assignment of arbitrary integers for the scores is based on the work by Likert (1932), and later by Torgerson (1958). Likert showed that the results of using arbitrary integers “. . . gave a result that was highly correlated with that obtained by normalizing the distribution on each item.” (Cliff & Keats, 2003, p. 53) Torgerson (1958) referred to this method of arbitrary integral scoring as *measurement by fiat*. This method was adopted for simplicity and ease of implementation.

Two final points must be made about ordinal scales. The first point is the distinction between a *proposed* scale and a scale. “A proposed scale is one that some investigator(s) put forward as having the requisite properties, and if it is indeed shown to have them, then it is recognized as a scale.” (Cliff, 1993, p. 65). The use of the word scale in the dissertation is referring to *proposed scales*. The second point addresses the use of statistics. The use of ordinal scales limits the researcher to but a few statistics such as “. . . rank order coefficient of correlation, r , Kendall’s W , and rank order analysis of

variance, medians, and percentiles.” (Kerlinger & Lee, 2000, p. 636) The statistical limitation imposed by the use of ordinal scales was judged to be acceptable for this research.

Development of Scales for Use in the Framework

The procedure used to develop the scale is based on the theoretical hierarchy in Figure 38 that supports the framework:

1. Framework: Contains constructs
2. Construct: Contains measurement objects
3. Measurement Object: Contains attributes
4. Attributes: Contains criteria that can be measured

By using reflexive measures the researcher ensured that each element in the cascading criteria represented reflections of the unobservable constructs. Each measurement object had a one-to-one relation with the concept. The measurement object was supported by a number of process, factor, type, and function attributes that contained criteria that could be measured. Attribute scale construction addressed two details from the literature (Hinkin, 1995). The first detail addressed the number of items in a measure. These relations were concerned with preserving content and construct validity and were simply addressed. Each attribute contained one criterion to be measured. The second detail, the scaling of the item, addressed the reliability of the data under conditions when two or more comparable scores per attribute were present. The reliability can be measured using the Cronbach (1951) Coefficient alpha. Because Likert-type ordinal scales have been shown to have increased reliability (as measured by Coefficient alpha) up to the use of 5 points and “. . . a definite leveling off in the increase in reliability after 5 scale points,” (Lissitz & Green, 1975, p. 13) the 60 framework scales in this research were purposefully

designed for increased reliability. The completed framework contained 3 three-point and 57 five-point scales.

Once the high-level design of the framework, constructs, measures, and scales was completed the detailed constructs, measurement objects and supporting attributes were developed.

Measurement Objects, Attributes, and Criteria

The framework included three construct elements; (1) functions, (2) structure, and (3) environment. Each construct of the framework was based on the theoretical concepts presented in Figure 37 and Table 50. The construction and rationale for the supporting measurement objects, attributes, and criteria in each construct was based on the following:

Functions Construct

The functions construct of the framework was constructed from the theoretical concepts underlying the framework depicted in Figure 37. This included the six essential functions of a software development project arranged in five areas (development, improvement & training, infrastructure, life cycle support, and management) from ISO/IEC Standard 12007 (1995) and the 22 process areas from the CMMISM. The six major software development functions were aligned with the 22 CMMISM process areas, 2 new process areas, and 2 measurable lifecycle support factors in a hierarchical form. Figure 40 depicts the 24 detailed processes and 2 factors that make up the functions element of the framework.

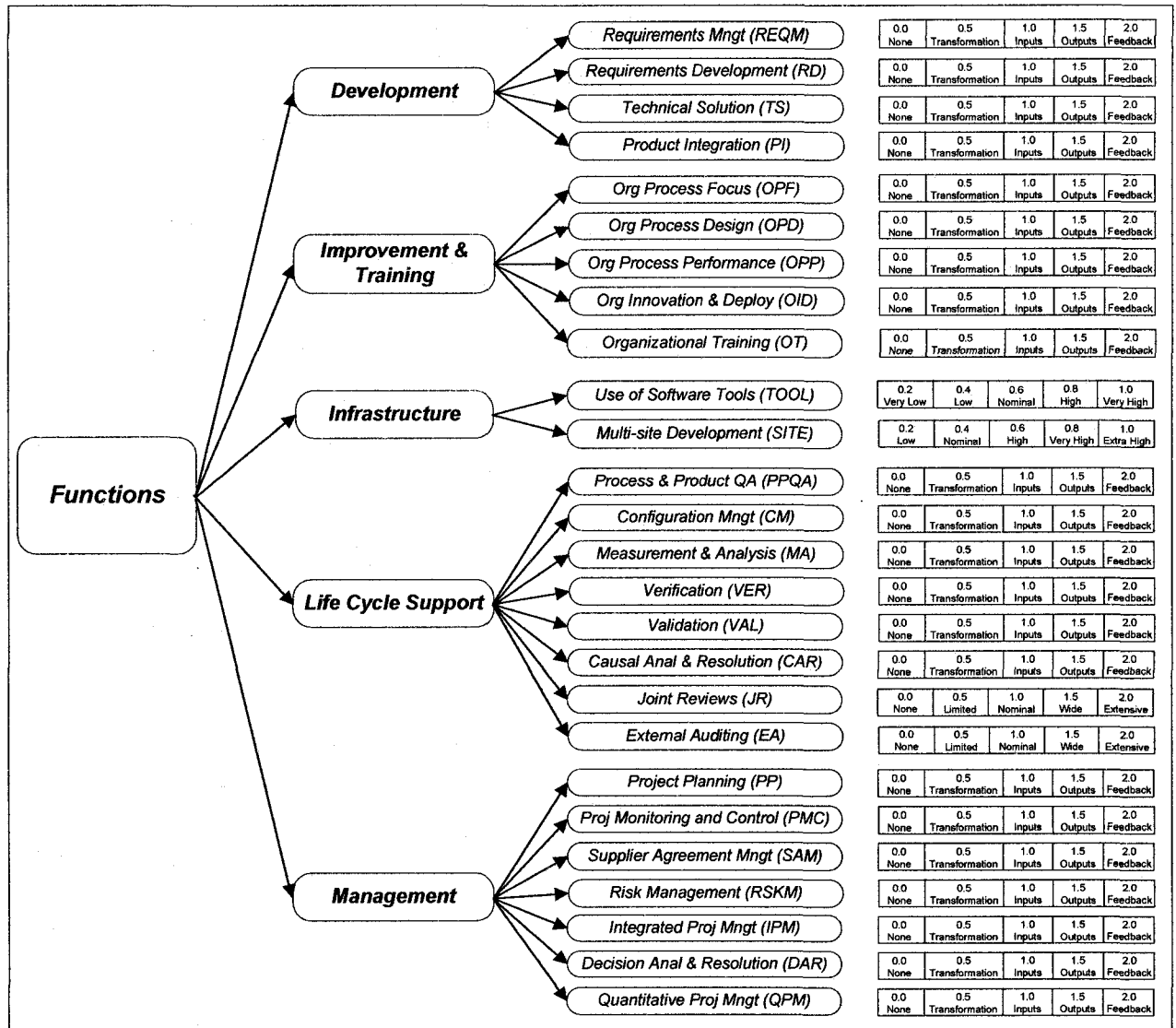


Figure 40: Functions Construct of the Framework

1. Process Areas.

22 of the measurable processes have been adopted from the CMMISM, 2 are from the COCOMO II, and 2 are new process areas suggested by ISO/IEC 12207. Table 52 lists the process/factor and the adoption source for the criteria. It is important to note that the framework utility was increased by using established criteria that are highly recognizable by all software professionals.

SW Development Project Essential Function	Measurable Process or Factor	Source of the Criteria
Development	REQM, RD, TS, PI	CMMI SM
Improvement & Training	OPF, OPD, OPP, OID, OT	CMMI SM
Infrastructure	TOOL, SITE	COCOMO II
Life Cycle Support	1. PPQA, CM, MA, VER, VAL, CAR 2. JR, EA	1. CMMI SM 2. Initial formal criteria for Joint Reviews (JR) and External Auditing (EA) are recommended.
Management	PP, PMC, SAM, RSKM, IPM, DAR, QPM	CMMI SM

Table 52: Functions Elements Criteria and Source

The significant change from the CMMISM is in how the criteria are measured.

Process measurement is approached from a systems viewpoint and uses elements from Banathy's Three Lens Model (Walton, 2004; Banathy, 1991) to analyze the process against four distinct criteria.

1. *Transformation.* How does the process engage in operations that attend to its purposes and transform the inputs into outputs?
2. *Input Processing.* How does the process receive, screen/assess, and process inputs?
3. *Output Processing.* How does the process assess and process the output?
4. *Feedback.* How does the process make adjustments and changes, and if needed, how does it transform itself based on information feedback coming to and from the system?

The four systems-based criteria are used to analyze and measure each process. Processes receive a progressive score based on a binary decision as to whether or not it has satisfied the measurement criteria and has objective quality evidence (OQE) to support the finding. The 24 processes use the measurement criteria in Table 53.

Measure	Descriptor	Measurement Criteria
0.0	None	The software development project does not perform the measured process as specified in the CMMI SM .
0.5	Transformation	The software development project satisfactorily performs the process as specified in the CMMI SM .
1.0	Inputs	The software development project satisfactorily (1) performs the process as specified in the CMMI SM , and (2) formally designates receives, screens, and processes inputs required for the transformation process.
1.5	Outputs	The software development project satisfactorily (1) performs the process as specified in the CMMI SM , (2) formally designates receives, screens, and processes inputs required for the transformation process, and (3) formally assesses and processes outputs from the transformation process.
2.0	Feedback	The software development project satisfactorily (1) performs the process as specified in the CMMI SM , (2) formally designates receives, screens, and processes inputs required for the transformation process, (3) formally assesses and processes outputs from the transformation process, and (4) includes a formal method for feedback that permits the process to make adjustments and changes, and if needed, transform itself based on information feedback coming to and from the transformation process.

Table 53: Process Measurement Criteria

The two new process factors in the functions element were required in the life cycle support functional area. The factors concern the Joint Review (JR) and External Audit (EA) processes. Neither process was formally addressed by the CMMISM, but included as formal processes in ISO/IEC 12207 (1995). The joint review process was addressed in sub-clause 6.6 of ISO/IEC 12207 (1995) and defined as “. . . a process for evaluating the status and products of an activity of a project as appropriate (ISO/IEC 12207, p. 39).” The audit process was addressed in sub-clause 6.7 and defined as “. . . a process for determining compliance with the requirements, plans, and contract as appropriate (ISO/IEC 12207, p. 40).” Furthermore, Keil & Robey (1999) discuss the use of independent external audits as a means for deescalating commitment in troubled projects. They go on to cite a study by Barton, Duchon & Dunegan (1989) that reports that an independent audit is more likely to produce objective evidence and to lead to the de-escalation of commitment to runaway projects. The research suggests that Joint

Review (JR) and External Audit (EA) processes would add value if included as process elements in the life cycle support functional area. The criteria for these processes are based on definitions in the respective sub-clauses of ISO/IEC 12207 (1995). The Joint Review (JR) process evaluation is based on the measurement criteria in Table 54.

Measure	Descriptor	Measurement Criteria
0.0	None	No joint reviews are conducted.
0.5	Limited	Only joint reviews required by contract are conducted.
1.0	Nominal	Joint reviews are used in a few process areas.
1.5	Wide	Joint reviews are used in most process areas.
2.0	Extensive	Joint reviews are present in all process areas.

Table 54: JR Measurement Criteria

The External Audit (EA) process evaluation is based on the measurement criteria in Table 55.

Measure	Descriptor	Measurement Criteria
0.0	None	No external audits are conducted.
0.5	Limited	Only external audits required by contract are conducted.
1.0	Nominal	External audits are used in a few process areas.
1.5	Wide	External audits are used in most process areas.
2.0	Extensive	External audits are present in all process areas.

Table 55: EA Measurement Criteria

2. Life Cycle Support Factors.

The two lifecycle support factors in the functions element address project infrastructure. Infrastructure is a core function in ISO/IEC 12207 that was not formally assessed by a process area in the CMMISM. The researcher adopted two established factors for infrastructure measurement from the Constructive Cost Model (COCOMO). COCOMO is a cost model for planning and executing software projects. It was developed by Barry Boehm (1981) and has been the most widely used and quoted predictive cost model for software development projects. It has been recently updated (Boehm et al, 2000) as COCOMO II. The first factor (TOOL) measures software tool capability and integration of the tool suite used in developing the software on the project. Tool capability and integration are not binary measures but have a range of values that

are functions of the level of adoption and integration of development tools on the project.

The life cycle factor for TOOL is based on the measurement criteria in Table 56.

Measure	Descriptor	Measurement Criteria
0.2	Very Low	The software development project tools are edit, code, debug with no integration.
0.4	Low	The software development project tools are simple, front end, backend CASE, with little integration.
0.6	Nominal	The software development project tools are basic life-cycle tools, with moderate integration.
0.8	High	The software development project tools are strong, mature life-cycle tools with moderate integration.
1.0	Very High	The software development project tools are strong, mature, proactive life-cycle tools that are well integrated with processes, methods and reuse.

Table 56: TOOL Measurement Criteria

The second factor (SITE) measures development team co-location and communications support for the software development team. This factor was also adopted from COCOMO II and uses a range of values. The life cycle factor for SITE were based on the measurement criteria in Table 57.

Measure	Descriptor	Measurement Criteria
0.2	Low	The software development site is multi-city and multi-company and uses individual phone and facsimile for communications.
0.4	Nominal	The software development site is multi-city and multi-company and uses narrow band e-mail for communications.
0.6	High	The software development site is in the same city or metro area and uses wideband electronic communications.
0.8	Very High	The software development site is in the same building or complex and uses wideband electronic communications and occasional video conferencing.
1.0	Extra High	The software development site is fully collocated and uses interactive multi-media communications methods.

Table 57: SITE Measurement Criteria

In summary, the framework's functions construct models a software development project's ability to perform the six major software development functions. The measurement criteria from Figure 40 assign up to 2 points for each of the 24 process areas (48 points) and 1 point for each of the 2 lifecycle support factors (2 points). The functions construct contributes a maximum of 50 points to the overall framework score.

Structure Construct

The structure construct of the framework was constructed from the theoretical concepts underlying the framework depicted in Figure 37. This concept included a solution structure based upon the proven perspectives of a synthesis of systemic methodologies and systems-based models which included the following:

1. *Cybernetic Perspective.* The three homeostatic methods of controlling variety and the basic cybernetic principles that address control and communications.
2. *Socio-Technical Systems Perspective.* All human activity systems have both a technical element (techniques and knowledge) and a social element (people, beliefs, relationships, skills).
3. *Systems-Based Models and Methods.* Addresses the essential elements from the Viable Systems Model required to maintain system viability.

The essential points from the *Cybernetic* and *Socio-Technical Systems Perspectives* and *Systems-Based Models and Methods* were assembled in a form where they could be analyzed and measured. Figure 41 is the structure element of the framework.

1. Social System.

The social system perspective addressed human activity systems which have a social element made up of people (Shani & Sena, 1994). This perspective was supported by 6 measurable personnel factors. The personnel factors addressed the capability and experience of the software development project's team and not that of the individual. The 6 personnel factors directly addressed *team risk*. "Team risk refers to issues associated with the project team members that can increase the uncertainty of the project's outcome, such as team member turnover, insufficient knowledge among team members, cooperation, motivation, and team communication issues." (Wallace, Keil &

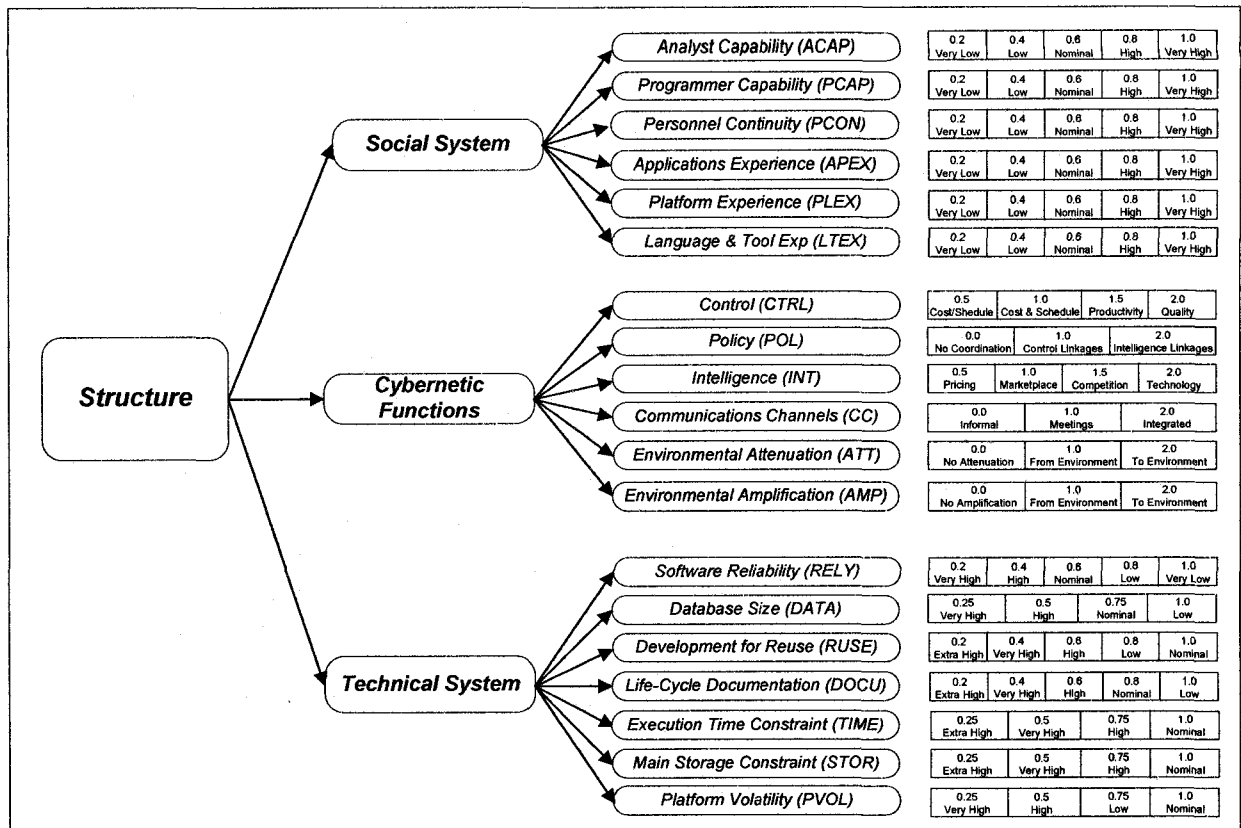


Figure 41: Structure Construct of the Framework

Rai, 2004, p. 293) Like TOOL and SITE, the 6 personnel factors have been adopted from the COCOMO II.

Analyst team and programmer team capabilities are the first two personnel measures. The major attributes considered in these factors were the team efficiency and thoroughness of the analyst team (ACAP) and programmer team (PCAP) and their ability to communicate and cooperate. Neither measure considered the experience of the analyst nor the programmer teams which were covered later (see APEX and LTEX). Both ACAP and PCAP measured the team's performance on a percentile scale, and is based on using the measurement criteria in Table 58.

Measure	Descriptor	Measurement Criteria
0.2	Very Low	15th-35th percentile.
0.4	Low	35th-55th percentile.
0.6	Nominal	55th-75th percentile.
0.8	High	75th-90th percentile.
1.0	Very High	>90th percentile.

Table 58: ACAP/PCAP Measurement Criteria

Personnel continuity (PCON) was the third personnel measure. This factor addressed the project's annual personnel turnover and was based on using the criteria in Table 59.

Measure	Descriptor	Measurement Criteria
0.2	Very Low	>48% per year
0.4	Low	24-47% per year
0.6	Nominal	13-23% per year
0.8	High	3-12% per year
1.0	Very High	<3% per year

Table 59: PCON Measurement Criteria

The final three personnel factors addressed the team's experience. Applications experience (APEX) measured the project team's equivalent level of experience using the specific type of application utilized in developing the software system. Platform experience (PLEX) measured the project team's experience using the specific type of platform (graphic user interface, database, networking, and distributed middleware) in developing the software system. Language and tool experience (LTEX) measured the project team's broad level of programming language and software tool experience in developing the software system. The APEX, PLEX, and LTEX factors were based on the criteria in Table 60.

Measure	Descriptor	Measurement Criteria
0.2	Very Low	<2 months.
0.4	Low	2-6 months.
0.6	Nominal	6-12 months.
0.8	High	1-6 years
1.0	Very High	>6 years

Table 60: APEX/PLEX/LTEX Measurement Criteria

2. Cybernetic Functions.

The cybernetic functions perspective addressed the cybernetic principles that regulate: (1) internal systems controls, (2) intelligence from outside of the system, (3) the coordination of policy, (4) communications channels, and (5) variety. The principles were supported by 5 measurable cybernetic factors.

The 1st cybernetic factor used cybernetic principles to address the systems elements for controlling internal system functions. System control was concerned with project actions that occurred *inside and now* (Espejo, 2004). The control structural filter screened near-term information from the internal environment of the project for use in policy making. In this case the operational managers were looking at the most commonly reported near-term control measures for cost, schedule, productivity, and quality of the software product (Jones, 2004). The criterion by which the control filter was evaluated escalates from simple cost and schedule controls to product quality. The cybernetic factor for control (CTRL) was based on the measurement criteria in Table 61.

Measure	Descriptor	Measurement Criteria
0.0	None	No performance controls.
0.5	Cost/Schedule	Cost or schedule data are used to control the performance of the software development project.
1.0	Cost & Schedule	Cost and Schedule data are used to control the performance of the software development project.
1.5	Productivity	Data on cost, schedule, and productivity data are used to control the performance of the software development project.
2.0	Quality	Data on cost, schedule, productivity and quality are used to control the performance of the software development project.

Table 61: CTRL Measurement Criteria

The 2nd cybernetic factor used cybernetic principles to address the systems elements for controlling external system intelligence. System intelligence was concerned with project actions that occur *outside and then* (Espejo, 2004). This filter screened far-term information from the external environment of the project for use in policy making.

In this case senior managers were looking at far-term performance measures for the marketplace (consumers), competition, emerging technologies, and pricing associated with the software product. The criterion through which the intelligence filter was evaluated escalates from simple pricing to the evaluation of emerging technologies. The cybernetic factor for intelligence (INT) was based on the measurement criteria in Table 62.

Measure	Descriptor	Measurement Criteria
0.0	None	No intelligence measures were used.
0.5	Pricing	Pricing data from consumer surveys and the competition is provided to intelligence functions of the project planning process (PP).
1.0	Marketplace	Pricing data from consumer surveys and the competition and consumer data from the marketplace are provided to intelligence functions of the project planning process (PP).
1.5	Competition	Pricing data from consumer surveys and the competition, consumer data from the marketplace, and marketplace data on the competition is provided to intelligence functions of the project planning process (PP).
2.0	Technologies	Pricing data from consumer surveys and the competition, consumer data from the marketplace, marketplace data on the competition, and information about emerging technologies are provided to intelligence functions of the project planning process (PP).

Table 62: INT Measurement Criteria

The 3rd cybernetic factor used cybernetic principles to address the systems elements for coordinating policy. This filter coordinated actions between the control and intelligence filters and created policy as an outcome (Espejo, 2004). In order to maintain viability a system must balance the need to take action on real-time systems requirements with far-term environmental requirements. The policy coordinating filter contains linkages between the control and intelligence elements designed to balance the needs of near-term control with those of far-term intelligence. The information provided from the policy element are used to define, adjust, and implement policies that positively affect the

software development project. The cybernetic factor for policy (POL) was based on the measurement criteria in Table 63.

Measure	Descriptor	Measurement Criteria
0.0	No coordination	No coordination between the intelligence or policy functions exists.
1.0	Control linkages	There is a formal linkage between the operational controls in the project measurement and control process (PMC) and the policy functions in the integrated project management (IPM) process.
2.0	Intelligence linkages	There is a formal linkage between the intelligence measures in the project planning process (PP) and the policy functions in the integrated project management (IPM) process.

Table 63: POL Measurement Criteria

The 4th cybernetic factor used cybernetic principles to address the systems elements for project communications. Communications were concerned with the channels through which information was shared within the project. The cybernetic factor for communications channels (CC) was based on the measurement criteria in Table 64.

Measure	Descriptor	Measurement Criteria
0.0	Informal	Informal communications channels
1.0	Meetings	Communications channels rely on meetings and reports
2.0	Integrated	Communications channels are formalized and included within the processes

Table 64: CC Measurement Criteria

The 5th cybernetic factor used cybernetic principles to control variety by addressing the systems elements for environmental attenuation and amplification. Environmental attenuation addressed the methods by which unwanted or unnecessary information and/or materials from the external environment are reduced. Environmental amplification addressed the methods by which necessary information and/or materials from the environment and/or the project are amplified. The cybernetic factor for environmental attenuation and amplification (ATT) was based on the measurement criteria in Table 65.

Measure	Descriptor	Measurement Criteria
0.0	None	No attenuation or amplifications methods are in place.
0.5	Informal from Environment	Informal methods for attenuating or amplifying information from the environment are in place.
1.0	Informal to/from Environment	Informal methods for attenuating or amplifying unwanted information to and from the environment are in place.
1.5	Formal from environment	Formal methods for attenuating or amplifying unwanted information from the environment are in place
2.0	Formal to/from Environment	Formal methods for attenuating or amplifying unwanted information to and from the environment are in place.

Table 65: ATT Measurement Criteria

3. Technical System.

The technical system perspective addressed human activity systems which have a technical element that include techniques and knowledge; commonly referred to as technical design (Shani & Sena, 1994). This perspective was supported by 9 measurable technical design factors. The technical design factors addressed the system that the software development team was tasked with developing. This included technical issues that affected system hardware, software, and documentation. 7 of the 9 factors were adopted from the COCOMO II.

The 1st technical design factor addressed software reliability (RELY). This was a measure of the extent to which the software must perform its intended function over a period of time. The technical design of the system ensured that in the event of complete system failure specific outcomes would occur. The criterion through which software reliability was evaluated escalates from a slight inconvenience to a major catastrophe. The technical design factor for software reliability (RELY) was based on the measurement criteria in Table 66.

Measure	Descriptor	Measurement Criteria
0.2	Very High	Catastrophic, total financial loss and risk to human life
0.4	High	Disastrous, high financial loss
0.6	Nominal	Moderate inconvenience, easily recoverable failures
0.8	Low	Low inconvenience, easily recoverable losses
1.0	Very Low	Slight inconvenience

Table 66: RELY Measurement Criteria

The 2nd technical design factor addressed database size (DATA). This was a measure of the extent to which large test data requirements affect product development. The factor was a measure of the ratio of bytes in the test database (D) to the number of source lines of code (SLOC) in the application program (P). The criteria through which database size was evaluated ranged from a very high ratio of D/P to a low ratio, which is a function of the system technical design. The technical design factor for database size (DATA) was based on the measurement criteria in Table 67.

Measure	Descriptor	Measurement Criteria
0.2	None	Not measured
0.4	Very High	$D/P > 1000$
0.6	High	$100 < D/P < 1000$
0.8	Nominal	$10 < D/P < 100$
1.0	Low	$D/P < 10$

Table 67: DATA Measurement Criteria

The 3rd technical design factor addressed the additional effort required to construct software components intended for reuse on this or future software development projects. The decision to reuse software components was characterized by how large the reuse effort was. The technical design factor for software component reuse (RUSE) was based on the measurement criteria in Table 68.

Measure	Descriptor	Measurement Criteria
0.2	Extra High	Across multiple product lines.
0.4	Very High	Across this software product line.
0.6	High	Across more than one software projects.
0.8	Low	On this software project.
1.0	Nominal	Reuse is not an element of the design

Table 68: RUSE Measurement Criteria

The 4th technical design factor addressed technical documentation. This was a measure of the affect of the technical design on life-cycle documentation requirements and was characterized by how the actual design affects the need for life-cycle documentation. The technical design factor for documentation (DOCU) was based on the measurement criteria in Table 69.

Measure	Descriptor	Measurement Criteria
0.2	Extra High	Design has created highly excessive life cycle needs.
0.4	Very High	Design has created excessive life-cycle needs.
0.6	High	Design has created many life-cycle needs.
0.8	Nominal	Design has created some life-cycle needs.
1.0	Low	Design is right-sized to life-cycle needs

Table 69: DOCU Measurement Criteria

The 5th technical design factor addressed system execution time. Execution time referred to how much of the customer specified system response time was purposefully designed to be used by the software system in executing the assigned tasks. This was a measure of the affect of the technical design on response time and was characterized by the percentage of the customer specified response time used in executing responses by the system. The technical design factor for execution time (TIME) was based on the measurement criteria in Table 70.

Measure	Descriptor	Measurement Criteria
0.2	Unmet	>100%
0.4	Extra High	90-95%
0.6	Very High	70-90%
0.8	High	50-70%
1.0	Nominal	< 50%

Table 70: TIME Measurement Criteria

The 6th technical design factor addressed the main storage constraint for the software system. Storage referred to how much of the customer specified main storage requirement was purposefully designed to be used by the software system. This was a measure of the affect of the technical design on storage capacity and was characterized by the percentage of the customer specified storage used in the software system design. The

technical design factor for storage (STOR) was based on the measurement criteria in Table 71.

Measure	Descriptor	Measurement Criteria
0.2	Unmet	>100%
0.4	Extra High	>95%
0.6	Very High	70-95%
0.8	High	50-70%
1.0	Nominal	< 50%

Table 71: STOR Measurement Criteria

The 7th technical design factor addressed the volatility of the platform for the system. Platform volatility referred to the design of the complex of hardware and software that made up the system and how often major changes could be expected to be required in the hardware and/or software. Platform volatility was characterized by how often major system changes must occur and was a function of the technical design. The technical design factor for platform volatility (PVOL) was based on the measurement criteria in Table 72.

Measure	Descriptor	Measurement Criteria
0.2	None	No change/update plan
0.4	Very High	Major changes every 2 weeks, minor changes every 2 days.
0.6	High	Major change every 2 months, minor changes every 1 week.
0.8	Low	Major changes every 6 months, minor changes every 2 weeks.
1.0	Nominal	Major changes every 12 months, minor changes every 1 month.

Table 72: PVOL Measurement Criteria

The two final technical design factors came from the systems and project management literature and addressed the complexity and technological certainty associated with the software development project.

The 8th technical design factor addressed the complexity of the product; the software system under development. “Characteristic of almost all good designs, in whatever sphere of activity they are produced, is a basic simplicity. A good design meets

its objectives and has no additional embellishments that detract from its main purpose.” (Budgen, 2003, p. 75) The converse of simplicity is complexity. The traditional measures for software complexity analyze architectural design (McCabe & Butler, 1994), program paths (McCabe, 1976), and object design (Chidamber & Kemerer, 1994) to predict testability, maintainability, and reliability. The measures are highly analytic, using sophisticated mathematical techniques to review design structure, code nesting and paths in an attempt to understand the effect of the relationships in the hierarchical structure. The software is purposefully designed and constructed in a hierarchical structure that has discrete but interacting levels. We can distinguish between the interactions *among* the systems elements and *within* the systems elements. Simon (1962) would designate a software system as *nearly decomposable*. A *nearly decomposable* system conforms to the following propositions (Simon, 1962, p. 474):

(a) In a nearly decomposable system, the short-run behavior of each of the component subsystems is approximately independent on the of the short-run behavior of the other components;

(b) In the long run, the behavior of any of the components depends in only an aggregate way on the behavior of the other components.

“The intra-component linkages are generally stronger than inter-component linkages.”

(Simon, 1962, p. 477) “Hierarchy and emergence contribute to complexity because new and interesting properties that cannot be found in the parts emerge and add a whole new dimension to understanding.” (Flood & Carson, 1993, p. 31)

The project management literature provides a less analytic but meaningful method to predict complexity basic on the principle of hierarchy. “The notion of different hierarchies inside a product or system, with different design and managerial implications is a dimension for distinction among projects.” (Shenhar & Dvir, 1996, p. 611) The

overall measure of a software system's complexity (COMP), which captured size, the number of system elements, variety, and interconnectedness was based on the criteria presented in Table 73.

Measure	Descriptor	Measurement Criteria
0.2	Super-System	A collection of independent systems.
0.4	Array	Widely dispersed collection of subsystems with a common mission.
0.6	System	Collection of subsystems with multiple functions.
0.8	Subsystem	Module performing multiple functions in a single functional area.
1.0	Program	Program performing a single function.

Table 73: COMP Measurement Criteria

The 9th technical design factor addressed the technological certainty of the product; the software system under development. While there is no established measure in the software industry the project management literature provides the theoretical basis for the use of technological uncertainty as a valid measure. (Shenhar & Dvir, 1996) Charette (1989) addresses the application of technology on software development projects and rates the technology, in order of increasing risk, as “. . . obsolescent, standard practice, best practicable, best available, latest.” (Charette, 1989, p. 76) Shenhar et al (2005) evaluate the extent to which new technology is used on a project as “. . . low-tech, medium-tech, high-tech, and super high-tech.” (Shenhar et al, 2005, p. 9) The technical design factor which evaluates a software project's adoption of new technology (TECH), was based on the criteria presented in Table 74

Measure	Descriptor	Measurement Criteria
0.2	Super High-Tech	Necessary technologies do not exist at project initiation.
0.4	Emerging Tech	Technology is in development but not yet released.
0.6	High-Tech	All or mostly new, but existing technologies.
0.8	Medium-Tech	Some new technology.
1.0	Low-Tech	No new technology is used.

Table 74: TECH Measurement Criteria

The measurement criteria from Figure 41 assigned up to 2 points for all 5 of the cybernetic functions (10 points) and 1 point for each of the 15 remaining social system and technical system factors (15 points). The structure construct contributed a maximum of 25 points to the overall framework score.

Environment Construct

The environment construct of the framework was constructed from the theoretical concepts underlying the framework depicted in Figure 37. This construct included the definition of the systems contextual environment which “. . . takes into account the particular entities, trends, patterns, stakeholders, and constraints external to the problem system.” (Keating, Kauffmann, & Dyer, 2001, p. 776) The contextual environment for a software development project included the following:

1. *External Controls.* Software development projects are subject to external controls which include laws, regulations and standards.
2. *Resources.* Software development project resources are provided from the external environment and include manpower, material, money, schedule, methods, and information.
3. *Stakeholders.* Software development projects are controlled and influenced by interested stakeholders which include owners/shareholders, external management, customers, suppliers, and users.

The essential points from these areas were assembled in a form where they could be analyzed and measured. Figure 42 is the environment construct of the framework.

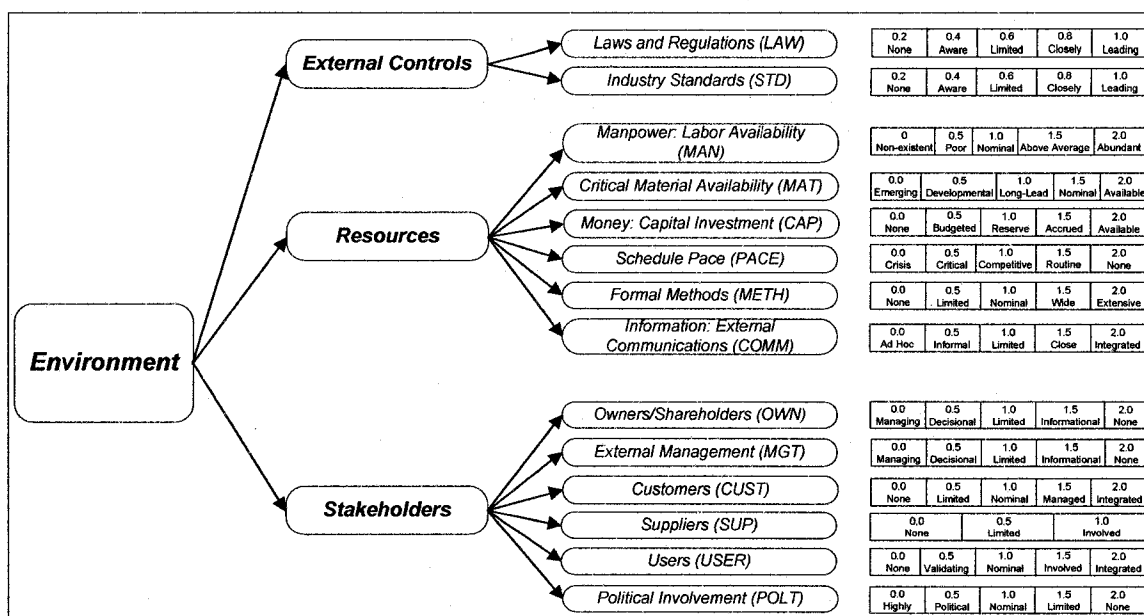


Figure 42: Environment Construct of the Framework

1. External Controls.

The external controls that affected software development projects appeared in two forms; government laws and regulations and industry standards. The 1st factor addressed government laws and regulations and measured the extent to which government laws and regulations were addressed by the project and/or parent organization. The involvement of the project and/or its parent organization can mitigate any negative effects of laws and regulations or help provide helpful guidance to lawmakers. The factor for laws and regulations (LAW) was based on the measurement criteria in Table 75.

Measure	Descriptor	Measurement Criteria
0.2	None	The project and/or the parent organization are not involved in influencing laws and regulations.
0.4	Aware	The project and/or the parent organization belong to a group that is involved in influencing laws and regulations.
0.6	Limited	The project and/or the parent organization have limited involvement in influencing laws and regulations.
0.8	Close	The project and/or the parent organization are closely involved in influencing laws and regulations.
1.0	Leading	The project and/or the parent organization are leaders in influencing laws and regulations.

Table 75: LAW Measurement Criteria

The 2nd factor addressed industry standards and measured the extent to which industry standards were addressed by the project and/or parent organization. The involvement of the project and/or its parent organization can mitigate any negative effects of industry standards or provide helpful guidance to standards committees and professional organizations that help enact standards. The factor for industry standards (STD) was based on the measurement criteria in Table 76.

Measure	Descriptor	Measurement Criteria
0.2	None	The project and/or the parent organization are not involved in influencing industry standards.
0.4	Aware	The project and/or the parent organization belong to a group that is involved in influencing industry standards.
0.6	Limited	The project and/or the parent organization have limited involvement in influencing industry standards.
0.8	Close	The project and/or the parent organization are closely involved in influencing industry standards.
1.0	Leading	The project and/or the parent organization are leaders in influencing industry standards.

Table 76: STD Measurement Criteria

2. Resources.

Software development project resources were provided from the external environment and included manpower, material, money, schedule, methods, and information. The 1st factor addressed labor availability. This was a measure of the labor availability for the skills required by the project. The availability of specific labor skills was a factor that has far reaching affects on projects that range from the cost for labor to the skill levels of project personnel. The factor for labor availability (MAN) was based on the measurement criteria in Table 77.

Measure	Descriptor	Measurement Criteria
0.0	Non-existent	None of the required labor skills are present in the local workforce requiring the project to import all labor skills or adopt remote development.
0.5	Poor	There are significant gaps in the required labor skills present in the local workforce requiring the project to import significant labor skills and use local and remote development.
1.0	Nominal	Some of the required labor skills are present in the local workforce. Key labor skills are required to be imported. The project is a hybrid of local and remote development.
1.5	Above Average	Most of the required labor skills are present in the local workforce. Only a few key labor skills are required to be imported. The project may be developed locally and use some limited remote development.
2.0	Abundant	All required labor skills are present in the local workforce permitting the project to do all development locally.

Table 77: MAN Measurement Criteria

The 2nd factor addressed the availability of critical material. This was a measure of the availability of critical materials (typically hardware and software components) required by the project. The timely availability of critical materials is a factor that can affect project cost and schedule. The factor for critical material availability (MAT) was based on the measurement criteria in Table 78.

Measure	Descriptor	Measurement Criteria
0.0	Emerging	Critical materials are dependent upon an emerging technology and do not exist at project initiation.
0.5	Developmental	Critical materials are in development but have not been released at project initiation.
1.0	Long-Lead	Critical materials are available but have a long lead time for delivery.
1.5	Nominal	Critical materials are available but have a well-defined delivery period.
2.0	Available	Critical materials are readily available.

Table 78: MAT Measurement Criteria

The 3rd factor addressed the availability of capital. This was a measure of the availability of capital required to fund indirect activities (i.e. process improvement, skill development, training, etc.) on the project. The availability of capital is a factor that can affect project readiness and the ability to adopt and implement improvement initiatives. The factor for capital (CAP) was based on the measurement criteria in Table 79.

Measure	Descriptor	Measurement Criteria
0.0	None	There are no capital funds available for the project.
0.5	Budgeted	Capital funds for the project are included in the project budget.
1.0	Reserve	Capital funds for the project must come from the project management reserve.
1.5	Accrued	Capital funds for the project come from project generated profits as they are accrued.
2.0	Available	Capital funds are readily available.

Table 79: CAP Measurement Criteria

The 4th factor addressed the project's schedule pace. This was a measure of the importance of achieving the schedule for the project. The pace of the project can affect a wide variety of decisions from project readiness to the ability to adopt and implement improvement recommendations. The factor for schedule pace (PACE) was based on the measurement criteria in Table 80.

Measure	Descriptor	Measurement Criteria
0.0	Crisis	Immediate delivery of the software system is necessary.
0.5	Critical	Completion time is crucial for success-window of opportunity.
1.0	Competitive	Time to market is important for the business.
1.5	Routine	Based on well-developed release schedule.
2.0	None	Not critical.

Table 80: PACE Measurement Criteria

The 5th factor addressed the methods used on the project. This was a measure of the adoption and implementation of formal methods on the project. The adoption and use of formal methods is indicative of the project's willingness and ability to adopt repeatable processes and activities. The factor for project methods (METH) was based on the measurement criteria in Table 81.

Measure	Descriptor	Measurement Criteria
0.0	None	No formal standards are used.
0.5	Limited	Formal standards are those required by contract.
1.0	Nominal	Formal standards are used in a few process areas.
1.5	Wide	Formal standards are used in most process areas.
2.0	Extensive	Formal standards are present in all process areas.

Table 81: METH Measurement Criteria

The 6th factor addressed information controls used on the project. This was a measure of how information between the project and the external environment was controlled. The adoption and use of formal information controls is indicative of the project's understanding with the respect to timely dissemination and flow of information. The factor for project information controls (COMM) was based on the measurement criteria in Table 82.

Measure	Descriptor	Measurement Criteria
0.0	Ad hoc	No information controls exist.
0.5	Informal	Informal information controls exist
1.0	Limited	Information controls are limited to formal correspondence.
1.5	Close	Information is closely monitored and dissemination controls are strictly enforced.
2.0	Integrated	Information controls are formalized and included within the processes.

Table 82: COMM Measurement Criteria

3. Stakeholders.

Software development projects are controlled and influenced by interested stakeholders. Stakeholders include owners/shareholder boards, external management, customers, suppliers, and users. Research on the affects of stakeholders has shown that they are an important element in project management and should be addressed by formal processes (Karlsen, 2002). The ability to understand and address the interests of the stakeholder groups decreases both organizational risk and user risk (Wallace, Keil & Rai, 2004). Five unique factors addressed the project's relationship with each stakeholder group.

The 1st factor addressed the owners and shareholder boards. This was a measure of the involvement of owners or shareholder boards on the project. The factor for owner/shareholder involvement (OWN) was based on the measurement criteria in Table 83.

Measure	Descriptor	Measurement Criteria
0.0	Managing	Owners or shareholder boards make day-to-day decisions on the project.
0.5	Decisional	Owners or shareholder boards are involved in making decisions about project-level activities.
1.0	Informational	Owners or shareholder boards are informed about overall project performance (i.e. cost, schedule and customer satisfaction).
1.5	Limited	Owners or shareholder boards have limited knowledge about the project (i.e. revenue contribution and profit/loss).
2.0	None	Owners or shareholders boards have no involvement with the project.

Table 83: OWN Measurement Criteria

The 2nd factor addressed external management. This was a measure of the involvement of management external to the project. Increasing involvement of external management receives lower scores because “. . . the literature identifies top management support as a factor contributing to the escalation of commitment to projects that are failing.” (Keil & Robey, 1999, p. 67) “An IT executive’s job is to keep an eye on the market and competitors. It is no longer essential to know *how*, but rather to know *why*.” (Karlsen, Gottshalk, & Andersen, 2002, p. 11) The factor for external management involvement (MGT) was based on the measurement criteria in Table 84.

Measure	Descriptor	Measurement Criteria
0.0	Managing	External management make day-to-day decisions on the project.
0.5	Decisional	External management is involved in making decisions about project-level activities.
1.0	Informational	External management is informed about overall project performance (i.e. cost, schedule and customer satisfaction).
1.5	Limited	External management has limited knowledge about the project (i.e. revenue contribution and profit/loss).
2.0	None	External management has no involvement with the project.

Table 84: MGT Measurement Criteria

The 3rd factor addressed the customer. This was a measure of the level of involvement of the customer on the project. The customer was defined as “. . . the company, organization, or person who is paying for the software system to be developed,” (Pfleeger, 1998, p. 14) and as such was differentiated from users. Increased

involvement of customers received a higher score because the literature showed that “. . . where regular project reviews and evaluations were conducted, de-escalation of commitment to failing projects was likely to be encouraged.” (Keil & Robey, 1999, p. 11)

The factor for customer involvement (CUST) was based on the measurement criteria in Table 85.

Measure	Descriptor	Measurement Criteria
0.0	None	No customer reviews or evaluations are conducted.
0.5	Limited	Customer reviews and evaluations occur only when the cost exceeds budget or the schedule slips.
1.0	Nominal	Periodic cost and schedule reports are sent to the customer.
1.5	Managed	The customer requires periodic face-to-face reviews of project progress and costs.
2.0	Integrated	Periodic reviews and evaluations with the customer are formalized and included within the project's processes.

Table 85: CUST Measurement Criteria

The 4th factor addressed the project's suppliers. This was a measure of the involvement of suppliers with the project. Suppliers are those that provided products and/or services for the project. This included subcontractors that performed any of the 24 software functional processes. The wide use of subcontractors to provide process support in highly specialized areas or when organic project resources are limited or unavailable may affect project success. Therefore, the involvement of suppliers in cost and schedule received an increasing score. The factor for supplier involvement (SUP) was based on the measurement criteria in Table 86.

Measure	Descriptor	Measurement Criteria
0.0	None	The supplier has no knowledge as to his involvement with the project (i.e. fee and delivery date).
0.5	Limited	The supplier's involvement is constrained to his product or service area on the project (i.e. limited project cost and schedule visibility).
1.0	Involved	The supplier's involvement is not constrained in any way (i.e. has complete cost and schedule visibility).

Table 86: SUP Measurement Criteria

The 5th factor addressed the software system's future users. "It is almost an *axiom* (my italics) of the MIS literature that user involvement is a necessary condition for successful development of computer based information systems." (Ives & Olsen, 1984, p. 586) The early wisdom stated the need to involve users in the various stages of the software development process (De Brabander & Edström, 1977; Markus & Robey, 1983) and was based on a few non-empirical studies that demonstrated a positive correlation between user involvement and various measures of system success (Gallagher, 1974; King & Rodriguez, 1981; Robey & Farrow, 1982). Further examination requires three definitions:

1. "User involvement refers to the participation in the system development process by representatives of the target user group" (Ives & Olsen, 1984, p. 587)
2. A user is defined as "... the person or people who will actually use the system: the ones who sit at the terminal or submit the data or read the output." (Pfleeger, 1998, p. 14)
3. "The degree of involvement refers to the amount of influence the user has over the final product." (Ives & Olson, 1984, p. 590)

The relationship between the degree of involvement of users in the development process and system success was the focus of three empirical studies (Baroudi, Olson & Ives, 1986; Tait & Vessey, 1988; McKeen & Guimaraes, 1997). The three studies validated the wisdom-based correlation between user involvement and systems success providing support for the use of a factor to measure the degree of involvement of users with the product being developed by the project.

Increasing levels of user involvement received higher scores because “. . . the lack of user involvement during system development is one of the most often cited risk factors in the literature.” (Keil & Robey, 1999, p. 67) The measurement criteria was based on the categories for user involvement cited by Ives and Olson (1984) and were measured by the criteria in Table 87.

Measure	Descriptor	Measurement Criteria
0.0	None	Users are unwilling or not invited to participate.
0.5	Validating	Users were involved in formal user acceptance testing.
1.0	Nominal	Users were involved in the development of requirements and user acceptance testing.
1.5	Involved	Users were involved in the development of requirements, validation of design, and user acceptance testing.
2.0	Integrated	Users' roles were formalized and included within the all development processes.

Table 87: USER Measurement Criteria

The measurement criteria from Figure 42 assigned up to 1 point for the 2 external control functions (2 points), 2 points for each of the 6 resource functions (12 points), 2 points for 5 of the stakeholder functions and 1 point for the supplier function (11 points). The environment construct contributed a maximum of 25 points to the overall framework score.

Verification of the Framework

A formal review of the completed framework was conducted prior to the formal validation using the two real-world case studies. The review examined the extent to which the framework looked like it measured what it was intended to measure (Nunnally, 1967). This was accomplished using a panel of three systems engineering experts who verified the framework using the guidelines contained in Appendix C. The formal comments on the framework criteria provided by the panel members were provided in Appendix E. The comments were generally positive and verified that the FSE Framework had the proper boundaries, was utilitarian, and pragmatic. By occurring prior

to the formal validation of the framework it allowed the researcher to incorporate the comments of experts prior to the case study validation.

One of the comments from the panel recommended the inclusion of a factor to address political events and the role of politics in software development projects. Additional research on this topic sustained the comment of the panelist and resulted in the inclusion of a new factor to address project politics. Politics (POLT) was added to the stakeholder measurement object in the environmental construct of the framework as a 6th factor. The particulars surrounding the construct of the POLT factor were as follows.

The project management (Pinto, 2000), management (Mayes & Allen, 1977; Tushman, 1977; Farrell & Petersen, 1982), and software development (Robey & Markus, 1984; Franz & Robey, 1984; Robey, Smith & Vijayasarathy, 1993; Sillince & Mouakket, 1997) literature discussed the patterns and effects of political behavior in organizations. While each study focused on a different aspect of political behavior they all addressed the negative effect of politics on performance. The key points were:

- **Software Development.** The nature of the software development process creates a high potential for conflict. Much of the inherent risk for conflict is caused by a development process that requires a diversity of skills in the presence of resource pressure and time constraints (Robey, Farrow & Franz, 1989).
- **Project Level Management.** Project managers do not have a stable base of organizational power (i.e. projects operate outside of the organizational matrix and draw resources from within the matrix) and are not routinely given the authority to issue performance evaluations on subordinates. Because of these

structural constraints project managers tend to use politics as a method to secure resources, control subordinates, and influence external decision makers (Pinto, 2000).

- **Organizational Leadership.** Productive, effective organizations require the presence of a leader willing to exercise power or to delegate sufficient authority to subordinates. In the absence of exercised power negative political behavior appears within the organization. Failure to exercise power and the presence of rising political activity causes the project to suffer both a loss of learning and the ability to reach agreement on essential issues (Levine & Rossmore, 1995).
- **Top Level Management.** Executives tend to develop stable political coalitions with one or possibly two other executives. Politics restricts information flow, is time-consuming, distractive, and dissipates the energy of executives. Firms with politically active top level management teams perform less well (Eisenhardt & Bourgeois, 1988).

The literature contained a number of definitions for workplace politics (Gandz & Murray, 1980) but the one that stood out as best capturing the meaning of politics common in organizations was offered by Eisenhardt & Bourgeois (1988).

Politics are the observable, but often covert, actions by which executives enhance their power to influence a decision. These actions include behind-the-scenes coalition formation, offline lobbying and cooptation attempts, withholding information, and controlling agendas. (pp. 737-738)

The definition was reinforced by the contrast made when compared to an *apolitical* environment in which the characteristics are (Eisenhardt & Bourgeois, 1988).

Straightforward influence tactics of open and forthright discussion, with full sharing of information, in settings open to all decision makers. (p. 738)

The *apolitical* actions can be viewed as legitimate while the political actions can be viewed as illegitimate. Farrell & Petersen (1982) have classified political behavior in organizations using three dimensions:

1. Internal-External: The political behavior focuses on resources within the organization or external to the organization.
2. Vertical-Lateral: The political behavior is directed up the organizational hierarchy or across the organizational hierarchy.
3. Legitimate-Illegitimate: defined by the political and apolitical definitions from Eisenhardt & Bourgeois (1988).

Figure 43 depicts some typical organizational behaviors using these dimensions.

Behavior is Focused on Resources	Internal to the Organization	<p><u>Legitimate</u></p> <ol style="list-style-type: none"> 1. Complain to supervisor 2. Bypassing the chain of command 3. Obstructionism <p><u>Illegitimate</u></p> <ol style="list-style-type: none"> 1. Sabotage 2. Symbolic protests 3. Mutinies 4. Riots 	<p><u>Legitimate</u></p> <ol style="list-style-type: none"> 1. Coalition forming 2. Exchanging favors <p><u>Illegitimate</u></p> <ol style="list-style-type: none"> 1. Threats 2. Reprisals
	External to the Organization	<p><u>Legitimate</u></p> <ol style="list-style-type: none"> 1. Lawsuit <p><u>Illegitimate</u></p> <ol style="list-style-type: none"> 1. Whistle blowing 2. Leaking information to the media or competition 3. Appeals to higher levels of management outside of the organization 	<p><u>Legitimate</u></p> <ol style="list-style-type: none"> 1. Talk with counterpart in another area of the project 2. Outside professional activity <p><u>Illegitimate</u></p> <ol style="list-style-type: none"> 1. Organizational duplicity 2. Defections
		Vertically	Laterally
		Behavior is Focused in Hierarchy	

Figure 43: Grid of Organizational Behavior

The factor for political involvement (POLT) was based upon the behaviors in Figure 43.

Decreased levels of political activity receive higher scores because the literature shows that "... firms with politically active top management teams perform less well."

(Eisenhardt & Bourgeois, 1988, p. 761) The factor for political involvement (POLT) was measured by the criteria in Table 88.

Measure	Descriptor	Measurement Criteria
0.0	Highly political	Political behaviors involve external management and parties' external to the project and/or parent organization.
0.5	Political	Political behaviors involve external management.
1.0	Nominal	Political behaviors occur up the project's management hierarchy.
1.5	Limited	Political behaviors occur across the project's management hierarchy
2.0	None	No political behaviors exist on the project.

Table 88: POLT Measurement Criteria

Deployment of the Framework

The completed Function-Structure-Environment or FSE Framework was now ready for deployment using the two validating case studies.

VALIDATION OF THE FRAMEWORK

The framework was validated using two real-world software development projects as case studies. The validation followed the steps in the Augmented Discoverers' Induction Procedure presented in Chapter 3 and presented in Table 39. This section will include a description of the logic used in evaluating each case using the FSE Framework. This is followed by a summary of the results from each case study and the validation data derived from the framework driven analysis.

Framework Evaluation

The narrative of the case study and the sources of the empirical data, for the Denver International Airport (DIA) Baggage Handling System (BHS) and the Federal

Bureau of Investigation (FBI) Virtual Case File (VCF) system are presented in Appendices F and G. The empirical evidence contained in the literature and interview questionnaires were deployed against the FSE Framework. The researcher reviewed the particulars contained in both the literature-based data and survey data for each of the 60 measurement objects in the framework. In order to arrive at a score the researcher had to determine how the case study data satisfied the measurement criteria established for each of the measurement objects. The logic used in the analysis is presented in Figure 44.

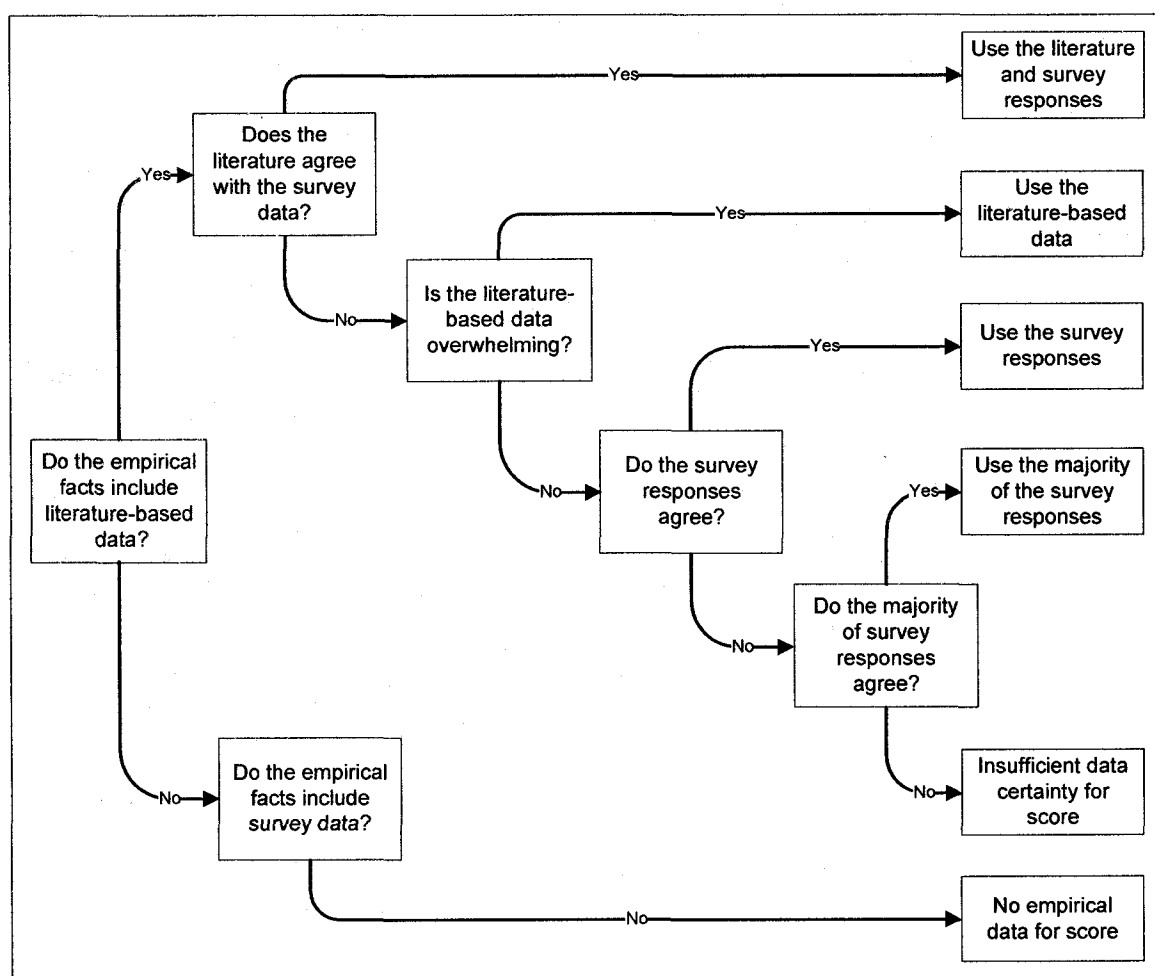


Figure 44: Decision Tree used Evaluate Empirical Data using the FSE Framework

Denver International Airport Baggage Handling System

Appendix F contains the detailed empirical evidence used for the evaluation and the tabulations against each of the FSE Framework's 60 measurement objects. The case study scores using the FSE Framework were as follows.

The functions element used 166 empirical data points to derive the scores against the FSE Framework. The project scored 9 of 50 total points in the functions element.

The functions element and associated area scores are shown in Table 89.

Area	Measurement Object	Data Points	Measure Score	Area Score
Development	REQM	10	0.5	3.0/8.0
	RD	17	0.5	
	TS	10	1.0	
	PI	10	1.0	
Improvement & Training	OPF	3	0.0	0.0/10.0
	OPD	4	0.0	
	OPP	4	0.0	
	OID	4	0.0	
	OT	3	0.0	
Infrastructure	TOOL	3	0.6	1.0/2.0
	SITE	3	0.4	
Life Cycle Support	PPQA	3	0.0	3.0/16.0
	CM	3	0.0	
	MA	4	0.0	
	VER	5	0.5	
	VAL	5	0.5	
	CAR	4	0.0	
	JR	5	1.0	
	EA	7	1.0	
Management	PP	16	0.5	2.0/14.0
	PMC	11	1.0	
	SAM	6	0.0	
	RSKM	8	0.0	
	IPM	8	0.5	
	DAR	7	0.0	
	QPM	3	0.0	
FUNCTION SCORE		166	9.0	9.0/50.0

Table 89: DIA BHS Function Element Scores

Four of the five areas contributed to the low score. The development area contributed 3 points but achieved only 37.5% of the maximum score. The improvement & training area contributed 0 points, which is a significant deficiency. The life cycle support area

contributed 3 points, 18.75% of the maximum score. Finally, the management area contributed 2 points, 14% of the maximum score.

The structure element used 126 empirical data points to derive the scores against the FSE Framework. The project scored 10.8 of 25 total points in the structure element. Two of the three areas contributed to the low score. The cybernetic functions area contributed 2 points but achieved only 20.0% of the maximum score. The technical system area contributed 4.8 points, which is 53% of the maximum score. The FSE Framework scores, by area and measurement object, are shown in Table 90.

Area	Measurement Object	Data Points	Measure Score	Area Score
Social System	ACAP	7	0.6	4.0/6.0
	PCAP	8	0.4	
	PCON	3	0.6	
	APEX	14	0.8	
	PLEX	5	0.8	
	LTEX	3	0.8	
Cybernetic Controls	CTRL	3	1.0	2.0/10.0
	POL	3	0.0	
	INT	4	0.0	
	CC	7	0.0	
	ATT	10	1.0	
Technical System	RELY	3	0.4	4.8/9.0
	DATA	3	1.0	
	RUSE	2	0.2	
	DOCU	2	0.4	
	TIME	2	0.6	
	STOR	2	0.6	
	PVOL	2	0.8	
	COMP	32	0.6	
	TECH	11	0.2	
STRUCTURE SCORE		126	10.8	10.8/25

Table 90: DIA BHS Structure Element Scores

The environment element used 93 empirical data points to derive the scores against the FSE Framework. The project scored 10.4 of 25 total points in the environment element. All three of the areas contributed less than 50% of the maximum scores. The external controls area contributed 0.4 points but achieved only 20.0% of the

maximum score. The resource area contributed 4.5 points, which is 37.5% of the maximum score. Finally, the stakeholder area contributed 5.5 points, 50% of the maximum score. The FSE Framework scores, by area and measurement object, are shown in Table 91.

Area	Measurement Object	Data Points	Measure Score	Area Score
External Controls	LAW	3	0.2	0.4/2.0
	STD	3	0.2	
Resources	MAN	6	1.0	4.5/12.0
	MAT	3	0.5	
	CAP	3	0.5	
	PACE	13	0.5	
	METH	3	1.0	
	COMM	5	1.0	
Stakeholders	OWN	14	1.0	5.5/11.0
	MGT	5	1.0	
	CUST	3	1.5	
	SUP	3	0.5	
	USER	6	1.5	
	POLT	23	0.0	
ENVIRONMENT SCORE		93	10.4	10.4/25

Table 91: DIA BHS Environment Element Scores

A complete analysis of this case has been provided in Chapter 6, Discussion of Results.

Federal Bureau of Investigation Virtual Case File System

Appendix G contains the detailed empirical evidence used for the evaluation and the tabulations against each of the FSE Framework's 60 measurement objects. The case study scores using the FSE Framework were as follows.

The functions element used 157 empirical data points to derive the scores against the FSE Framework. The project scored 7.5 of 50 total points in the functions element. The functions element and associated area scores are shown in Table 92. Four of the five areas contributed to the low score. The development area contributed 2 points but achieved only 25% of the maximum score. The improvement & training area contributed 0 points, which is a significant deficiency. The life cycle support area contributed 3.5

points, 21.87% of the maximum score. Finally, the management area contributed 1 point, 7% of the maximum score.

Area	Measurement Object	Data Points	Measure Score	Area Score
Development	REQM	2	0.5	2.0/8.0
	RD	27	0.5	
	TS	9	0.5	
	PI	6	0.5	
Improvement & Training	OPF	0	0.0	0.0/10.0
	OPD	0	0.0	
	OPP	0	0.0	
	OID	0	0.0	
	OT	0	0.0	
Infrastructure	TOOL	0	0.6	1.0/2.0
	SITE	1	0.4	
Life Cycle Support	PPQA	1	0.0	3.5/16.0
	CM	2	0.5	
	MA	4	0.0	
	VER	1	0.5	
	VAL	0	0.5	
	CAR	1	0.0	
	JR	6	1.0	
	EA	8	1.0	
Management	PP	15	0.5	1.0/14.0
	PMC	14	0.5	
	SAM	30	0.0	
	RSKM	3	0.0	
	IPM	25	0.0	
	DAR	1	0.0	
	QPM	1	0.0	
FUNCTION SCORE		157	7.5	7.5/50.0

Table 92: FBI VCF Function Element Scores

The structure element used 44 empirical data points to derive the scores against the FSE Framework. The project scored 10.2 of 25 total points in the structure element. Two of the three areas contributed to the low score. The cybernetic functions area contributed 1 point but achieved only 10% of the maximum score. The social system area contributed 2.8 points, which is 47% of the maximum score. The FSE Framework scores, by area and measurement object, are shown in Table 93.

Area	Measurement Object	Data Points	Measure Score	Area Score
Social System	ACAP	1	0.2	2.8/6.0
	PCAP	0	0.6	
	PCON	6	0.2	
	APEX	0	0.6	
	PLEX	0	0.6	
	LTEX	0	0.6	
Cybernetic Controls	CTRL	0	1.0	1.0/10.0
	POL	14	0.0	
	INT	0	0.0	
	CC	1	0.0	
	ATT	0	0.0	
Technical System	RELY	6	0.4	6.4/9.0
	DATA	0	0.8	
	RUSE	1	1.0	
	DOCU	2	0.8	
	TIME	1	0.4	
	STOR	0	1.0	
	PVOL	0	0.6	
	COMP	8	0.6	
	TECH	4	0.8	
STRUCTURE SCORE		44	10.2	10.2/25

Table 93: FBI VCF Structure Element Scores

The environment element used 38 empirical data points to derive the scores against the FSE Framework. The project scored 12.4 of 25 total points in the environment element. All three of the areas contributed less than 70% of the maximum scores. The external controls area contributed 1.4 points and achieved 70.0% of the maximum score. The resource area contributed 6.0 points, which is 50% of the maximum score. Finally, the stakeholder area contributed 5.0 points, 45% of the maximum score. The FSE Framework scores, by area and measurement object, are shown in Table 94. A complete analysis of this case has been provided in Chapter 6, Discussion of Results.

SUMMARY

This chapter has shown the results of the research. It has two major elements; framework construction and framework validation.

The framework was constructed using *Discoverers' Induction*. The induction

Area	Measurement Object	Data Points	Measure Score	Area Score
External Controls	LAW	0	0.6	1.4/2.0
	STD	0	0.8	
Resources	MAN	1	1.5	6.0/12.0
	MAT	0	2.0	
	CAP	0	1.0	
	PACE	8	0.0	
	METH	2	0.5	
	COMM	0	1.0	
Stakeholders	OWN	1	1.0	5.0/11.0
	MGT	13	1.0	
	CUST	3	1.5	
	SUP	0	0.0	
	USER	8	1.5	
	POLT	2	0.0	
ENVIRONMENT SCORE		38	12.4	10.4/25

Table 94: FBI VCF Environment Element Scores

decomposed 34 scholarly articles into 481 empirical facts which served as the *true facts* for the induction. The 481 *true facts* were classified into 33 subcategories. The 33 subcategories were arranged into five categories and three concepts. The three theoretical concepts were used to construct the framework for software development (see Figure 37). The resulting framework included 14 constructs and 60 measurement objects that satisfied all of the theoretical elements from the induction. The 60 measurement objects used ordinal scales to evaluate the measurement criteria. The completed framework is named the Function, Structure, and Environment (FSE) Framework.

The FSE Framework was validated using two case studies. Each case study was a completed software development project that was considered to be a failure in some aspect. The data from each case study (see Appendices F and G) was evaluated using the 60 measurement objects in the FSE Framework. Scores for each case study were tabulated from the criteria in the 60 measurement objects. The next Chapter will discuss the results.

CHAPTER VI: DISCUSSION OF RESULTS

Chapter 5 presented the results of the inductive development of the framework for software development and the case study validation of the framework. This chapter provides a discussion of the conclusions drawn from the research and is subdivided into four major research outcomes. The 1st section discusses the results of the research as measured by the objectives of the study and the research questions. The 2nd section discusses the use on an inductive method in engineering research. The 3rd section presents the notion of a new paradigm called software systems engineering. The 4th and most important section discusses the FSE Framework areas and measures and their ability to predict the performance as measured by its application on the DIA BHS and the FBI VCF systems. Finally, a cross-case analysis is conducted, further reinforcing the validity of the framework.

RESEARCH OBJECTIVES AND QUESTIONS

This section discusses the major conclusions drawn from the research. The purpose of the research was to develop and apply a systems-based framework for the analysis of software development project performance. The research purpose was supported by research objectives and research questions. The presentation of the research conclusions will begin by reviewing the research purpose and questions identified in Chapter 1.

The research had two objectives. The first objective was to inductively develop a literature based, systemic framework to analyze software development project performance. The second objective was to deploy the generalizable and transportable

analysis framework, applying it to completed software development projects. Based on these objectives the research was focused on answering two research questions:

1. *How does systems theory apply to the analysis of software development project performance?*
2. *What results from the application of a systems-based analysis framework for analyzing performance on a software development project?*

The central issue to be considered in the conclusion is whether the purpose of the research was met, and whether the research questions were answered. The basic answer is that the research did fulfill these requirements and is supported by the achievement of the following significant research outcomes:

- ✓ Employment of a qualitative, inductive method to develop a theoretical framework as an engineering research methodology for complex socio-technical systems.
- ✓ Application of systems theory as a lens for viewing software projects that may serve as the *genesis of a new paradigm* in software engineering; software systems engineering.
- ✓ Emergence of a structured, systemic, three-dimensional framework to be used for predicting software development project performance.
- ✓ Delineation of the areas and measures required to holistically predict software development project performance.

When considered against the outcomes, it can be stated that the purpose of the research, as stated in the objectives and research questions, was met. Each of the research outcomes are worthy of additional discussion.

INDUCTIVE METHOD IN ENGINEERING RESEARCH

The use of a qualitative, inductive method to develop a theoretical framework as an engineering research methodology for complex socio-technical systems is a groundbreaking technique. This is a significant outcome of the research and will be fully discussed in Chapter 7 in the section on *Future Research: Methodological Issues*.

SOFTWARE SYSTEM ENGINEERING

The research has developed an alternative to the widely accepted uni-dimensional model for software development project performance. The three-dimensional FSE Framework is a new, systemic view that uses the dimensions for function, structure and environment in the analysis of software development project performance. The new holistic view may become the *genesis for a new paradigm* for predicting software development project performance; a *paradigm* that requires a shift from the previously accepted model or pattern that viewed software development project performance as a uni-dimensional model of functional processes. The *genesis for the new paradigm* contributes a model that is, in Thomas Kuhn's words, "... an object for further articulation and specification under new or more stringent conditions." (Kuhn, 1996, p. 23) The research has generated a framework that provides a new view of software development project performance; a view based upon systems principles. The application of systems principles provided a theoretical lens through which software development project performance was viewed. The *systems lens* showed that three levels of analysis existed; function, structure, and environment. The *genesis for the new paradigm* fulfills the call for *software based systems engineering* as an element of a paradigm shift for software engineering (Wernick & Hall, 2004) and provides the

software engineering community with a generalizable and transportable framework for evaluating software development projects.

FRAMEWORK EMERGENCE

Systems theory, based on the hierarchy of laws, principles, theorems, hypotheses, and axioms that are systems knowledge served as the *source of the idea* for a systems-based framework with which to evaluate software development project performance. The body of systems knowledge provided the foundation elements that shaped the new viewpoint. *The idea* that a holistic, systems-based lens could be used to evaluate software development projects was *superinduced* upon the empirical facts. The framework that resulted from this induction is now an element of theory residing on the theoretical continuum as a theory of the middle range. As such, the framework exists in the range between hypotheses and grand theory.

Because software development projects are part of the real-world, a three-dimensional world of rich contextual content, a framework that included the elements of context was needed. The three-dimensional FSE Framework incorporated the traditional functional analysis of software processes with two new systems-based holistic elements; structure and environment. Structure analyzed the socio-technical system and cybernetic controls. Environment analyzed the resources, stakeholders, and external controls. The inclusion of context exposes software development projects to a more rigorous analysis than the uni-dimensional assessment frameworks could provide. The FSE Framework was validated for the domain in Figure 45 and will require additional validation. The FSE Framework, developed using a *systems lens*, is emerging and represents a valid alternative to the uni-dimensional frameworks currently in use.

FRAMEWORK AREAS AND MEASURES

In Chapter 5 the completed FSE Framework was presented as a hierarchical model with 3 elements, 11 areas, and 60 measurement objects. The FSE Framework model is a skeletal frame, used to predict software development project performance. When the skeletal frame of the framework is populated with the empirical evidence from a software development project it can predict performance based on an analysis of the evidence arrayed against the 60 measurement objects. A discussion of the FSE Framework's 3 elements, 11 areas, and 50 measurement objects, using the validating case studies is discussed.

Project Type	Federal Government				Federal with 3-4 year duration	
	State Government					
	Local Government			Consortium with 2-3 year duration		
	Government-Commercial Consortium					
	Commercial					
		< 1 Year	1-2 Years	2-3 Years	3-4 Years	> 4 Years
		Project Duration				

Figure 45: Domain of Software Development Projects in Research

DIA BHS Case Study

The empirical facts associated with the DIA BHS software development project were evaluated using the FSE Framework and the details of the case study are contained

in Appendix F. The essential characteristics of the DIA BHS project were characterized using the dimensions in the NCTPO *Pentagon Model*. The project was characterized as follows: Novelty, breakthrough new-to-the-world software; Complexity, collection of subsystems with multiple functions; Technology, super high-tech using technologies that did not exist at project initiation; Pace, time critical where the completion time is crucial for success window of opportunity; Organizational Maturity: ad hoc. Figure 46 shows how these dimensions plot on the NCTPO *Pentagon Model*.

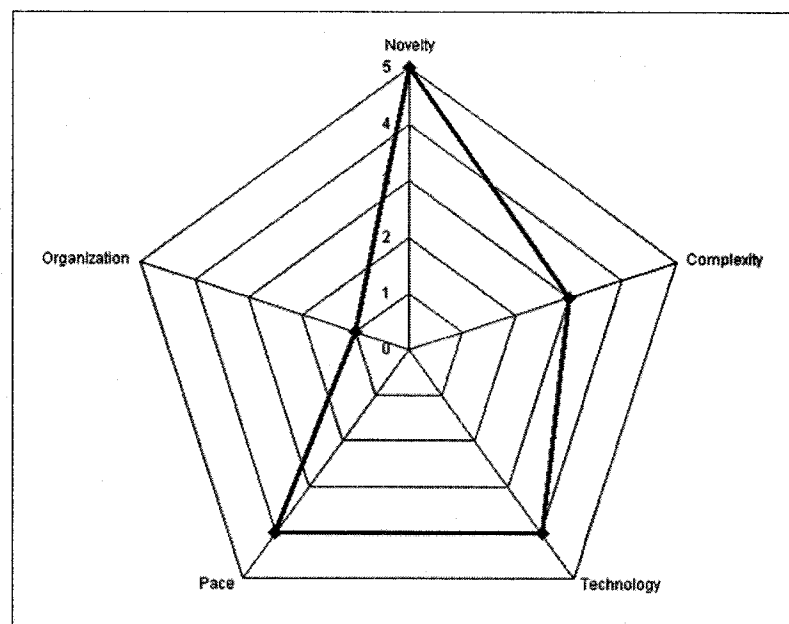


Figure 46: DIA BHS Project Characteristics on NCTPO Pentagon Model

A high-level summary of the application of the FSE Framework to the case study is contained in Table 95.

Framework Element	Areas	Measurement Objects	Overall Score	Detailed Data
Function	5	26	9.0/50	Table 89
Structure	3	20	10.8/25	Table 90
Environment	3	14	10.4/25	Table 91
Total	11	60	30.2/100	

Table 95: High-Level FSE Framework Result for DIA BHS

Because the FSE Framework used three dimensions to evaluate development project performance the results in Table 95 could be plotted on a 3D surface. The DIA BHS project can be represented as the point (9.0, 10.8, 10.4), on the 3D image in Figure 47. The vector from the origin (0, 0, 0) to (9.0, 10.8, 10.4) depicted the project's position on the FSE Framework Grid. The ideal or perfect project would reside at (50, 25, 25). The 3D surface representation would be a particularly useful analysis tool when a number of completed projects could be plotted and a surface density calculated based on macro-level project performance.

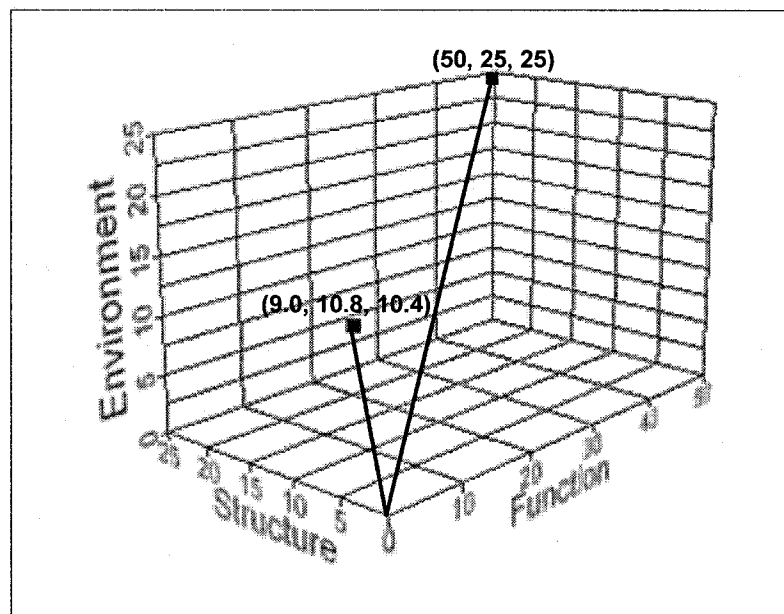


Figure 47: DIA BHS on the FSE Framework Grid

Additional analysis was conducted on the DIA BHS by reviewing each of the FSE Framework's 11 areas. This was accomplished by plotting the scores for each of the element areas on a multi-characteristic Kiviat diagram.

Development Area: Figure 48 shows the four measurement objects for the development area and the associated scores on a four-sided Kiviat diagram. The diagram predicts marginal performance in the development area. The higher scores (0.6 out of a

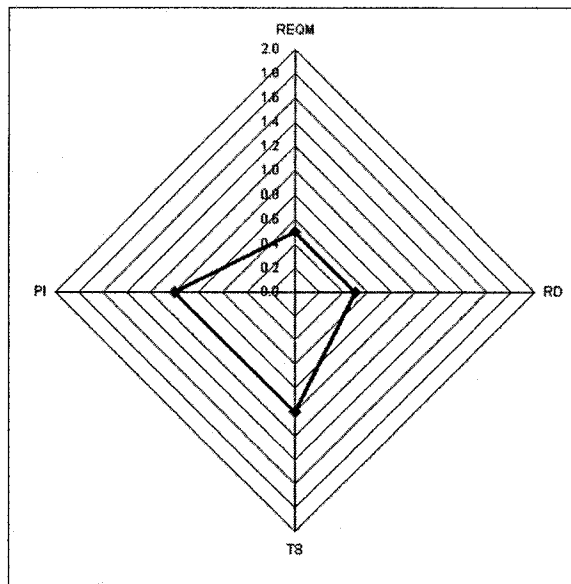


Figure 48: Diagram for DIA BHS Development Area

possible 2.0) in the Product Integration and Technical Solution areas predict difficulty with software analysis, design, and construction (TS) and with integration and testing (PI). The much lower scores (0.3 out of 2.0) in the Requirements Management (REQM) and Requirements Development (RD) areas predict poor performance in the development and management of customer requirements. The framework measurement objects and the associated Kiviat diagram were able to predict poor performance based on empirical data. These predictions were validated by the data from the case study in Appendix G.

Improvement & Training Area: A similar analysis was conducted in the Improvement and Training area. The Kiviat diagram in Figure 51 (all zeros) predicted severe problems in recognizing the need for improvements and in the planning, design, and implementation of an improved process or technique, if required. The diagram also predicted problems with training and the use of just-in-time or on-the-job training as

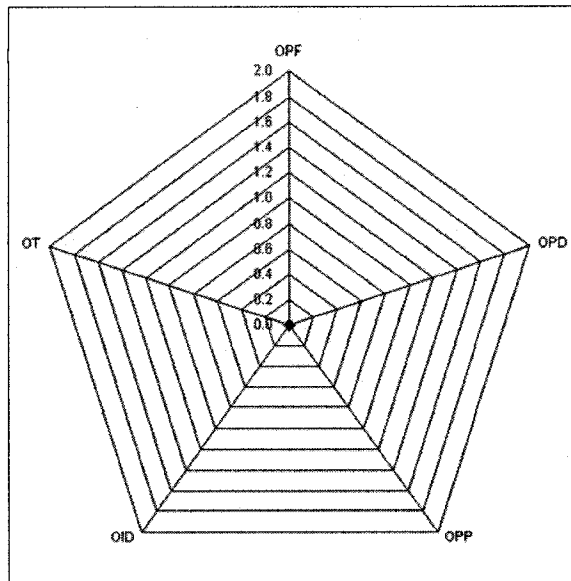


Figure 49: Diagram for DIA BHS Improvement & Training Area

indicators of the deficient situation. Not all of these predictions were able to be validated by data from the case study in Appendix F, although one of the surveys noted that the project used a great deal of *on-the-job training*.

Life Cycle Support Area: Figure 50 shows the eight measurement objects for the life cycle support area on an eight-sided Kiviat diagram. The analysis of the eight

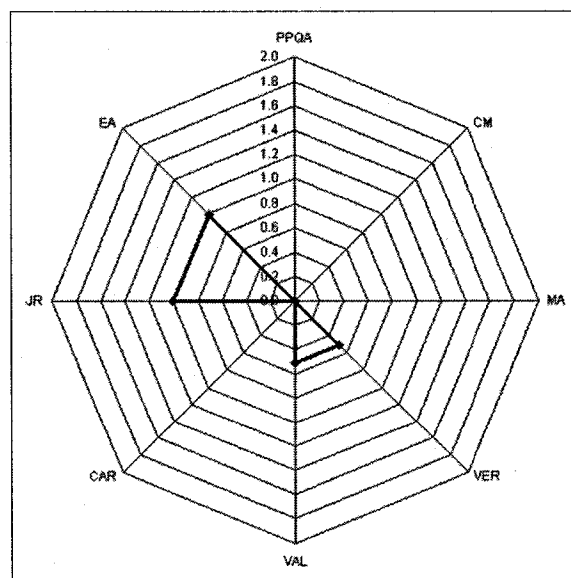


Figure 50: Diagram for DIA BHS Life Cycle Support Area

processes in the life cycle support area predicted major deficiencies in the project's ability to track software components (CM), evaluate quality (PPQA), conduct measurement and analysis (MA) and determine causal analysis and resolution (CAR). The diagram predicted that processes to verify that software products met customer requirements (VER), and validated that the software was capable of performing its intended function when included as part of the system (VAL) would be marginal and the source of problems. It also shows that the project used limited customer reviews (JR) and external audits (EA) to monitor compliance. These predictions were validated by the data from the case study in Appendix F.

Management Area: The framework analysis of the seven measurement objects in the management area predicted similar problems. Figure 51 shows that the project failed to implement any processes for the management of risk (RSKM), management of

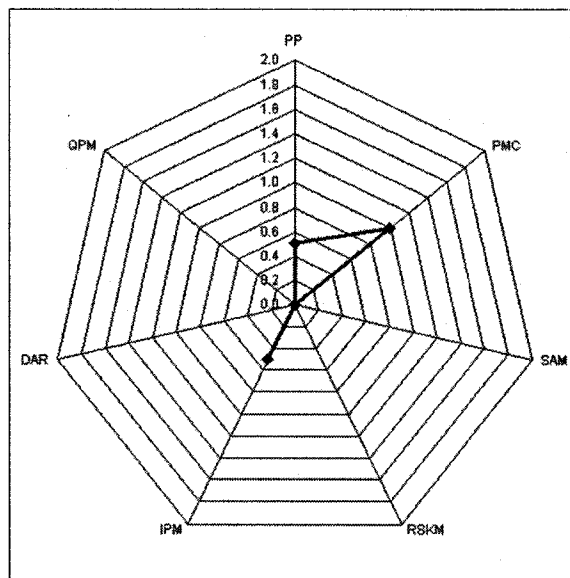


Figure 51: Diagram for DIA BHS Management Area

supplier agreements (SAM), the use of quantitative measures for project management (QPM) or any methods for formal decision analysis and resolution (DAR). The diagram predicted that the project would be able to measure and control ordinary characteristics

such as cost and schedule (PMC). The diagram predicted problems with project planning (PP) and integrated project management (IPM). In summary, the absence of a formal risk management program and only rudimentary methods for project planning and management were particularly ominous warning for a project that included high complexity (COMP) and emerging technology (TECH). Figure 51 shows that the overall management area was inadequate for the software development project and is validated by the data from the case study in Appendix F.

Social System Area: The framework analysis of the six measurement objects in the management area predicted fewer problems. Figure 52 shows that the project had significant experience with the specific application (APEX), the language and tools they

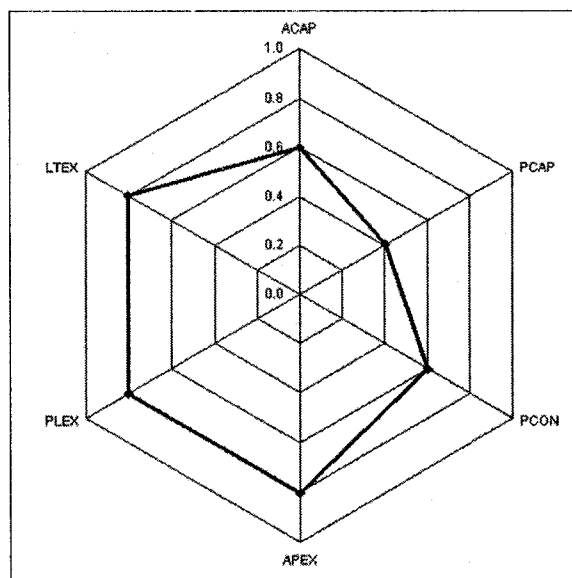


Figure 52: Diagram for DIA BHS Social System Area

were using for development (LTEX), and the platform on which the application was being implemented (PLEX). The project had a nominal score for personnel continuity (PCON) which predicted no problems associated with personnel turnover. The lower scores for programmer (PCAP) and analyst capability (ACAP) predict problems with the

delivery of the applications for which they were responsible. These predictions were validated by the data from the case study in Appendix F.

Cybernetic Functions Area: Figure 53 depicts the five measurement objects for the cybernetic functions area and their scores on a five-sided Kiviat diagram. This shows

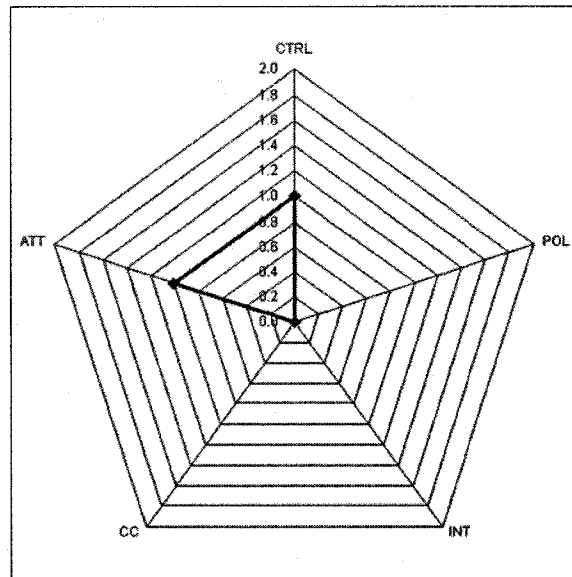


Figure 53: Diagram for DIA BHS Cybernetic Controls Area

that the project used only two of the five cybernetic functions, controls (CTRL) and environmental attenuation (ATT), to the level where a score was able to be recorded. This predicts that the project would have trouble with internal communications (CC), intelligence about the external environment and the affect on the project (INT), and the coordination between the controls for the BHS project and the larger DIA project. The framework predicted that the cybernetic controls were totally insufficient for the development project and is clearly validated by the data from the case study in Appendix F.

Technical System Area: The framework analysis of the nine measurement objects in the technical system area predicted more problems. Figure 54 shows that the project had reasonable scores for five of the nine technical system measurement criteria.

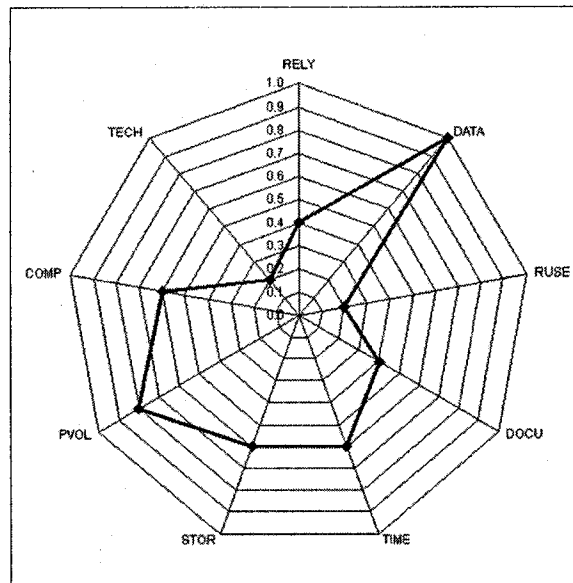


Figure 54: Diagram for DIA BHS Technical System Area

The database size (DATA) was fully addressed and measures for complexity (COMP), platform volatility (PVOL), main storage (STOR), and system response time (TIME) were adequately addressed. Low scores were received for life cycle documentation (DOCU) and systems reliability (RELY) which predicted problems in these areas. A very low score was received for the process used to address the reuse of software components (RUSE), a factor that predicted increased complexity in the design of the software. The final measurement object addressed the use of emerging technology (TECH) on the project, which predicted massive problems in the implementation and testing for the project. The overall technical system in place for the project was severely insufficient and is validated by the data from the case study in Appendix F.

Resource Area: Figure 55 depicts the six measurement objects for the resource area and their scores on a six-sided Kiviat diagram. The figure predicted resource problems for the projects. The project nominally addressed the availability of personnel (MAN), external communications (COMM), and the use of formal methods (METH). A low score in the availability of critical materials (MAT) predicted problems with the

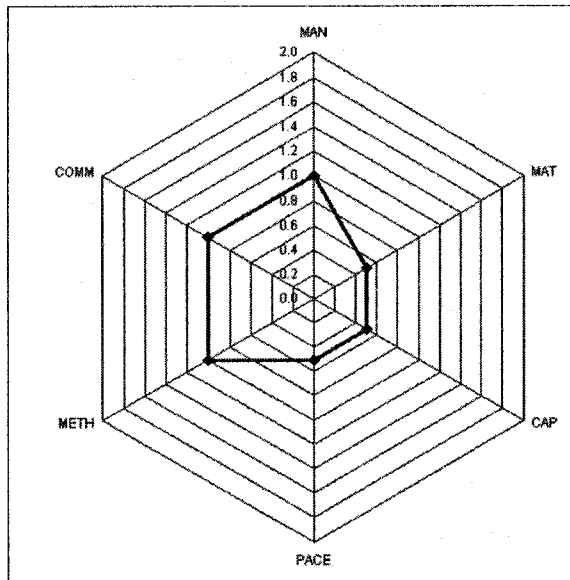


Figure 55: Diagram for DIA BHS Resources Area

implementation schedule for the system. Funding capital investment for improvement initiatives and training as part of the project budget (CAP) is a risky practice when the project has a compressed schedule and is using emerging technology, and the low score predicted problems in this area. The schedule pace for the project (PACE) received a low score due to the critical nature of the baggage handling system. This measure predicted problems associated with the schedule, a fact that is omnipresent in the data presented in the case study in Appendix F.

Stakeholder Area: The framework analysis of the six measurement objects in the stakeholder area predicted severe problems. Figure 56 shows that the project addressed two of the six stakeholder measurement criteria sufficiently. The project addressed the involvement of the system's users (USER) and the customer (CUST). However, the project did not fully involve its suppliers (SUP) in the project. The involvement of the company's board of directors (OWN) and management external to the project (MGT) were nominal. Finally, the project existed within a highly political

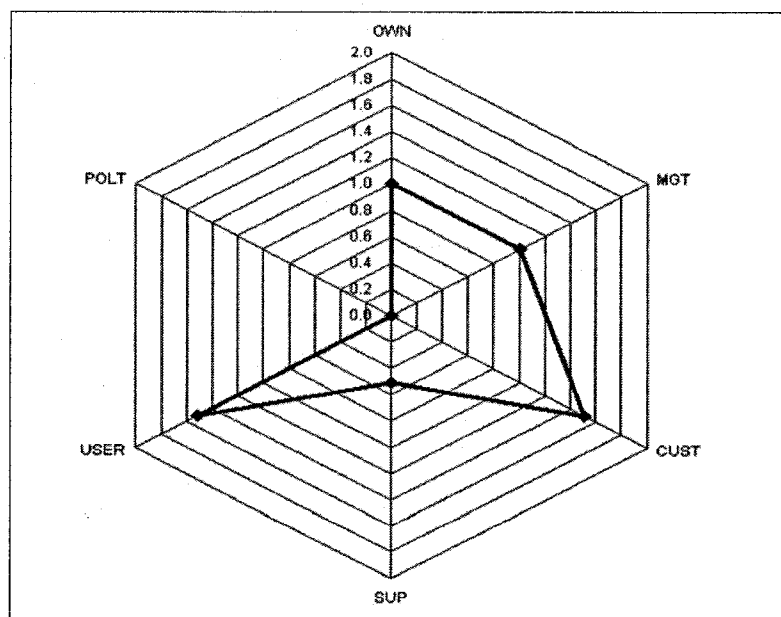


Figure 56: Diagram for DIA BHS Stakeholders Area

environment (POLT) which predicted a variety of problems that affected the project. The political environment shows that the project had a serious deficiency in their ability to address the stakeholders associated with the project. The framework predicted that the political environment (in Figure 56), coupled with the absence of cybernetic controls (in Figure 53), would cause severe problems for the project. This prediction is clearly validated by the data from the case study in Appendix F.

Infrastructure and External Control Areas: Infrastructure and external controls are two separate areas from two separate elements of the framework. However, it is convenient to plot them on the same diagram because of the magnitude of their measurement objects are equal. The framework measures for external controls show that neither the project nor the parent company (BAE Systems) were involved with the development of laws (LAW) or standards (STD) that affected the software system they were designing. The framework predicted that this situation may have long-term

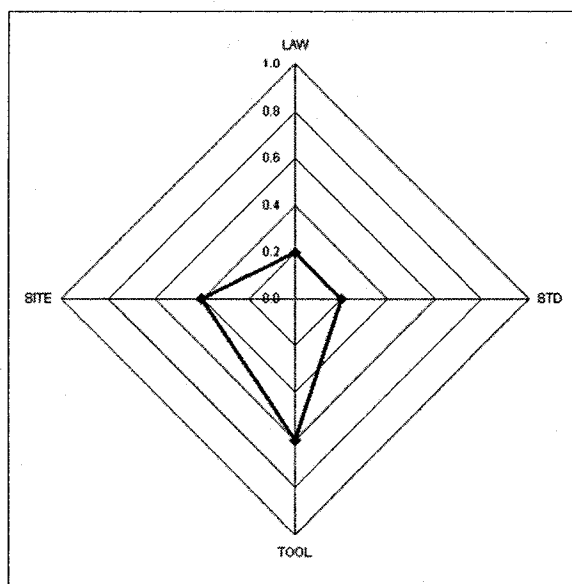


Figure 57: Diagram for DIA BHS External Controls and Infrastructure Areas

consequences for the project. The framework measures for infrastructure showed that the software tools (TOOL) in-use on the project were of average utility, but that the location of the development team (SITE) may be of concern. Neither of these predictions were able to be validated by data from the case study in Appendix F.

In summary, the FSE Framework performed well by predicting poor performance for the DIA BHS. The prediction was based on the project's cumulative score and supported by cascading level of detail in the elements, areas, and measurement objects. The framework was able to show how the function, structure, and environment elements contributed to the overall score. Further analysis showed how each of the eleven major areas contributed to the score. Finally, the most detailed analysis was provided by the framework's 60 measurement objects, each of which contributed the basic measures used in the overall prediction for performance. The use of multi-characteristic Kiviat diagrams provided an easily comprehensible graphic used to predict strengths, weaknesses, and predicted performance for each of the 11 major areas.

FBI VCF System Case Study

The empirical facts associated with the FBI VCF system software development project were evaluated using the FSE Framework and the details of the case study are contained in Appendix G. The essential characteristics of the FBI VCF project were characterized using the dimensions in the NCTPO *Pentagon Model*. The project was characterized as follows: Novelty, replacement software; Complexity, collection of subsystems with multiple functions; Technology, medium-tech with some new technology; Pace, Blitz where an immediate solution was necessary; Organizational Maturity: ad hoc. Figure 46 shows how these dimensions plot on the NCTPO *Pentagon Model*.

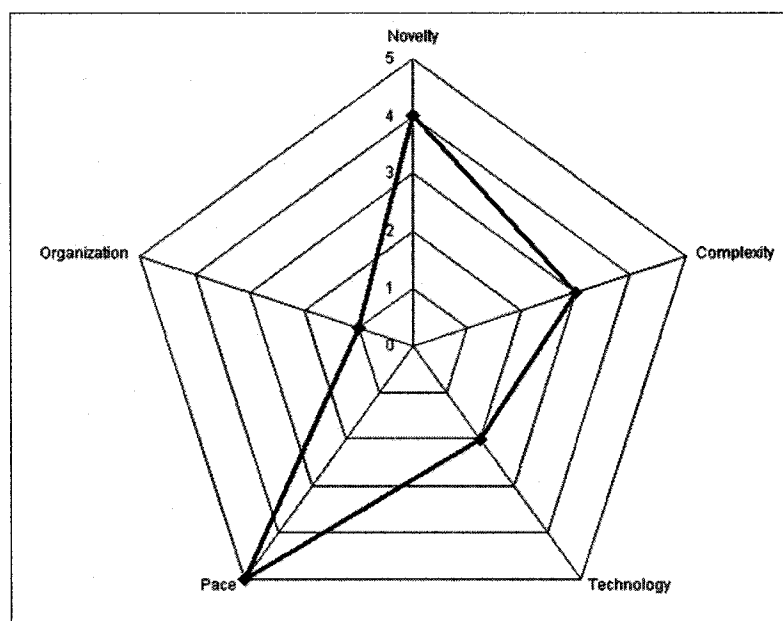


Figure 58: FBI VCF Project Characteristics on NCTPO Pentagon Model

A high-level summary of the application of the FSE Framework to the case study is contained in Table 95.

Because the FSE Framework used three dimensions to evaluate development project performance the results in Table 96 could be plotted on a 3D surface. The FBI

Framework Element	Areas	Measurement Objects	Overall Score	Detailed Data
Function	5	26	7.5/50	Table 92
Structure	3	20	10.2/25	Table 93
Environment	3	14	12.4/25	Table 94
Total	11	60	30.1/100	

Table 96: High-Level FSE Framework Result for FBI VCF

VCF project can be represented as the point (7.5, 10.2, 12.4), on the 3D image in Figure 59. The vector from the origin (0, 0, 0) to (7.5, 10.2, 12.4) depicted the project's position

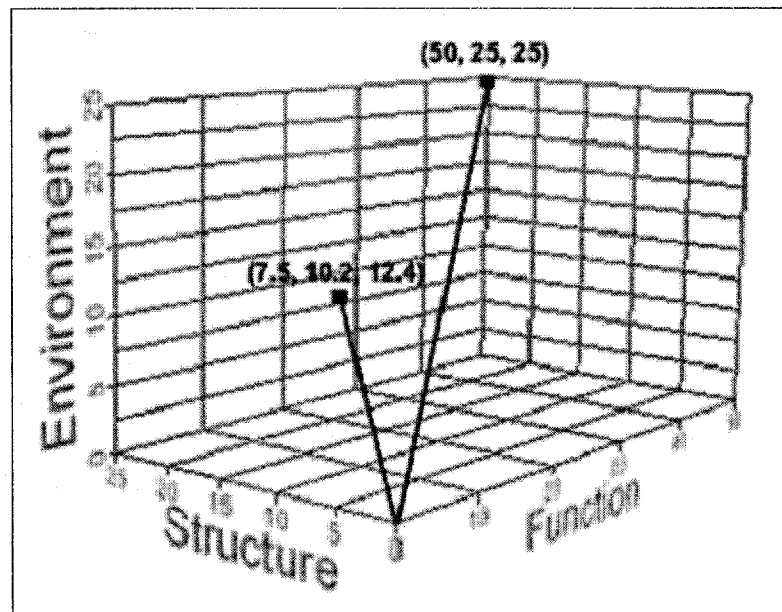


Figure 59: FBI VCF on the FSE Framework Grid

on the FSE Framework Grid. The ideal or perfect project would reside at (50, 25, 25). The 3D surface representation would be a particularly useful analysis tool when a number of completed projects could be plotted and a surface density calculated based on macro-level project performance.

Additional analysis was conducted on the FBI VCF by reviewing each of the FSE Framework's 11 areas. This was accomplished by plotting the scores for each of the element areas on a multi-characteristic Kiviat diagram.

Development Area: Figure 60 shows the four measurement objects for the development area and the associated scores on a four-sided Kiviat diagram.

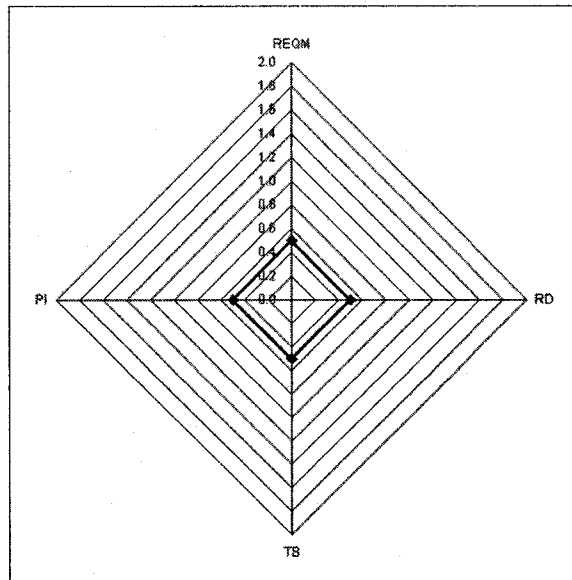


Figure 60: Diagram for FBI VCF Development Area

The diagram predicts marginal performance in the development area. All of the scores show that the project has a basic capability in each area but does not perform the process at a sustainable level. The low scores in the Product Integration and Technical Solution areas predict difficulty with software analysis, design, and construction (TS) and with integration and testing (PI). The equally low scores in the Requirements Management (REQM) and Requirements Development (RD) areas predict poor performance in the development and management of customer requirements. The framework measurement objects and the associated Kiviat diagram were able to predict poor performance based on empirical data. These predictions were validated by the data from the case study in Appendix G.

Improvement & Training Area: A similar analysis was conducted in the Improvement & Training area. The Kiviat diagram in Figure 61 (all zeros) predicted

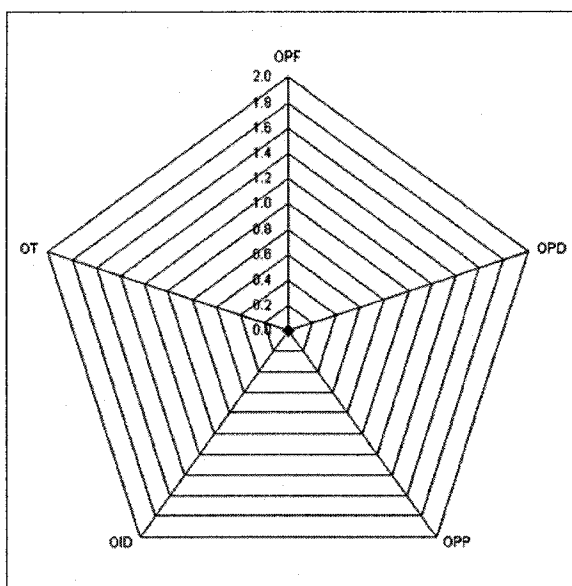


Figure 61: Diagram for FBI VCF Improvement & Training Area

severe problems in recognizing the need for improvements and in the planning, design, and implementation of an improved process or technique, if required. The diagram also predicted problems with training and the use of just-in-time or on-the-job training as indicators of the deficient situation. Not all of these predictions were able to be validated by data from the case study in Appendix G.

Life Cycle Support Area: Figure 62 shows the eight measurement objects for the life cycle support area on an eight-sided Kiviat diagram. The analysis of the eight processes in the life cycle support area predicted major deficiencies in the project's ability evaluate quality (PPQA), conduct measurement and analysis (MA) and determine causal analysis and resolution (CAR). The diagram predicted that processes used to track software components (CM), verify that software products met customer requirements (VER), and validated that the software was capable of performing its intended function when included as part of the system (VAL) would be marginal and the source of problems. It also shows that the project used limited customer reviews (JR) and external

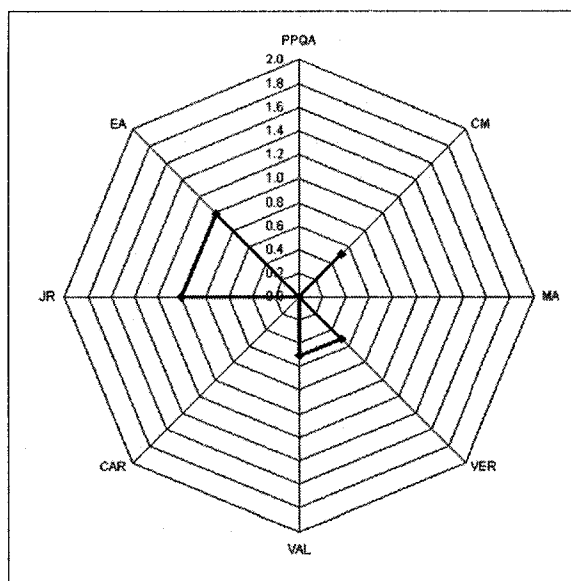


Figure 62: Diagram for FBI VCF Life Cycle Support Area

audits (EA) to monitor compliance. The project's failure to adopt the recommendations in the external audits confirms the finding associated with the absence of a causal analysis and resolution (CAR) process.

Management Area: The framework analysis of the seven measurement objects in the management area predicted similar problems. Figure 51 shows that the project

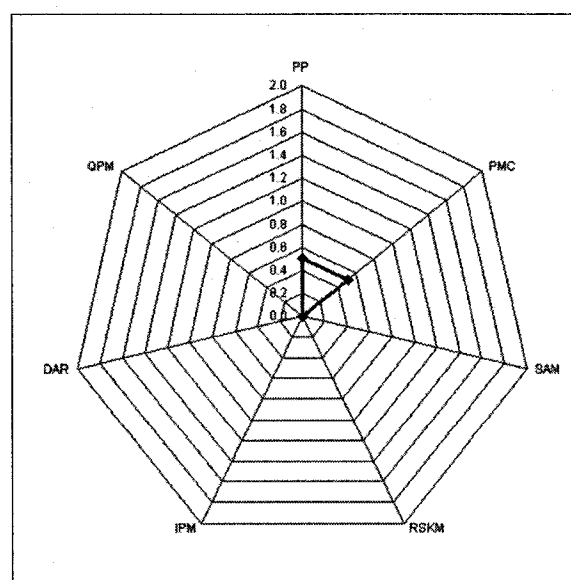


Figure 63: Diagram for FBI VCF Management Area

failed to implement any processes for the management of risk (RSKM), management of supplier agreements (SAM), integrated project management (IPM), the use of quantitative measures for project management (QPM) or any methods for formal decision analysis and resolution (DAR). The diagram predicted that the project would be able to measure and control ordinary characteristics such as cost and schedule (PMC) and conduct basic project planning (PP) activities. These predictions are validated by the major audit findings by the FBI and Department of Justice Inspector General's reports and the GAO audits. Figure 63 shows that the overall management area was inadequate for the software development project and is validated by additional facts in the case study in Appendix G.

Social System Area: The framework analysis of the six measurement objects in the management area predicted fewer problems. Figure 64 shows that the project had nominal experience with the specific application (APEX), the language and tools they

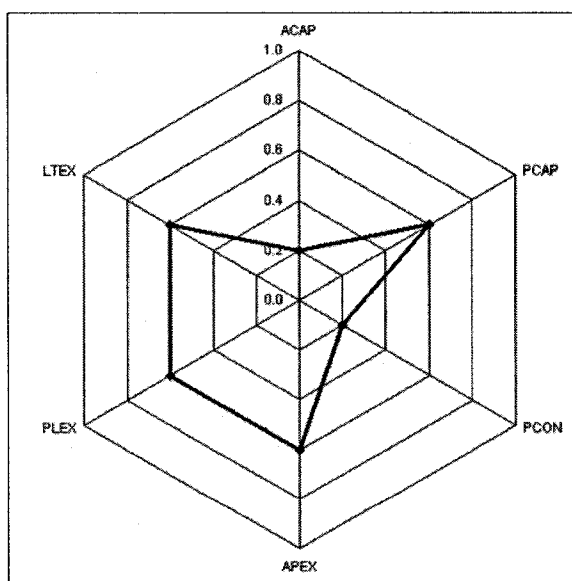


Figure 64: Diagram for FBI VCF Social System Area

were using for development (LTEX), the platform on which the application was being implemented (PLEX) and for programmer capability (PCAP). However, the project had a very low score for personnel continuity (PCON) which predicted problems associated with the turnover of key personnel on the project. The very low score for analyst capability (ACAP) predicted problems with the development of the system requirements for which they were responsible. These predictions were validated by the data from the case study in Appendix G.

Cybernetic Functions Area: Figure 65 depicts the five measurement objects for the cybernetic functions area and their scores on a five-sided Kiviat diagram. This shows

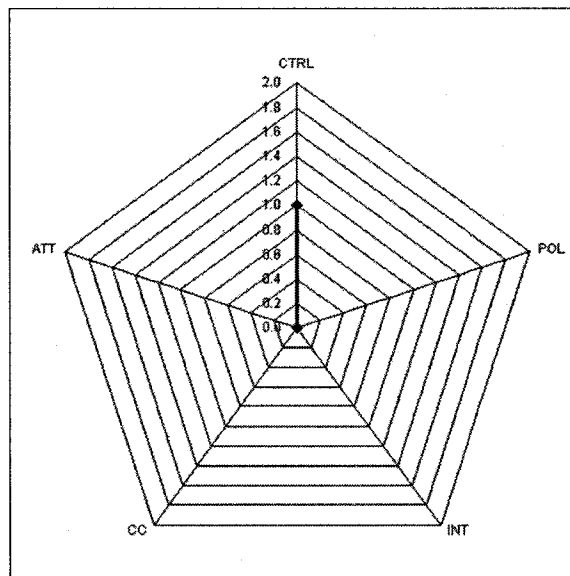


Figure 65: Diagram for FBI VCF Cybernetic Controls Area

that the project used only one of the five cybernetic functions, controls (CTRL) to the level where a score was able to be recorded. This predicts that the project would have trouble with internal communications (CC), intelligence about the external environment and the affect on the project (INT), the coordination between the controls for the VCF project and the larger Trilogy project, and environmental attenuation (ATT). The

framework predicted that the level of cybernetic controls were totally insufficient for the development project and are clearly validated by the data from the case study in Appendix G.

Technical System Area: The framework analysis of the nine measurement objects in the technical system area predicted more problems. Figure 66 shows that the project had reasonable scores for five of the nine technical system measurement criteria.

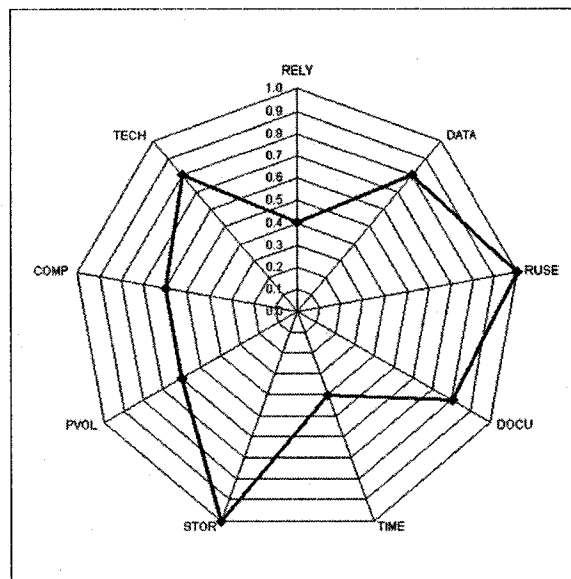


Figure 66: Diagram for FBI VCF Technical System Area

The database size (DATA) was fully addressed and measures for complexity (COMP), platform volatility (PVOL), main storage (STOR), and system response time (TIME) were adequately addressed. Low scores were received for life cycle documentation (DOCU) and systems reliability (RELY) which predicted problems in these areas. A very low score was received for the process used to address the reuse of software components (RUSE), a factor that predicted increased complexity in the design of the software. The final measurement object addressed the use of emerging technology (TECH) on the project, which predicted massive problems in the implementation and

testing for the project. The overall technical system in place for the project was severely insufficient and is validated by the data from the case study in Appendix G.

In the Environment element the FBI VCF scored points in the external control (1.4 points), resource (6.0 points), and stakeholder (5.0 points) areas.

Resource Area: Figure 67 depicts the six measurement objects for the resource area and their scores on a six-sided Kiviat diagram. The figure predicted resource problems for the projects. The project nominally addressed external communications (COMM) and capital investment for improvement initiatives and training (CAP). Neither the availability of critical materials (MAT) nor personnel (MAN) were predicted to cause problems. The failure to implement formal methods (METH) in more than a few process areas predicted problems with system integration activities related to common practices.

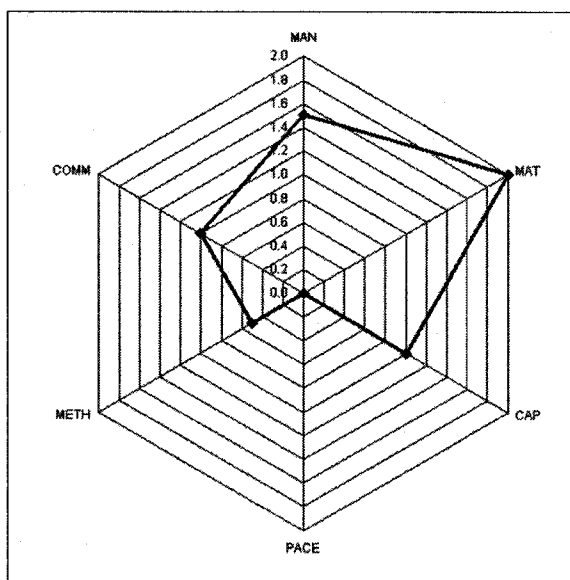


Figure 67: Diagram for FBI VCF Resources Area

However, the most problematic area was the prediction of problems related to the schedule pace. Projects that have a *blitz* pace must be highly integrated and use activities

that ensure coordination between functions. This predicted problems associated with the schedule, a fact that is omnipresent in the data presented in the case study in Appendix G.

Stakeholder Area: The framework analysis of the six measurement objects in the stakeholder area predicted severe problems. Figure 68 shows that the project addressed two of the six stakeholder measurement criteria sufficiently. The project addressed the involvement of the system's users (USER) and the customer (CUST).

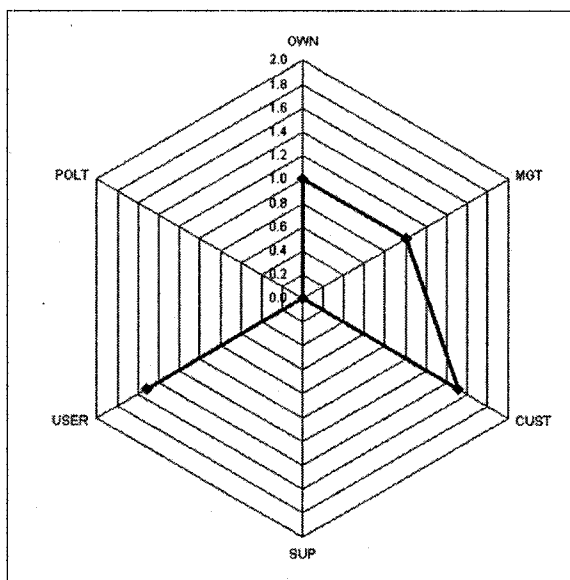


Figure 68: Diagram for FBI VCF Stakeholders Area

However, the project did not fully involve its suppliers (SUP) in the project. The involvement of the company's owners (OWN) predicted no problems in that area. The framework predicted a nominal amount of problems associated with management external to the project (MGT). Finally, the project existed within a highly political environment (POLT) which predicted a variety of problems that affected the project. The political environment shows that the project had a serious deficiency in their ability to address the stakeholders associated with the project. The framework predicted that the political environment (in Figure 68), coupled with the absence of cybernetic controls (in

Figure 65), would cause severe problems for the project. This prediction is clearly validated by the data from the case study in Appendix G.

Infrastructure and External Control Areas: Infrastructure and external controls are two separate areas from two separate elements of the framework. However, it is convenient to plot them on the same diagram because of the magnitude of their measurement objects are equal. Figure 69 shows that the project or parent company (SAIC) were involved with the development of laws (LAW) or standards (STD) that affected the software system they were designing. The framework predicted no negative consequences associated with these external controls.

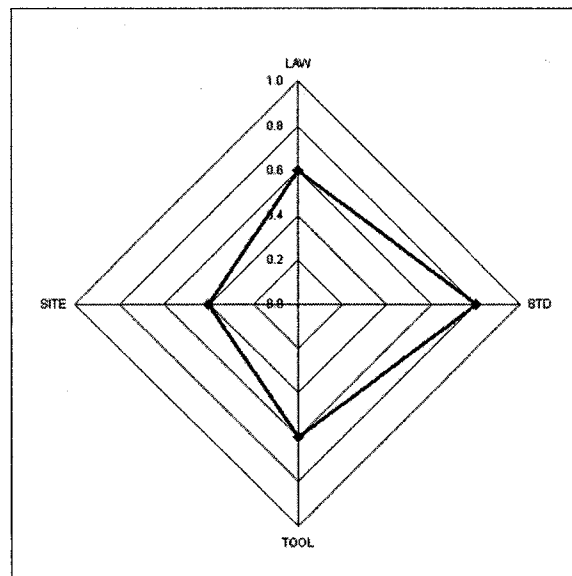


Figure 69: Diagram for FBI VCF External Controls and Infrastructure Areas

The framework measures for infrastructure showed that the software tools (TOOL) in-use on the project were of average utility, but that the location of the development team (SITE) may be of concern. Neither of these predictions were able to be validated by data from the case study in Appendix G.

In summary, the FSE Framework performed well by predicting poor performance for the FBI VCF project. The prediction was based on the project's cumulative score and

supported by cascading level of detail in the elements, areas, and measurement objects. The framework was able to show how the function, structure, and environment elements contributed to the overall score. Further analysis showed how each of the eleven major areas contributed to the score. Finally, the most detailed analysis was provided by the framework's 60 measurement objects, each of which contributed the basic measures used in the overall prediction for performance. The use of multi-characteristic Kiviat diagrams provided an easily comprehensible graphic used to predict strengths, weaknesses, and predicted performance for each of the 11 major areas.

CROSS-CASE ANALYSIS

The framework was able to predict, based on the empirical data from both cases that each project would perform poorly. Furthermore, the framework was able to isolate areas where specific processes could be predicted to adversely affect performance. Cross-case analysis was conducted by plotting the NCTPO project characteristics and specific area scores for both cases using Kiviat diagrams.

Project Characteristics: Figure 70 shows that novelty and pace will be significant high-level factors requiring controls during the execution of each project.

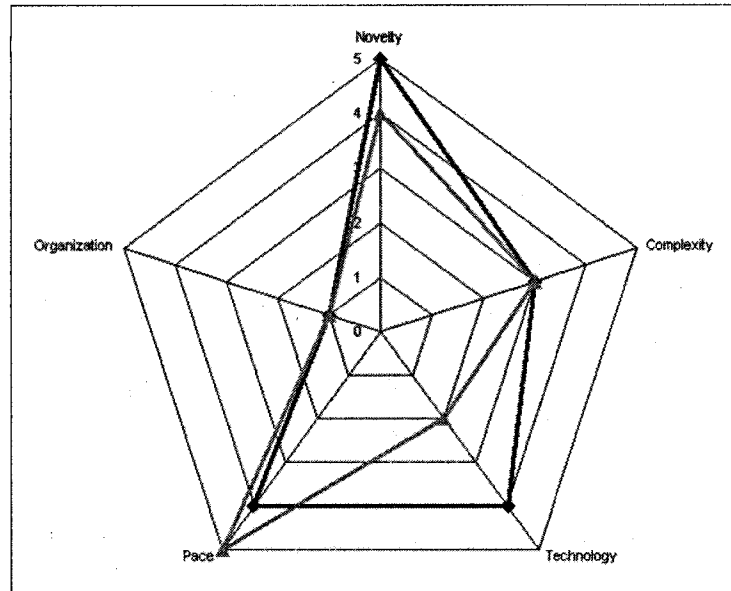


Figure 70: Cross-Case Diagram for Project Characteristics

Improvement & Training Area: Figure 61 (all zeros) predicted severe problems in recognizing the need for improvements and in the planning, design, and implementation of an improved process or technique, if required. The diagram also predicted problems with training. Not all of these predictions were able to be validated by data from either of the case studies in Appendices F and G.

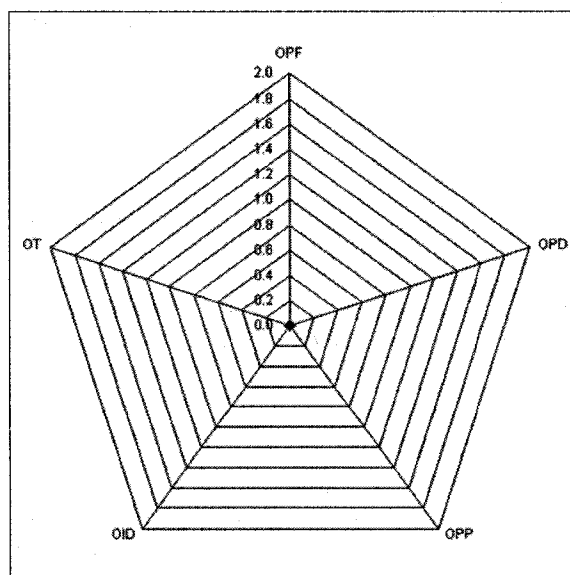


Figure 71: Cross-Case Diagram for Improvement & Training Area

Life Cycle Support Area: Figure 62 predicted major deficiencies in each project's ability evaluate quality (PPQA), conduct measurement and analysis (MA) and determine causal analysis and resolution (CAR). The diagram predicted that processes used to track software components (CM), verify that software products met customer requirements (VER), and validated that the software was capable of performing its

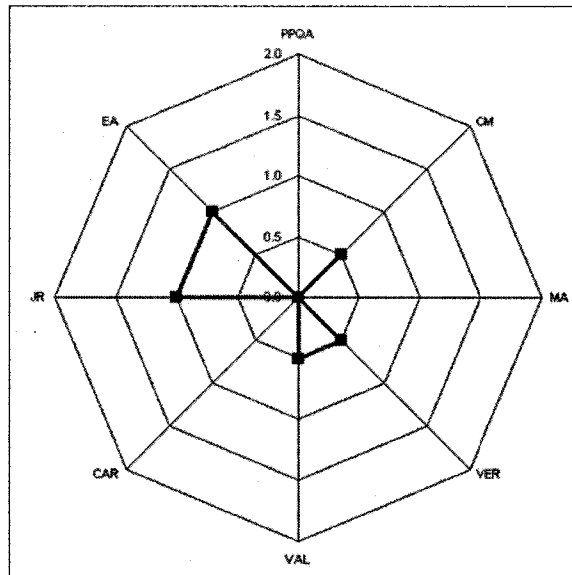


Figure 72: Cross-Case Diagram for Life Cycle Support Area

intended function when included as part of the system (VAL) would be marginal and the source of problems. Finally, each project used limited customer reviews (JR) and external audits (EA) assist in monitoring compliance.

Management Area: Figure 63 shows that each project failed to implement any processes for risk management (RSKM), supplier agreements (SAM), quantitative

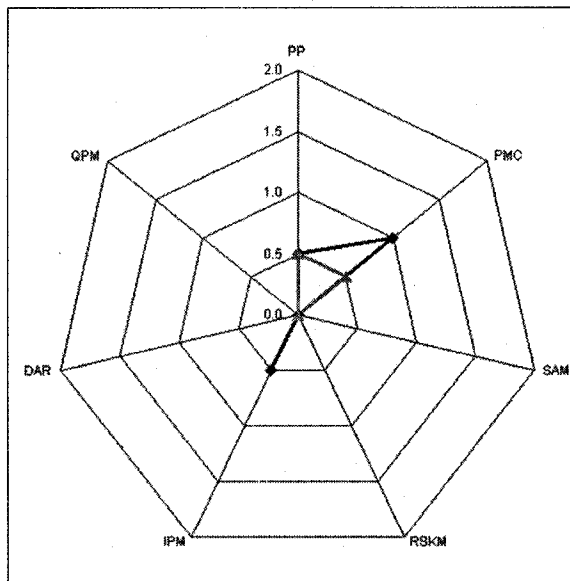


Figure 73: Cross-Case Diagram for Management Area

project management (QPM), and formal decision analysis and resolution (DAR). The diagram predicted that each project would be able to measure and control ordinary characteristics such as cost and schedule (PMC) and conduct basic project planning (PP) activities. These predictions are validated by the facts in Appendices F and G and are believed to be major factors contributing to poor project performance.

Cybernetic Functions Area: Figure 65 predicted problems for each project based on the lack of formal cybernetic control functions. Neither project addresses project communications (CC), intelligence functions aimed at understanding the external environment (INT), and policies (POL) that ensure internal controls (CTRL) are balanced with the environment external to the project. The framework predicted that the level of

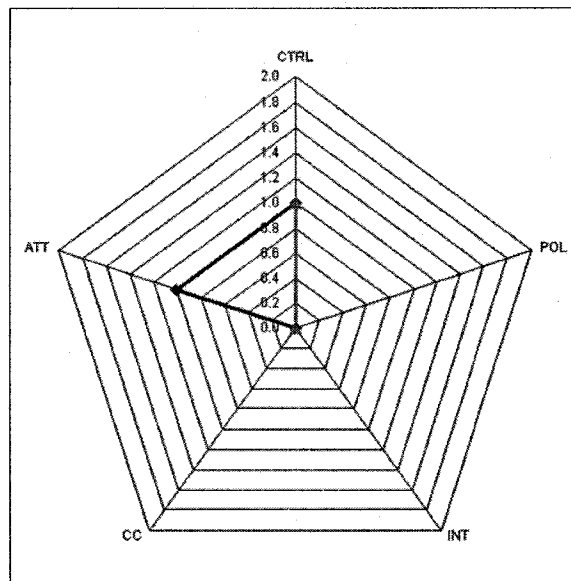


Figure 74: Cross-Case Diagram for Cybernetic Functions Area

cybernetic controls in-use on each project were totally insufficient.

Resource Area: Figure 75 predicted problems associated with the pace of the project. It also predicts problems associated with the use of standards (METH) in only

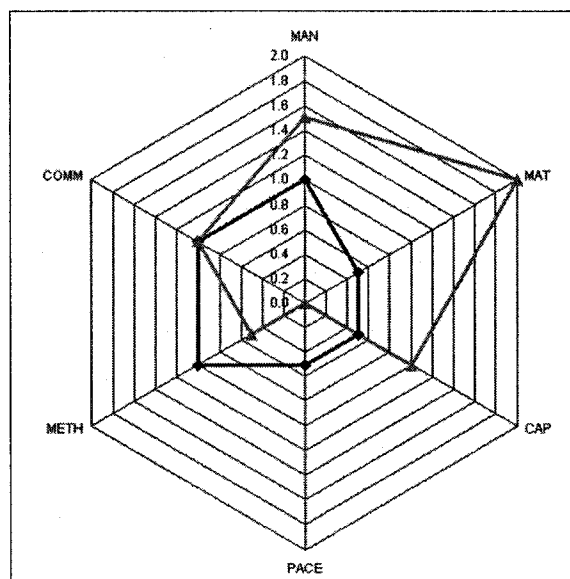


Figure 75: Cross Case Diagram for Resources Area

a few process areas, a contributing cause for problems during system integration activities.

Stakeholder Area: Figure 68 predicts problems due to politics on each project.

The diagram also shows that each project has done a good job addressing the concerns of the systems user's (USER) and customer (CUST). Finally, the diagram predicted a nominal amount of problems associated with management external to each project (MGT).

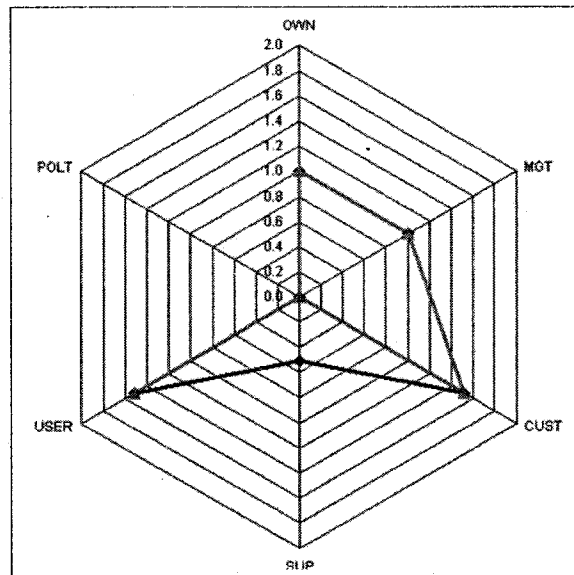


Figure 76: Cross-Case Diagram for Stakeholders Area

In summary, the cross-case analysis shows that the FSE Framework performed well by predicting poor performance for both projects. The prediction was based on the projects' cumulative scores and was supported by cascading level of detail in the areas and measurement objects. Further analysis showed how six of the eleven areas predicted poor performance for each project. Additional analysis focused on the linkage between process area predictions (i.e. the intensive political environment (in Figure 68) coupled with the absence of cybernetic functions (in Figure 65) is an area for future research.

SUMMARY

The chapter has presented the results of the research and how it fulfilled the objectives of the study and answered the principal research questions. It also discussed the future use of the inductive method in engineering research and presented the notion of a new paradigm called software systems engineering. Most importantly, the chapter discussed the FSE Framework areas and measures and demonstrated their ability to predict performance as measured by their application on the DIA BHS and the FBI VCF systems. Finally, a cross-case analysis was conducted. The cross-case analysis showed that the FSE Framework was able to isolate areas where specific processes indicated poor performance. The cross-case analysis provided additional validity for the framework. The next chapter will discuss the conclusions of the research.

CHAPTER VII: CONCLUSION

Chapter 6 presented a discussion of the results of the application of the framework for software development to the case studies. This chapter presents the limitations of the study, the implications of the results, and makes recommendations for areas in which further research may be directed.

LIMITATIONS OF THE STUDY

Before discussing the implications for the research, it is appropriate to mention its limitations. Three limitations are discussed.

Limitation 1: Information Sources

The case study research included a questionnaire that relied upon the memories of professionals that reported on projects that had been completed or cancelled some time earlier. This factor raised the prospect that there may be some error in the data, especially when the questionnaire answer was not supported by empirical evidence from the literature. Two factors were used to mitigate the possibility of error. The first factor ensured that more than one respondent was used in the case study. The second factor required the researcher to use formal logic in determining how the data was used to rate the project against the measurement criteria. Figure 44 contains the logic used in comparing the questionnaire answers to the empirical data collected from the literature.

Limitation 2: Scale Development and Measurement

The scales developed for the framework were ordinal scales. No effort to calibrate the scales against empirical evidence was conducted. This effort is a logical extension of this research and should be considered if the framework is adopted for wider use.

Limitation 3: Software Development Project Domains

The research did not include completed software development projects from all applicable domains. In order to fully describe the large field of diverse software development projects the selection criteria included categories that were mutually-exclusive, exhaustive, and comparable (Gerring, 2001). The selection criteria were: (1) *project type* in which the software development project was characterized by the customer to whom it was delivering the software, and (2) *project duration* in which case the software development project was characterized by the duration, from start to finish, for complete delivery of the software. The research included projects in the domain in Figure 45.

IMPLICATIONS OF THE RESULTS

The implications of the results of this research, for both research and practice are addressed in the section.

Implications for Research

The results of the research study contribute to existing and future research in several important ways. First, the study provides evidence that a systems-based framework for software development project performance can reliably predict development project performance. This is an important point because a framework, as an extension of theory is tasked with representing a large variety of observational facts (Maxwell, 1962). The development of a framework requires the same rigor as the development of a theory and must be based on scientific inquiry. Failure to base the development of a framework on rigorous research may limit the utility of the framework by failing to include relevant or exclude irrelevant data. The use of a formal method for

the development of a framework, based on systemic principles, ensures that the framework addresses all of the relevant data and follows William Whewell's directions (1858):

When we inquire, what facts are to be made the materials of Science, perhaps the answer which we should most commonly receive would be, that they must be True Facts, as distinguished from any mere inferences or opinions of our own. (p. 51)

The inclusion of only *true facts* is an essential element in the construction of a valid and applicable framework. The use of a formal inductive method, as presented in Chapter 4, ensures that the researcher is following an established and rigorous path of scientific research; one that follows the generally accepted Canons of Science.

Second, the study provides a framework which may be used to conduct additional research on software development projects. The ability to expand the research to projects with different characteristics is an immediate objective. Research may be directed toward areas in the software development domain (see Figure 45) that were not a part of this study. Additional research in specific software industry's and software types may be of interest. Both areas will serve to extend the applicability and utility of the framework.

Third, the research makes a significant contribution to the body of knowledge on qualitative research in engineering. The use of a formal inductive method, *Discoverers' Induction*, in an engineering study is an important step in the acceptance of qualitative methods. This is particularly important because the mechanical models in the quantitative research methods have proven to be of very limited value in situations where the contextual environment of the research study has had any real significance. The increased use qualitative methods, prevalent in the sociological and psychological

domains, in engineering studies may contribute to the understanding of a variety of real-world phenomena.

Implications for Practice

The results of the research study contribute to the practice of software engineering and the software engineering management knowledge area (Abran & Moore, 2004). The framework provides a systems-based instrument that may be used to evaluate and predict software development project performance. The framework may be applied to development project's that are in planning, in-progress, or completed. The framework provides a more robust framework with which the software practitioner may evaluate software projects. The framework's three dimensions include provisions that go beyond the traditional measures for development processes. The framework includes measures that address the surrounding environment, the socio-technical system, and the cybernetic functions required to ensure system viability. The software practitioner will be able to use the framework in a variety of modes. Of particular interest are:

The FSE Framework may be used as a pre-project design and diagnostics tool. The prospective development project may use the FSE Framework as part of a preparation or readiness-to-start review to understand where a project will need assistance. The analysis may be extended to make bid/no-bid decisions based on potential development projects.

The FSE Framework may be employed as a customer-based audit tool. A customer can conduct a multi-level review (i.e. 3 elements, 11 areas, 60 objects) to understand where developer may have areas for concern.

The FSE Framework may be utilized as a project management maintenance tool. An on-going project may use the FSE Framework to conduct an in-progress review of the total project or an element, area, or object to better understand and indicate performance.

The FSE Framework may be used as a post-project analysis tool. The ability to conduct a rapid post project analysis is an important but under-performed function on software development projects. The ability to have a clear framework for capturing performance related issues immediately after completion can contribute significantly to the understanding of and development of lessons-learned.

The FSE Framework may be employed as a project or organizational-level improvement analysis tool. Many software development organizations struggle to understand where they should apply their limited resources in order to gain improvements in development performance. The FSE Framework can indicate where improvement is needed, and focus improvement efforts on the 3 element, 11 areas, or 60 objects. Once the FSE Framework has been employed for a number of projects indicators can be developed that show which objects, areas, and elements have the greatest effect on project level performance. The ability to use the FSE Framework as an organizational level strategic improvement tool has clear potential for software developers.

FUTURE RESEARCH RECOMMENDATIONS

One of the roles of rigorous scholarly research is to provide paths for further research. This section considers the current state of the systems and software bodies of knowledge and the relationship to the research findings. Taken in total, these clearly indicate fertile areas for additional research using the FSE framework. The development and articulation of the concept of a systems-based software development project

framework provided some substance to the notions surrounding holistic analysis.

However, there are many areas yet to be addressed by additional rigorous research. The following areas for future research are recommended.

Future Research: Philosophical Issues

The research presented a systems philosophy that was a product of the worldview of the researcher and the focus of the research. The research addressed the need for a holistic approach to the complex social system called a software development project. A system-based framework was developed to address the complex of processes required to develop a software system. The processes were social in nature and included a rich contextual environment that significantly affected the development performance. The scope of the research included materials from sociology, psychology, computer science, management, and engineering. The central philosophical question becomes:

- Can a single systems-based framework (model) be conceived that is both sufficiently inclusive and implementable, in order to address all aspects of the complex social system for software development?

Future Research: Theoretical Issues

Much of the discussion in the literature and in the data presented in the research study was focused upon trying to develop a framework to predict software development project performance in order to transform the complex social system of software development. A very narrow definition of software development project performance was adopted for the dissertation (i.e. the degree to which a software development project accomplishes its cost, schedule and business goals during the development of the software.) However, this narrow definition did not include the performance of the software after delivery. Future research should include moving the definition of software

development project performance from this narrow conceptual definition to a theoretical level, where the definition moves to a more complete level that clearly articulates what performance means. Specifically, the research must move forward to develop a theoretical construct for software development project performance that establishes:

- How is software development project performance defined? What elements of the complex social system influence performance? How can these elements be characterized to capture the contribution (positive or negative) to overall system performance?

Future Research: Methodological Issues

A unique qualitative method of induction was used in this research. *Discoverers' Induction* was coupled with modern techniques (open, axial, and selective coding) to decompose and classify empirical facts and aid in the construction of the conception. This method relied upon the idea, where the researcher brought the idea to bear upon the facts. It is important to note that the source of the conception from the mind of the researcher was based upon academic training and real-world experience. Both of these sources served to create ideas in the mind that correspond closely enough with reality that true theories about the real-world could be developed using these ideas as their conceptions. This area of qualitative research should be considered by researchers that will be generating middle-range theory, especially if the researcher is considering using grounded theory. As discussed in Chapter 4, being able to use an idea as the basis for the generation of theory is not an option in grounded theory research. Grounded theory does not include preconceived theory at the start of the technique, requiring the researcher to be *pure in mind*, a condition that is particularly complicated to achieve when generating theory. Specific research should be focused upon:

- Can *Discoverers' Induction* (as updated in this research) continue to be used as a modern method for inductively developing middle range theory when the researcher has an *idea*? Further validation of this technique would be a substantial contribution to the field of engineering management.

Future Research: FSE Framework Issues

The FSE framework requires additional research. The validation and calibration of the measurement objects, and its application to additional domains are areas for further investigation. Research questions include:

- Can the FSE Framework be used to predict performance across all types of software development projects?
- Are the measurement objects included in the FSE Framework inclusive enough and parsimoniously distributed?
- Are the measurement object scales in the FSE framework properly calibrated?
Can a better measure be developed?
- Can the framework be used to predict the emergence of industry specific patterns for software development?
- Can regression analysis be used to find correlations between framework elements in order to further clarify the size of the framework?

SUMMARY

This chapter presented the limitations of the study, the implications of the results, and recommendations for future research. Future research directions were proposed with the emphasis in four areas. (1) Philosophical concepts that address the ability to use a single framework to predict the behavior of the complex social system of software

development. (2) The theoretical definition for software development project performance. (3) The methodological issues surrounding the use of *Discoverers'* *Induction* for future qualitative research where a specific idea is used to develop theories of the middle range. (4). Extension of work on the FSE Framework with attention on the validation and calibration of the measurement objects, and its application to additional software development project domains.

REFERENCES

- Abdel-Hamid, T.K. (1984). *The Dynamics of Software Development Project Management: An Integrative Systems Dynamics Perspective* (Doctoral Dissertation, Massachusetts Institute of Technology, 1984).
- Abdel-Hamid, T.K. (1988). "Understanding the '90% Syndrome' in Software Project Management: A Simulation-Based Case Study," *Journal of Systems and Software*, Vol. 8, No. 4, pp. 319-330.
- Abdel-Hamid, T.K. (1989a). "The Dynamics of Software Project Staffing: A System Dynamics Based Simulation Approach," *IEEE Transactions on Software Engineering*, Vol. 15, No. 2, pp. 109-119.
- Abdel-Hamid, T.K. (1989b). "A Study of Staff Turnover, Acquisition, and Assimilation and Their Impact on Software Development Cost and Schedule," *Journal of Management Information Systems*, Vol. 6, No.1, pp. 21-40.
- Abdel-Hamid, T.K. (1990). "Investigating the Cost/Schedule Trade-Off in Software Development," *IEEE Software*, Vol. 7, No. 1, pp. 97-105.
- Abdel-Hamid, T. K. (1992). "Investigating the Impacts of Managerial Turnover/Succession on Software Project Performance," *Journal of Management Information Systems*, Vol. 9, No.2, pp. 127-144.
- Abdel-Hamid, T.K. (1993). "A Multiproject Perspective of Single-Project Dynamics," *Journal of Systems and Software*, Vol. 22, No. 3, pp. 151-166.
- Abdel-Hamid, T.K. (1996). *The Slippery Path to Productivity Improvement*. *IEEE Software*, Vol. 13, No. 4, pp. 43-52.
- Abdel-Hamid, T.K. & Madnick, S.E. (1983). "The Dynamics of Software Project Scheduling," *Communications of the ACM*, Vol. 26, No. 5, pp. 340-346.
- Abdel-Hamid, T.K. & Madnick, S.E. (1989). "Lessons Learned from Modeling the Dynamics of Software Project Management," *Communications of the ACM*, Vol. 32, No. 12, pp. 1426-1455.
- Abdel-Hamid, T.K. & Madnick, S.E. (1990). "The Elusive Silver Lining: How we Fail to Learn from Software Development Failures," *Sloan Management Review*, Vol. 32, No. 1, pp. 39-48.
- Abdel-Hamid, T.K. & Madnick, S.E. (1991). *Software Project Dynamics: An Integrated Approach*. Englewood Cliffs, NJ: Prentice Hall, Inc.
- Abdel-Hamid, T.K. Sengupta, K. & Ronan, D. (1999). "Software Project Control: An Experimental Investigation of Judgment with Fallible Information," *IEEE Transactions on Software Engineering*, Vol. 19, No. 6, pp. 603-612.
- Abdel-Hamid, T.K., Sengupta, K. & Sweet, C. (1999). "The Impact of Goals on Software Management: An Experimental Investigation," *MIS Quarterly*, Vol. 23, No. 4, pp. 531-555.

- Abran, A. & Moore, J. (Eds.) (2004). *Guide to the Software Engineering Body of Knowledge, 2004 Version*. Los Alamitos: The Institute of Electrical and Electronics Engineers.
- Achinstein, P. (1965). "Theoretical Models," *British Journal for the Philosophy of Science*, Vol. 16, No. 62, pp. 102-120.
- Ackoff, R.L. (1971). "Toward a System of Systems Concepts," *Management Science*, Vol. 17, No. 11, pp. 661-671.
- Ackoff, R.L. (1974). "The Systems Revolution," *Long Range Planning*, Vol. 7, No. 6, pp. 2-20.
- Ackoff, R.L. (1979a). "The Future of Operational Research Is Past," *Journal of the Operational Research Society*, Vol. 30, No. 2, pp. 93-104.
- Ackoff, R.L. (1979b). "Resurrecting the Future of Operational Research," *Journal of the Operational Research Society*, Vol. 30, No. 3, pp. 198-199.
- Adams, K.M. (1986). *The Effect of Cutouts on Strength of GRP for Naval Ship Hulls* (Master's Thesis, Massachusetts Institute of Technology, 1986).
- Adman, P. & Warren, L. (2000). "Participatory Sociotechnical Design of Organizations and Information Systems – An Adaptation of ETHICS Methodology," *Journal of Information Technology*, Vol. 15, No. 1, pp. 39-51.
- Ahire, S.L. & Devaraj, S. (2001). "An Empirical Comparison of Statistical Construct Validation Approaches," *IEEE Transactions on Engineering Management*, Vol. 48, No. 3, pp. 319-329.
- Aladwani, A.M. (2002). "An Integrated Performance Model of Information Systems Projects," *Journal of Management Information Systems*, Vol. 19, No. 1, pp. 185-210.
- Alavi, M. & Carlson, P. (1992). "A Review of MIS Research and Disciplinary Development," *Journal of Management Information Systems*, Vol. 8, No. 4, pp. 45-62.
- Araujo, L. (1995). "Designing and Refining Hierarchical Coding Frames." In U. Kelle (Ed.). *Computer-Aided Qualitative Data Analysis: Theory Methods and Practice*. (pp. 96-104). Thousand Oaks: Sage Publications.
- Ashby, R. (1947). "Principles of the Self-Organizing Dynamic System." *Journal of General Psychology*, Vol. 37, pp. 125-128.
- Ashby, R. (1956). *An Introduction to Cybernetics*. London: Chapman & Hall, Ltd.
- Augustine, N. (2002). "Ethics and the Second Law of Thermodynamics," *The Bridge*, Vol. 32, No. 3, pp. 4-7.
- Baccarini, D. (1999). "The Logical Framework Method for Defining Project Success," *Project Management Journal*, Vol. 30, No. 4, pp. 25-32.
- Bacharach, S.B. (1989). "Organizational Theories: Some Criteria for Evaluation," *Academy of Management Review*, Vol. 14, No. 4, pp. 496-515.

- Bai, G. & Lindberg, L. (1999). "A Sociocybernetic Approach to Information Systems Development," *Kybernetes*, Vol. 28, No. 6/7, pp. 792-809.
- Bailer-Jones, D.M. (2003). "When Scientific Models Represent," *International Studies in the Philosophy of Science*, Vol. 17, No. 1, pp. 59-74.
- Banathy, B.H. (1991). *Systems Design of Education: A Journey to Create the Future*. Englewood Cliffs: Educational Technology Publications.
- Banathy, B.H. (1996). *Designing Social Systems in a Changing World*. New York: Plenum.
- Banker, R.D. & Kemerer, C.F. (1992). "Performance Evaluation Metrics for Information Systems Development: A Principal-Agent Model," *Information Systems Research*, Vol. 3, No. 4, pp. 379-400.
- Baroudi, J.J., Olson, M.H. & Ives, B. (1986). "An Empirical Study of the Impact of User Involvement on System Usage and Information Satisfaction," *Communications of the ACM*, Vol. 29, No. 3, pp. 232-238.
- Barton, S.L., Duchon, D. & Dunegan, K.J. (1989) "An Empirical Test of Staw and Ross's Prescriptions for the Management of Escalation of Commitment Behavior in Organizations," *Decision Sciences*, Vol. 20, No. 3, pp. 532-544.
- Beer, S. (1979). *The Heart of the Enterprise*. New York: John Wiley and Sons.
- Beer, S. (1981). *The Brain of the Firm*. New York: John Wiley and Sons.
- Beer, S. (1984). "The Viable Systems Model: Its Provenance, Development, Methodology and Pathology," *Journal of the Operational Research Society*, Vol. 35, No 1, pp. 7-26.
- Beishon, R.J. (1976). *Systems Organization: The Management of Complexity* (2nd ed.). New York: Harper and Row for the Open University Press.
- Benbasat, I., Goldstein, D.K. & Mead, M. (1987). "The Case Research Strategy in Studies of Information Systems," *MIS Quarterly*, Vol. 11, No. 3, pp. 369-386
- Bennatan, E. (2000). *On Time Within Budget: Software Project Management Practices and Techniques* (3rd ed.). New York: John Wiley and Sons.
- Bennetts, P.D.C., Wood-Harper, A.T. & Mills, S. (2000). "An Holistic Approach to the Management of Information Systems Development--A View Using a Soft Systems Approach and Multiple Viewpoints," *Systemic Practice and Action Research*, Vol. 13, No. 2, pp. 189-205.
- Berdie, D.R., Anderson, J.F. & Niebuhr, M.A. (1986). *Questionnaires: Design and Use* (2nd ed.). Metuchen, NJ: The Scarecrow Press.
- von Bertalanffy, L. (1968). *General System Theory: Foundations, Development, Applications* (Rev. ed.). New York: George Braziller.
- Blanchard, B. & Fabrycky, W. (1998). *Systems Engineering and Analysis* (3rd ed.). Upper Saddle River: Prentice Hall.

- Blau, P.M. (1970). "A Formal Theory of Differentiation in Organizations," *American Sociological Review*, Vol. 35, No. 2, pp. 201-218.
- Blum, B.I. (1994). "A Taxonomy of Software Development Methods," *Communications of the ACM*, Vol. 37, No. 11, pp. 82-94.
- Blumer, H. (1970). "Methodological Principles of Empirical Science." In N.K. Denzin (Ed.), *Sociological Methods: A Sourcebook* (pp. 20-39). Chicago: Aldine Publishing Company.
- Boehm, B.W. (1981). *Software Engineering Economics*. Englewood Cliffs: Prentice-Hall.
- Boehm, B. (2000). "Unifying Software Engineering and Systems Engineering," *IEEE Computer*, Vol. 33, No. 3, pp. 114-116.
- Boehm, B. (2006). "Some Future Trends and Implications for Systems and Software Engineering Processes," *Systems Engineering*, Vol. 9, No. 1, pp. 1- 19.
- Boehm, B.W., Abts, C., Brown, A.W., Chulani, S., Clark, B.K., Horowitz, E., Madachy, R., Reifer, D.J., & Steece, B. (2000). *Software Cost Estimation with COCOMO II*. Upper Saddle River: Prentice Hall PTR.
- Bohr, N. (1928) "The Quantum Postulate and the Recent Development of Atomic Theory," *Nature*, Vol. 121(Supplement), No. 3050, pp. 580-590.
- Bohr, N. (1937). "Causality and Complementarity," *Philosophy of Science*, Vol. 4, No. 3, pp. 289-298.
- Boloix, G. & Robillard, P.N. (1995). "A Software System Evaluation Framework," *IEEE Computer*, Vol. 28, No. 12, pp.17-26.
- Bonoma, T.V. (1985). "Case Research in Marketing: Opportunities, Problems, and a Process," *Journal of Marketing Research*, Vol. 22, No. 2, pp. 199-208.
- Boulding, K. (1956). "General Systems Theory – The Skeleton of Science," *Management Science*, Vol. 2, No. 3, pp. 197-208.
- Bourgeois, L.J. (1979). "Toward a Method of Middle-Range Theorizing," *Academy of Management Review*, Vol. 4, No. 3, pp. 443-447.
- Bowers, P. (2001). "Raytheon Stands Firm on Benefits of Process Improvement," *Crosstalk*, Vol. 14, No. 3, pp. 9-12.
- Brooks, F.P. (1987). "No Silver Bullet: Essence and Accidents of Software Engineering," *IEEE Computer*, Vol. 20, No. 4, pp. 10-19.
- Budgen, D. (2003). *Software Design* (2nd ed.). Harlow, England: Pearson Education.
- Burrell, G. & Morgan, G. (1979). *Sociological Paradigms and Organizational Analysis: Elements of the Sociology of Corporate Life*. London: Heinemann.
- Butts, R.E. (1965). "Necessary Truth in Whewell's Theory of Science," *American Philosophical Quarterly*, Vol. 2, No. 3, pp. 161-181.
- Camilleri, S.F. (1962). "Theory, Probability and Induction in Social Research," *American Sociological Review*, Vol. 27, No. 2, pp. 170-178.

- Campbell, J. (1982). *Grammatical Man: Information, Entropy, Language and Life*. New York: Simon and Schuster.
- Card, D.N. (2000). "Sorting Out Six Sigma and the CMM," *IEEE Software*, Vol. 17, No. 3, pp. 11-13.
- Carlile, P.R. & Christensen, C.M. (2005). "The Cycles of Theory Building in Management Research," Unpublished Manuscript.
- Carmines, E.C. & Zeller, R.A. (1979). *Reliability and Validity Assessment* Beverly Hills: Sage Publications.
- Carroll, J.M. & Swatman, P.A. (2000). "Structured-case: A Methodological Framework for Building Theory in Information Systems Research," *European Journal of Information Systems*, Vol. 9, No. 4, pp. 235-242.
- Cepeda, G. & Martin, D. (2005). "A Review of Case Studies Publishing in *Management Decision* 2003-2004: Guides and Criteria for Achieving Quality in Qualitative Research," *Management Decision*, Vol. 43, No. 6, pp. 851-876.
- Charette, R.N. (1989). *Software Engineering Risk Analysis and Management*. New York: McGraw-Hill Book Company.
- Charette, R.N. (1991). *Applications Strategies for Risk Analysis*. New York: McGraw-Hill Book Company.
- Charette, R.N., Dwinnell, L.M. & McGarry, J. (2004). "Understanding the Roots of Process Performance Failure," *Crosstalk*, Vol. 17, No. 8, pp. 18-22.
- Checkland, P.B. (1978). "The Origins and Nature of 'Hard' Systems Thinking," *Journal of Applied Systems Analysis*. Vol. 5, No. 2, pp. 99-110.
- Checkland, P.B. (1993). *Systems Thinking, Systems Practice*. New York: John Wiley & Sons.
- Checkland, P. & Scholes, J. (1999). *Soft Systems Methodology in Action*. Chichester: John Wiley & Sons, Ltd.
- Chen, W. & Hirschheim, R. (2004). "A Paradigmatic and Methodological Examination of Information Systems Research from 1991 to 2001," *Information Systems Journal*, Vol. 14, No. 3, pp. 197-235.
- Cherns, A. (1976). "The Principles of Sociotechnical Design," *Human Relations*, Vol. 29, No. 8, pp. 783-792.
- Cherns, A. (1987). "The Principles of Sociotechnical Design Revisited," *Human Relations*, Vol. 40, No. 3, pp. 153-161.
- Chestnut, H. (1967). *Systems Engineering Methods*. New York: Wiley
- Chiang, I.R. & Mookerjee, V.S. (2004). "A Fault Threshold Policy to Manage Software Development Projects," *Information Systems Research*, Vol.15, No. 1, pp. 3-21.
- Chidamber, S.R. & Kemerer, C.F. (1994). "A Metrics Suite for Object-Oriented Design," *IEEE Transactions on Software Engineering*, Vol. 20, No. 6, pp. 476-493.

- Christensen, C.M. & Raynor, M.E. (2003). "Why Hard-Nosed Executives Should Care About Management Theory," *Harvard Business Review*, Vol. 81, No. 9, pp. 67-74.
- Churchman, C.W. (1968). *The Systems Approach*. New York: Dell Publishing.
- Clemson, B. (1991). *Cybernetics: A New Management Tool*. Philadelphia: Gordon and Breach Science Publishers.
- Cliff, N. (1993). "What Is and Isn't Measurement." In G. Keren & C. Lewis (Eds.), *A Handbook for Data Analysis in the Behavioral Sciences: Methodological Issues* (pp. 59-93). Hillsdale, NJ: Lawrence Erlbaum Associates, Publishers.
- Cliff, N. & Keats, J.A. (2003). *Ordinal Measurement in the Behavioral Sciences*. Mahwah, NJ: Lawrence Erlbaum Associates.
- Clouse, A. & Wells, C. (2000). "Transitioning from EIA/IS-731 to CMMI," *Crosstalk*, Vol. 13, No. 7, pp. 15-20.
- Coallier, F. (2003). "International Standardization in Software and Systems Engineering," *Crosstalk*, Vol. 16, No. 2, pp. 18-22.
- Cockburn, A. (2000). "Selecting a Project's Methodology," *IEEE Software*, Vol. 17, No. 4, pp. 64-71.
- Cohen, J. (1960). "A Coefficient of Agreement for Nominal Scales," *Educational and Psychological Measurement*, Vol. 20, No. 1, pp. 37-46.
- Conant, R.C. & Ashby, W.R. (1970). "Every Good Regulator of a System Must be a Model of that System," *International Journal of Systems Science*, Vol. 1, No. 2, pp. 89-97.
- Coombs, C.H. (1964). *A Theory of Data*. New York: John Wiley.
- Coombs, C.H., Raiffa, H. & Thrall, R.M. (1954). "Some Views on Mathematical Models and Measurement Theory," *Psychological Review*, Vol. 61, No. 2, pp. 132-144.
- Corbin, J. & Strauss, A. (1990). "Grounded Theory Research: Procedures, Canons, and Evaluative Criteria," *Qualitative Sociology*, Vol. 13, No. 1, pp. 3-21.
- Costner, H.L. (1969). "Theory, Deduction, and the Rules of Correspondence," *American Journal of Sociology*, Vol. 75, No. 2, pp. 245-263.
- Creswell, J.W. (2003). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. (2nd ed.). Thousand Oaks: Sage Publications.
- Cronbach, L.J. (1951). "Coefficient alpha and the internal structure of tests," *Psychometrika*, Vol. 16, No. 3, pp. 297-334.
- Crosby, P.B. (1979). *Quality is Free*. New York: McGraw-Hill Book Company.
- Cusumano, M.A. (1989). "The Software Factory: A Historical Interpretation," *IEEE Software*, Vol. 6, No. 2, pp. 23-30.
- Cusumano, M.A. & Kemerer, C.F. (1990). "A Quantitative Analysis of U.S. and Japanese Practice and Performance in Software Development," *Management Science*, Vol. 36, No. 11, pp. 1384-1406.

- Cusumano, M., MacCormack, A., Kemerer, C.F. & Crandall, B. (2003) "Software Development Worldwide: The State of the Practice," *IEEE Software*, Vol. 20, No. 6, pp. 28-34.
- Darke, P., Shanks, G. & Broadbent, M. (1998). "Successfully Completing Case Study Research: Combining Rigor, Relevance and Pragmatism," *Information Systems Journal*, Vol. 8, No. 4, pp. 273-289.
- Day, J. (2000) "Software Development as Organizational Conversation: Analogy as a Systems Intervention," *Systems Research and Behavioral Science*, Vol. 17, No. 4, pp. 349-358.
- DeBrabander, B. & Edström, A. (1977). "Successful Information System Development Projects," *Management Science*, Vol. 24, No. 2, pp. 191-199.
- Deephouse, C., Mukhopadhyay, T., Goldenson, D.R. & Kellner, M.I. (1996). "Software Processes and Project Performance," *Journal of Management Information Systems*, Vol. 12, No. 3, pp. 187-205.
- Denzin, N.K. (1971). "The Logic of Naturalistic Inquiry," *Social Forces*, Vol. 50, No. 2, pp. 166-182.
- Denzin, N.K. & Lincoln, Y.S. (2000). "The Discipline and Practice of Qualitative Research." In N.K. Denzin & Y.S. Lincoln (Eds.). *Handbook of Qualitative Research* (2nd ed.). (pp. 1-28). Thousand Oaks: Sage Publications.
- DeVellis, R.F. (2003). *Scale Development: Theory and Applications*. (2nd ed.). Thousand Oaks: Sage Publications.
- Dewey, J. (1938). *Logic: The Theory of Inquiry*. New York: Holt, Rinehart and Winston.
- Diaz, M. & Sligo, J. (1997). "How Software Process Improvement Helped Motorola," *IEEE Software*, Vol. 14, No. 5, pp. 75-81.
- Dorfman, M. & Thayer, R. (Eds.). (2002). *Software Engineering*. New York: Wiley-IEEE Computer Society Press.
- Dubin, R. (1975). "Causality and Social System Analysis," *International Journal of General Systems*, Vol. 2, pp. 107-113.
- Dubin, R. (1978). *Theory Building*. (Rev. ed.). New York: The Free Press.
- Ducasse, C.J. (1951a). "Whewell's Philosophy of Scientific Discovery, Part I," *The Philosophical Review*, Vol. 60, No. 1, pp. 56-69.
- Ducasse, C.J. (1951b). "Whewell's Philosophy of Scientific Discovery, Part II," *The Philosophical Review*, Vol. 60, No. 2, pp. 213-244.
- Dutton, J.E. & Dukerich, J.M. (1991). "Keeping an Eye on the Mirror: Image and Identity in Organizational Adaptation," *Academy of Management Journal*, Vol. 34, No. 3, pp. 517-554.
- Duvall, L.M. (1995). "A Study of Software Management: The State of Practice in the United States and Japan," *Journal of Systems and Software*, Vol. 31, No. 2, pp. 109-124.

- Dyba, T. (2005). "An Empirical Investigation of the Key Factors for Success in Software Process Improvement," *IEEE Transactions on Software Engineering*, Vol. 31, No. 5, pp. 410-424.
- Edwards, P.N. (2000). "The World in a Machine: Origins and Impacts of Early Computerized Global Systems Models." In A.C. Hughes & T.P. Hughes (Eds.), *Systems, Experts, and Computers: The Systems Approach in Management and Engineering, World War II and After* (pp. 221-253). Cambridge: MIT Press.
- Edwards, J.R. & Bagozzi, R.P. (2000). "On the Nature and Direction of relationships Between Constructs and Measures," *Psychological Methods*, Vol. 5, No. 2, pp. 155-174.
- Eisenhardt, K.M. (1989). "Building Theories from Case Study Research," *Academy of Management Review*, Vol. 14, No. 4, pp. 532-550.
- Eisenhardt, K.M. & Bourgeois, L.J. (1988). "Politics of Strategic Decision Making in High-Velocity Environments: Toward a Midrange Theory," *Academy of Management Journal*, Vol. 31, No. 4, pp. 737-770.
- El Eman, K. & Birk, A. (2000). "Validating the ISO/IEC 15504 Measures of Software Development Process Capability," *Journal of Systems and Software*, Vol. 51, No. 2, pp. 119-149.
- Emery, F.E. (Ed). (1969). *Systems Thinking*. Harmondsworth: Penguin Books.
- Engstrom, E.W. (1957). "System Engineering – A Growing Concept," *Electrical Engineering*, Vol. 76, pp. 113-116.
- Espejo, R. (2004). "The Footprint of Complexity: The Embodiment of Social System," *Kybernetes*, Vol. 33, No. 3/4, pp. 671-700.
- Etzioni, A. (1968). *The Active Society: Theory of Societal and Political Processes*. New York: The Free Press.
- Farhoomand, A.F. & Drury, D.H. (1999). "A Historiographical Examination of Information Systems," *Communications of AIS*, Vol. 1, Art. 19, pp. 1-27.
- Farrell, D. & Petersen, J.C. (1982). "Patterns of Political Behavior in Organizations," *Academy of Management Review*, Vol. 7, No. 3, pp. 403-412.
- Ferguson, J. & Sheard, S. (1998). "Leveraging Your CMM Efforts for IEEE/EIA 12207," *IEEE Software*, Vol. 15, No. 5, pp. 23-28.
- Ferguson, P., Humphrey, W., Khajenoori, S., Macke, S. & Matvya, A. (1997). "Results of Applying the Personal Software Process," *IEEE Computer*, Vol. 30, No. 5, pp. 24-31.
- Firestone, W.A. (1990). "Accommodation: Toward a Paradigm-Praxis Dialectic." In E. G. Guba (Ed.). *The Paradigm Dialog*. (pp. 105-135). Newbury Park: Sage Publications.
- Firestone, W.A. (1993). "Alternative Arguments for Generalizing from Data as Applied to Qualitative Research," *Educational Researcher*, Vol. 22, No. 4, pp. 16-23.
- Fisch, M. (1985a). "Whewell's Consilience of Inductions – An Evaluation," *Philosophy of Science*, Vol. 52, No. 2, pp. 239-255.

- Fisch, M. (1985b). "Necessary and Contingent Truth in William Whewell's Antithetical Theory of Knowledge," *Studies in History and Philosophy of Science*, Vol. 16, No. 4, pp. 275-314.
- Fisch, M. (1991). *William Whewell, Philosopher of Science*. New York: Oxford University Press.
- Fitzgerald, B. & O'Kane, T. (1999). "A Longitudinal Study of Software Process Improvement," *IEEE Software*, Vol. 16, No. 3, pp. 37-45.
- Flagle, C.D., Huggins, W.H. & Roy, R.H. (1960). *Operations Research and Systems Engineering*. Baltimore: The Johns Hopkins Press.
- Flood, R.L. (1990). *Liberating Systems Theory*. New York: Plenum Press.
- Flood, R.L. & Carson, E.R. (1993). *Dealing with Complexity: An Introduction to the Theory and Application of Systems Science* (2nd ed.). New York: Plenum Press.
- Forrester, J.W. (1958). "Industrial Dynamics: A Major Breakthrough for Decision Makers," *Harvard Business Review*, Vol. 36, No. 4, pp. 37-66.
- Forrester, J.W. (1961). *Industrial Dynamics*. Cambridge: MIT Press.
- Forrester, J.W. (1969). *Urban Dynamics*. Cambridge: MIT Press.
- Forrester, J.W. (1973). *World Dynamics*. Cambridge: Wright-Allen Press.
- Frank, P.G. (1957). *Philosophy of Science: The Link between Science and Philosophy*. Englewood Cliffs, NJ: Prentice Hall.
- Franz, C.R. & Robey, D. (1984). "An Investigation of User-Led System Design: Rational and Political Perspectives," *Communications of the ACM*, Vol. 27, No. 12, pp. 1202-1209.
- Freese, L. (1980). "Formal Theorizing," *Annual Review of Sociology*, Vol. 6, pp. 187-212.
- Futrell, R., Shafer, D. & Shafer, L. (2002). *Quality Software Project Management*. Upper Saddle River: PTR Prentice Hall.
- Gallagher, C.A. (1974). "Perceptions of the Value of a Management Information System," *Academy of Management Journal*, Vol. 17, No. 1, pp. 46-55.
- Gandz, J. & Murray, V.V. (1980). "The Experience of Workplace Politics," *Academy of Management Journal*, Vol. 23, No. 2, pp. 237-251.
- van Genuchten, M. (1991). "Why is Software Late? An Empirical Study of Reasons for Delay in Software Development," *IEEE Transactions on Software Engineering*, Vol. 17, No. 6, pp. 582-590.
- Gerring, J. (2001). *Social Science Methodology: A Criterial Framework*. Cambridge: Cambridge University Press.
- Gharajedaghi, J. (1999). *Systems Thinking: Managing Chaos and Complexity*. Boston: Butterworth-Heinemann.

- Gibbs, G.R. (2002). *Qualitative Data Analysis: Explorations with NVivo*. New York: Open University Press.
- Gibbs, J.P. (1972). "Causation and Theory Construction," *Social Science Quarterly*, Vol. 52, No. 4, pp. 815-826.
- Gibbs, W.W. (1994). "Software's Chronic Crisis," *Scientific American*, Vol. 271, No. 3, pp. 86-95.
- Gibson, J. (1991). *How to Do Systems Analysis*. Prentice-Hall, Unpublished Manuscript.
- van Gigch, J.P. (1974). *Applied General Systems Theory*. New York: Harper & Row Publishers.
- Giere, R.N. (2004). "How Models are Used to Represent Reality," *Philosophy of Science*, Vol. 71, No. 5, pp. 742-752.
- Gilb, T. (1988). *Principles of Software Engineering Management*. Harlow: Addison-Wesley Longman Ltd.
- Glaser, B.G. (1978). *Theoretical Sensitivity: Advances in the Methodology of Grounded Theory*. Mill Valley: Sociology Press.
- Glaser, B.G. & Strauss, A.L. (1967). *The Discovery of Grounded Theory, Strategies for Qualitative Research*. Chicago: Aldine Publishers.
- Goode, H. & Machol R. (1957). *Systems Engineering: An Introduction to the Design of Large-Scale Systems*. New York: McGraw-Hill Book Company.
- Grojean, C.A. (2005). "Microsoft's IT Organization Uses PSP/TSP to Achieve Engineering Excellence," *Crosstalk*, Vol. 18, No. 3, pp. 8-12.
- Guba, E.G. & Lincoln, Y.S. (1994). "Competing Paradigms in Qualitative Research," in Denzin, N.K. & Lincoln, Y.S. (Eds.), *Handbook of Qualitative Research*, Newbury Hill: Sage Publications.
- Habermas, J. (1984). *The Theory of Communicative Action--Reason and the Rationalization of Society*. (Vol. I). Boston, MA: Beacon Press.
- Haley, T.J. (1996). "Software Process Improvement At Raytheon," *IEEE Software*, Vol. 13, No. 6, pp. 33-41.
- Hall, A. (1962). *A Methodology for Systems Engineering*. Princeton: D. Van Nostrand Company, Inc.
- Hammond, D. (2002). "Exploring the Genealogy of Systems Thinking," *Systems Research and Behavioral Science*, Vol. 19, No. 5, pp. 429-439.
- Hammond, D. (2003). *The Synthesis of Science: Exploring the Social Implications of General Systems Theory*. Boulder: University Press of Colorado.
- Hardgrave, B.C., Davis, F.D. & Riemenschneider, C.K. (2003). "Investigating Determinants of Software Developers' Intentions to Follow Methodologies," *Journal of Management Information Systems*, Vol. 20, No. 1, pp. 123-151.

- Harter, D.E & Slaughter, S.A. (2003). "Quality Improvement and Infrastructure Activity Costs in Software Development: A Longitudinal Analysis," *Management Science*, Vol. 49, No. 6, pp. 784-800.
- Harter, D.E., Krishnan, M.S. & Slaughter, S.A. (2000). "Effects of Process Maturity on Quality, Cycle Time, and Effort in Software Product Development," *Management Science*, Vol. 46, No. 4, pp. 451-466.
- Hartman, F. & Ashrafi, R.A. (2002). "Project Management in the Information Systems and Information Technologies Industries," *Project Management Journal*, Vol. 33, No.3, pp. 5-15.
- Hempel, C.G. & Oppenheim, P. (1948). "Studies in the Logic of Explanation," *Philosophy of Science*, Vol. 15, No. 2, pp. 135-175.
- Hinkin, T. (1995). "A Review of Scale Development Practices in the Study of Organizations," *Journal of Management*, Vol. 21, No. 5, pp. 967-988.
- Hirschheim, R., Klein, H.K. & Lyytinen, K. (1996). "Exploring the Intellectual Foundations of Information Systems," *Accounting, Management and Information Technologies*, Vol. 6, No. 1/2, pp. 1-64.
- Hitch, C. (1953). "Sub-optimization in Operations Problems," *Journal of the Operational Research Society of America*, Vol. 1, No. 3, pp. 87-99.
- Holland, J.H., Holyoak, K.J., Nisbett, R.E. & Thagard, P.R. (1986). *Induction: Processes of Inference, Learning, and Discovery*. Cambridge: MIT Press.
- Hughes, A.C. & Hughes, T.P. (Eds.) (2000). *Systems, Experts, and Computers: The Systems Approach in Management and Engineering, World War II and After*. Cambridge: MIT Press.
- Humphrey, W. (1988). "Characterizing the Software Process: A Maturity Framework," *IEEE Software*, Vol. 5, No. 2, pp. 73-79.
- Humphrey, W. (1989). *Managing the Software Process*. Reading, Massachusetts: Addison-Wesley Publishing Company.
- Humphrey, W. (1995). *A Discipline for Software Engineering*. Reading, Massachusetts: Addison-Wesley Publishing Company.
- Humphrey, W. (1996a). *Introduction to the Personal Software ProcessSM*. Reading, Massachusetts: Addison-Wesley Publishing Company.
- Humphrey, W. (1996b). "Using A Defined and Measured Personal Software Process," *IEEE Software*, Vol. 13, No. 3, pp. 77-88.
- Humphrey, W. (2000). *Introduction to the Team Software Process*. Reading, Massachusetts: Addison-Wesley Publishing Company.
- Humphrey, W. (2002). *Winning with Software: An Executive Strategy*. Boston: Addison-Wesley Publishing Company.
- Humphrey, W., Snyder, T.R. & Willis, R.R. (1991). "Software Process Improvement at Hughes Aircraft," *IEEE Software*, Vol. 8, No. 4, pp. 11-23.

- Ibbs, C.W. & Kwak, Y.H. (2000). "Assessing Project Management Maturity," *Project Management Journal*, Vol. 31, No. 1, pp. 32-43.
- Ibrahim, L. & Weszka, J. (2004). "There is More to Process Improvement than Just CMM," *Crosstalk*, Vol. 17, No. 6, pp. 11-15.
- IEEE 610.12 (1990). *IEEE Standard 610.12: IEEE Standard Glossary of Software Engineering Terminology*. New York: The Institute of Electrical and Electronics Engineers.
- IEEE 1220. (1998). *IEEE Standard 1220: Application and Management of the Systems Engineering Process*. New York: The Institute of Electrical and Electronics Engineers.
- Iivari, J. (1991). "A Paradigmatic Analysis of Contemporary Schools of IS Development," *European Journal of Information Systems*, Vol. 1, No. 4, pp. 249-272.
- Iivari, J. & Hirschheim, R. (1996). "Analyzing Information Systems Development: A Comparison of Eight IS Development Approaches," *Information Systems*, Vol. 21, No. 7, pp. 551-575.
- Iivari, J., Hirschheim, R. & Klein, H.K. (1998). "A Paradigmatic Analysis Contrasting Information Systems Development Approaches and Methodologies," *Information Systems Research*, Vol. 9, No. 2, pp. 164-193.
- Ives, B. & Olson, M.H. (1981). "User Involvement and MIS Success: A Review of Research," *Management Science*, Vol. 30, No. 5, pp. 586-603.
- International Council on Systems Engineering (INCOSE). (2004, June) *Systems Engineering Handbook* (INCOSE-TP-2003-016-02 Version 2a) Seattle, WA: INCOSE Technical Board.
- ISO/IEC 12207 (1995). *ISO/IEC Standard 12207: Information Technology – Software Lifecycle Processes*. Geneva: International Organization for Standardization and International Electrotechnical Commission.
- ISO/IEC 15504 (2004). *ISO/IEC Standard 15504: Information Technology – Process Assessment*. Geneva: International Organization for Standardization and International Electrotechnical Commission.
- ISO/IEC 90003 (2004) *ISO/IEC Standard 90003: Software Engineering – Guidelines for the Application of ISO 9001 to Computer Software*. Geneva: International Organization for Standardization and International Electrotechnical Commission.
- Jackson, M. (1990). "Beyond a System of Systems Methodologies," *Journal of the Operational Research Society*, Vol. 41, No. 8, pp. 657-668.
- Jackson, M.C. (1991). *Systems Methodology for the Management Sciences*. New York: Plenum Press.
- Jackson, M. & Keys, P. (1984). "Towards a System of Systems Methodologies," *Journal of the Operational Research Society*, Vol. 35, No. 6, pp. 473-486.

- Jacobs, D.A., Keating, C.B. & Fernandez, A.A. (2000). "Team-Based Methods to Analyze High-Technology Production Systems," *Engineering Management Journal*, Vol. 12, No. 1, pp. 15-22.
- Jacobson, I., Booch, G. & Rumbaugh, J. (1999). "The Unified Process," *IEEE Software*, Vol. 16, No. 3, pp. 96-102.
- Jarke, M., Mylopoulos, J., Schmidt, J.W. & Vassiliou, Y. (1992). "DAIDA: An Environment for Evolving Information Systems," *ACM Transactions on Information Systems*, Vol. 10, No. 1, pp. 1-50.
- Jensen, R.W. & Tonies, C.C. (1978). *Software Engineering*. Englewood Cliffs: Prentice-Hall.
- Jevons, W.S. (1913). *The Principles of Science: A Treatise on Logic and Scientific Method*. (2nd ed.). London: MacMillan and Company. (Original work published 1877).
- Jiang, J.J., Klein, G. & Discenza, R. (2002a). "Perception Differences of Software Success: Provider and User Views of System Metrics," *Journal of Systems and Software*, Vol. 63, No. 1, pp. 17-27.
- Jiang, J.J., Klein, G. & Discenza, R. (2002b). "Pre-project Partnering Impact on an Information System Project, Project Team and Project Manager," *European Journal of Information Systems*, Vol. 11, No. 2, pp. 86-97.
- Jiang, J.J., Klein, G., Hwang, H., Huang, J. & Hung, S. (2004). "An Exploration of the Relationship between Software Development Process Maturity and Project Performance," *Information & Management*, Vol. 41, No. 3, pp. 279-288.
- Jick, T.D. (1979). "Mixing Qualitative and Quantitative Methods: Triangulation in Action," *Administrative Science Quarterly*, Vol. 24, No. 4, pp. 602-611.
- Jones, C. (1995). "Patterns of Large Software Systems: Failure and Success," *IEEE Computer*, Vol. 28, No. 3, pp. 86-87.
- Jones, C. (2004). "Software Project Management Practices: Failure versus Success," *Crosstalk*, Vol. 17, No. 10, pp. 5-9.
- Kahneman, D. (2003a). "A Perspective on judgment and Choice: Mapping Bounded Rationality," *American Psychologist*, Vol. 58, No. 9, pp. 697-720.
- Kahneman, D. (2003b). "Maps of Bounded Rationality: Psychology for Behavioral Economics," *The American Economic Review*, Vol. 93, No. 5, pp. 1449-1475.
- Kahneman, D. & Tversky, A. (1979). "Prospect Theory: An Analysis of Decision under Risk," *Econometrica*, Vol. 47, No. 2, pp. 263-292.
- Kahneman, D., Slovic, P. & Tversky, A. (Eds.) (1982). *Judgment under Uncertainty: Heuristics and Biases*. New York: Cambridge University Press.
- Kaplan, A. (1964). *The Conduct of Inquiry: Methodology for Behavioral Science*. San Francisco: Chandler Publishing Company.

- Kaplan, B. & Duchon, D. (1988). "Combining Qualitative and Quantitative Methods in Information Systems Research: A Case Study," *MIS Quarterly*, Vol. 12, No. 4, pp. 571-586.
- Karlsen, J.T. (2002). "Project Stakeholder Management," *Engineering Management Journal*, Vol. 14, No. 4, pp. 19-24.
- Karlsen, J.T., Gottshalk, P. & Andersen, E.S. (2002). "External or Internal Focus? A Comparison of IT Executives and IT Project Manager Roles," *Engineering Management Journal*, Vol. 14, No. 2, pp. 5-11.
- Kaufmann, F. (1942). "The Logical Rules of Scientific Procedure," *Philosophical and Phenomenological Research*, Vol. 2, No. 4, pp. 457-471.
- Keating, C., Kauffmann, P. & Dryer, D. (2001). "A Framework for Systemic Analysis of Complex Issues," *Journal of Management Development*, Vol. 20, No. 9/10, pp. 772-784.
- Keating, C. & Varela, M. (2003). *Project Management Systems*, Unpublished manuscript, Old Dominion University.
- Keil, M., Mann, J. & Rai, A. (2000). "Why Software Projects Escalate: An Empirical Analysis and Test of Four Theoretical Models," *MIS Quarterly*, Vol. 24, No. 4, pp. 631-664.
- Keil, M. & Robey, D. (1999). "Turning Around Troubled Software Projects: An Exploratory Study of the Deescalation of Commitment to Failing Courses of Action," *Journal of Management Information Systems*, Vol. 15, No. 4, pp. 63-87.
- Kelle, U. & Laurie, H. (1995). "Computer Use in Qualitative Research and Issues of Validity." In U. Kelle (Ed.). *Computer-Aided Qualitative Data Analysis: Theory Methods and Practice*. (pp. 19-28). Thousand Oaks: Sage Publications.
- Kelley, H.H. & Thibault, J.W. (1978). *Interpersonal Relations: A Theory of Interdependence*. New York: John Wiley & Sons.
- Kemeny, J.G. (1959). *A Philosopher Looks at Science*. Princeton, NJ: D. Van Nostrand Company.
- Kendra, K. & Taplin, L.J. (2004). "Project Success: A Cultural Framework," *Project Management Journal*, Vol. 35, No. 1, pp. 30-45.
- Kerlinger, F.N. & Lee, H.B. (2000). *Foundations of Behavioral Research* (4th ed.). Fort Worth: Harcourt College Publishers.
- King, W.R. & Rodriguez, J.I. (1981). "Participative Design of Strategic Decision Support Systems: An Empirical Assessment," *Management Science*, Vol. 27, No. 6, pp. 717-726.
- King, G., Keohane, R.O. & Verba, S. (1994). *Designing Social Inquiry: Scientific Inference in Qualitative Research*. Princeton: Princeton University Press.
- Klein, H.K. & Myers, M.D. (1999). "A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems," *MIS Quarterly*, Vol. 23, No. 1, pp. 67-93.

- Klir, G. (2001). *Facets of Systems Science* (2nd ed.). New York: Kluwer Academic/Plenum Publishers.
- Kolence, K.W. (1973). "The Software Empiricist," *ACM SIGMETRICS Performance Evaluation Review*, Vol. 2, No. 2, pp. 31-36.
- Kolence, K.W. & Kiviat, P.J. (1973). "Software Unit Profiles & Kiviat Figures," *ACM SIGMETRICS Performance Evaluation Review*, Vol. 2, No. 3, pp. 2-12.
- Kossiakoff, A. & Sweet, W. (2003). *Systems Engineering Principles and Practice*. Hoboken: John Wiley and Sons.
- Kuhn, T.S. (1996). *The Structure of Scientific Revolutions*. (3rd ed.). Chicago: The University of Chicago Press.
- Landry, M. & Banville, C. (1992). "A Discipline Methodological Pluralism for MIS Research," *Accounting, Management & Information Technology*, Vol. 2, No. 2, pp. 77-97.
- Laudan, L. (1981). *Science and Hypothesis: Historical Essays on Scientific Methodology*. Dordrecht Holland: D. Reidel Publishing Company.
- Lawshe, C.H. (1975). "A Quantitative Approach to Content Validity," *Personnel Psychology*, Vol. 28, No. 4, pp. 563-575.
- Lawson, H.W. (1990). "Philosophies for Engineering Computer-Based Systems," *IEEE Computer*, Vol. 23, No. 12, pp. 52-63.
- LeCompte, M.D. & Goetz, J.P. (1982). "Problems of Reliability and Validity in Ethnographic research," *Review of Educational Research*, Vol. 52, No. 1, pp. 31-60.
- Lee, A.S. (1989a). "A Scientific Methodology for MIS Case Studies," *MIS Quarterly*, Vol. 13, No. 1, pp. 33-50.
- Lee, A.S. (1989b). "Case Studies as Natural Experiments," *Human Relations*, Vol. 42, No. 2, pp. 117-137.
- Lee, A.S. & Baskerville, R.L. (2003). "Generalizing Generalizability in Information Systems Research," *Information Systems Research*, Vol. 14, No. 3, pp. 221-243.
- Leedy, P.D. & Ormrod, J.E. (2001). *Practical Research: Planning and Design* (7th ed.). Upper Saddle River: Prentice-Hall.
- Leonard-Barton, D.A. (1990). "A Dual Methodology for Case Studies: Synergistic Use of a Longitudinal Single Site with Replicated Multiple Sites," *Organization Science*, Vol. 1, No. 3, pp. 248-266.
- Levine, H.G. & Rossmore, D. (1995). "Politics and the Function of Power in a Case Study of IT Implementation," *Journal of Management Information Systems*, Vol. 11, No. 3, pp. 115-133.
- Lewis, M.E. & Grimes, A.J. (1999). "Metatriangulation: Building Theory from Multiple Paradigms," *Academy of Management Review*, Vol. 24, No. 4, pp. 672-690.
- Lieblein, E. (1986). "The Department of Defense Software Initiative – A Status Report," *Communications of the ACM*, Vol. 29, No. 8, pp. 734-744.

- Lijphart, A. (1971). "Comparative Politics and the Comparative Method," *American Political Science Review*, Vol. 65, No. 3, pp. 682-693.
- Likert, R.A. (1932). "A Technique for the Measurement of Attitudes," *Archives of Psychology*, Vol. 23, No. 140, pp. 1-55.
- Linberg, K.R. (1999). "Software Developer Perceptions about Software Project Failure: A Case Study," *Journal of Systems and Software*, Vol. 49, No. 2/3, pp. 177-192.
- Lincoln, Y.S. & Guba, E.G. (1985). *Naturalistic Inquiry*. Newbury Park: Sage Publications.
- Lincoln, Y.S. & Guba, E.G. (2000). "Paradigmatic Controversies, Contradictions, and Emerging Confluences." In N.K. Denzin & Y.S. Lincoln (Eds.). *Handbook of Qualitative Research* (2nd ed.). (pp. 163-188). Thousand Oaks: Sage Publications.
- Lissitz, R.W. & Green, S.B. (1975). "Effect of the Number of Scale Points on Reliability: A Monte Carlo Approach," *Journal of Applied Psychology*, Vol. 60, No. 1, pp. 10-13.
- Lovitts, B.E. (2005). "How to Grade a Dissertation," *Academe*, Vol. 91, No. 6, pp. 18-23.
- Machol, R. (Ed.). (1965). *Systems Engineering Handbook*. New York: McGraw-Hill Book Company.
- MacKenzie, K.D. (1976). *A Theory of Group Structures, Vol. 1*. New York: Gordon & Breach.
- Mann, C. (2002). "Why Software is So Bad," *Technology Review*. Vol. 105, No. 6, pp. 33-38.
- Manzoni, L.V. & Price, R.T. (2003). "Identifying Extensions Required by RUP (Rational Unified Process) to Comply with CMM (Capability Maturity Model) Levels 2 and 3," *IEEE Transactions on Software Engineering*, Vol. 29, No. 2, pp. 181-192.
- Markus, M.L. (1983). "Power, Politics, and MIS Implementation," *Communications of the ACM*, Vol. 26, No. 6, pp. 430-444.
- Markus, M.L. & Robey, D. (1983). "The Organizational Validity of Management Information Systems," *Human Relations*, Vol. 36, No. 3, pp. 203-226.
- Mathieu, R.G. (2002). "Top-Down Approach to Computing," *IEEE Computer*, Vol. 35, No. 1, pp. 138-139.
- Maxwell, G. (1962). "Theories, Frameworks, and Ontology," *Philosophy of Science*, Vol. 29, No. 2, pp. 132-138.
- Mayes, B.T. & Allen, R.W. (1977). "Toward a Definition of Organizational Politics," *Academy of Management Review*, Vol. 2, No. 4, pp. 672-678.
- McCabe, T.J. (1976). "A Complexity Measure," *IEEE Transactions on Software Engineering*, Vol. SE2, No. 4, pp. 308-320.
- McCabe, T.J. & Butler, C.W. (1994). "Design Complexity Measurement and Testing," *Communications of the ACM*, Vol. 32, No. 12, pp. 1415-1425.
- McGarry, F. & Decker, W. (2002). "Attaining Level 5 in CMM Process Maturity," *IEEE Software*, Vol. 19, No. 6, pp. 87-96.

- McKeen, J.D. & Guimaraes, T. (1997). "Successful Strategies for User Participation in Systems Development," *Journal of Management Information Systems*, Vol. 14, No. 2, pp. 133-150.
- Merton, R.K. (1968). *Social Theory and Social Structure*. New York: The Free Press.
- Merton, R.K., Fiske, M. & Kendall, P.L. (1990). *The Focused Interview: A Manual of Problems and Procedures*. (2nd ed.). New York: Free Press.
- Meyer, M.A. & Booker, J.M. (2001). *Eliciting and Analyzing Expert Judgment: A Practical Guide*. Philadelphia: Society for Industrial and Applied Mathematics.
- Mihm, J., Loch, C. & Huchzermeier, A. (2003). "Problem-Solving Oscillations in Complex Engineering Projects," *Management Science*, Vol. 49, No. 6, pp. 733-750.
- Miles, M.B. & Huberman, A.M. (1994). *Qualitative Data Analysis: An Expanded Sourcebook*. Thousand Oaks: Sage Publications.
- Mingers, J. (2001). "Combining IS Research Methods: Towards a Pluralist Methodology," *Information Systems Research*, Vol. 12, No. 3, pp. 240-259.
- Minnich, I. (2002). "EIA IS 731 Compared to CMMI-SE/SW," *Systems Engineering*, Vol. 5, No. 1, pp. 62-72.
- Mintzberg, H. (1979). "An Emerging Strategy of 'Direct' Research," *Administrative Sciences Quarterly*, Vol. 24, No. 4, pp. 582-589.
- Mishler, E.G. (1979). "Meaning in Context: Is There any Other Kind?" *Harvard Educational Review*, Vol. 49, No. 1 pp. 1-19.
- Moore, J.W. (2006). "Converging Software and Systems Engineering Standards," *IEEE Computer*, Vol. 39, No. 9, pp. 106-108.
- Morgan, G. & Smircich, L. (1980). "The Case for Qualitative Research," *Academy of Management Review*, Vol. 5, No. 4, pp. 491-500.
- Moser, C.A. & Kalton, G. (1972). *Survey Methods in Social Investigation*. New York: Basic Books.
- Mullins, N.C. (1974). "Theory Construction from Available materials: A System for Organizing and Presenting Propositions," *American Journal of Sociology*, Vol. 80, No. 1, pp. 1-15.
- Munck, G.L. (1998). "Canons of Research Design in Qualitative Analysis," *Studies in Comparative International Development*, Vol. 33, No. 3, pp. 18-45.
- Murugappan, M. & Keeni, G. (2003). "Blending CMM and Six Sigma to Meet Business Goals," *IEEE Software*, Vol. 20, No. 2, pp. 42-48.
- Mylopoulos, J. (1998). "Information Modeling in the Time of the Revolution," *Information Systems*, Vol. 23, No. 3-4, pp. 127-155.
- Nagel, E. (1951). *The Structure of Science: Problems in the Logic of Scientific Explanation*. New York: Harcourt, Brace & World.

- Nakamori, Y. & Sawaragi, Y. (2000). "Complex Systems Analysis and Environmental Modeling," *European Journal of Operational Research*, Vol. 122, No. 2, pp. 178-189.
- Naur, P. & Randell, B. (Eds.) (1969). Report on a conference sponsored by the NATO Science Committee at Garmisch, Germany, October 7-11, 1968. Brussels: North Atlantic Treaty Organization.
- NAVSEA (2007). *Submarine Safety Requirements Manual*, NAVSEA 0924-062-0010. Washington: Naval Sea Systems Command.
- Nguyen, T.N. (2006). "A Decision Model for Managing Software Development Projects," *Information & Management*, Vol. 43, No. 1, pp. 63-75.
- Niazi, M., Wilson, D. & Zowghi, D. (2005a). "A Maturity Model for the Implementation of Software Process Improvement: An Empirical Study," *Journal of Systems and Software*, Vol. 74, No. 2, pp. 155-172.
- Niazi, M., Wilson, D. & Zowghi, D. (2005b). "A Framework for Assisting the Design of Effective Software Process Improvement Implementation Strategies," *Journal of Systems and Software*, Vol. 78, No. 2, pp. 204-222.
- Nidumolu, S.R. & Subramani, M.R. (2004). "The Matrix of Control: Combining Process and Structure Approaches to Managing Software Development," *Journal of Management Information Systems*, Vol. 20, No. 3, pp. 159-196.
- Nunnally, J.C. (1967). *Psychometric Theory* (3rd ed.). New York: McGraw-Hill
- Orlikowski, W.J. & Baroudi, J.J. (1991). "Studying Information Technology in Organizations: Research Approaches and Assumptions," *Information Systems Research*, Vol. 2, No. 1, pp. 1-28.
- Patton, E. & Appelbaum, S.H. (2003). "The Case for Case Studies in Management Research," *Management Research News*, Vol. 26, No. 5, pp. 60-71.
- Patton, M.Q. (1987). *How to Use Qualitative Methods in Evaluation*. Newbury Park: Sage Publications.
- Patton, M.Q. (2002). *Qualitative Research & Evaluation Methods*. (3rd ed.). Thousand Oaks: Sage Publications.
- Paulk, M. (1995). "How ISO 9001 Compares with the CMM," *IEEE Software*, Vol. 12, No. 1, pp. 74-83.
- Paulk, M.C., Curtis, B., Chrissis, M.B. & Weber, C.V. (1993). "Capability Maturity Model, Version 1.1," *IEEE Software*, Vol. 10, No. 4, pp. 18-27.
- Pemberton, M.A. (1993). "Modeling Theory and Composing Process Models," *College Composition and Communication*, Vol. 44, No. 1, pp. 40-58.
- Petkova, O. & Petkov, D. (2003). "A Holistic Approach Towards the Validation and Legitimation of Information Systems," *Kybernetes*, Vol. 32, No. 5/6, pp. 703-714.
- Pfleeger, S. (1998). *Software Engineering: Theory and Practice*. Upper Saddle River, NJ: Prentice-Hall.

- Phillips, D. (1997). *The Software Project Manager's Handbook: Principles that Work at Work*. Los Alamitos: IEEE Computer Society Press.
- Phillips, M. (1941). "Problems of Questionnaire Investigation," *The Research Quarterly of the American Association for Health, Physical Education, and Recreation*, Vol. 12, No. 3, pp. 528-537.
- van der Pijl, G.J., Swinkels, G.J.P. & Verrijdt, J.G. (1997). "ISO 9000 versus CMM: Standardization and Certification of IS Development," *Information & Management*, Vol. 32, No. 6, pp. 267-274.
- Pinto, J.K. (2000). "Understanding the Role of Politics in Successful Project Management," *International Journal of Project Management*, Vol. 18, No.2, pp. 85-91.
- Pinto, J.K. & Slevin, D.P. (1988). "Project Success: Definitions and Measurement Techniques," *Project Management Journal*, Vol. 19, No. 1, pp. 67-72.
- PMI (2004). *A Guide to the Project Management Body of Knowledge* (3rd ed.). Newtown Square: Project Management Institute.
- Popper, K. (1959). *The Logic of Scientific Discovery*. London: Routledge.
- Potts, C. (1993). "Software-Engineering Research Revisited," *IEEE Software*, Vol. 10, No. 5, pp. 19-28.
- Pressman, R. (2001). *Software Engineering: A Practitioner's Approach* (5th ed.). New York: McGraw-Hill.
- Purvis, R.L., McCray, G.E. & Roberts, T. (2004). "Heuristics and Biases in Information Systems Project Management," *Engineering Management Journal*, Vol. 16, No 2, pp. 19-27.
- Quade, E.S. & Miser, H.J. (Eds.) (1985). *Handbook of Systems Analysis: Overview of Uses, Procedures, Applications, and Practice*. New York: North-Holland.
- Ramsay, D.A., Boardman, J.T. & Cole, A.J. (1996). "Reinforcing Learning, Using Soft Systemic Frameworks," *International Journal of Project Management*, Vol. 14, No. 1, pp. 31-36.
- RAND (1952). *Suboptimization in Operations Problems (RAND Report P-326)*. Santa Monica: Charles J. Hitch.
- Rautiainen, K., Lassenius, C. & Sulonen, R. (2002). "4CC: A Framework for Managing Software Product Development," *Engineering Management Journal*, Vol. 14, No. 2, pp. 27-32.
- Reichelt, K. & Lyneis, J. (1999). "The Dynamics of Project Performance: Benchmarking the Drivers of Cost and Schedule Overrun," *European Management Journal*, Vol.17, No. 2, pp. 135-150.
- Reichenbach, H. (1936). "Induction and Probability," *Philosophy of Science*, Vol. 3, No. 1, pp. 124-126.
- Reichenbach, H. (1938). "On Probability and Induction," *Philosophy of Science*, Vol. 5, No. 1, pp. 21-45.

- Reichenbach, H. (1944). *Philosophic Foundations of Quantum Mechanics*. Berkeley: University of California Press.
- Reichenbach, H. (1951). *The Rise of Scientific Philosophy*. Berkeley: University of California Press.
- Reifer, D. (Ed.) (2002). *Software Management*. New York: Wiley-IEEE Computer Society Press.
- Richards, L. (1999). *Using NVivo in Qualitative Research*. London: Sage Publications.
- Ríos, J.P. (2004). "A Self-organizing Network for the Systems Community," *Kybernetes*, Vol. 33, No. 3/4, pp. 590-606.
- Roberts, T.L., Gibson, M.L., Fields, K.T. & Rainer, R.K. (1998). "Factors that Impact Implementing a System Development Methodology," *IEEE Transactions on Software Engineering*, Vol. 24, No. 8, pp. 640-649.
- Robey, D. & Farrow, D.L. (1982). "User Involvement in Information System Development: A Conflict Model and Empirical Test," *Management Science*, Vol. 28, No. 1, pp. 73-85.
- Robey, D., Farrow, D.L. & Franz, C.R. (1989). "Group Process and Conflict in System Development," *Management Science*, Vol. 35, No. 10, pp. 1172-1191.
- Robey, D. & Markus, M.L. (1984). "Rituals in Information System Design," *MIS Quarterly*, Vol. 8, No. 1, pp. 5-15.
- Robey, D., Smith, L.A. & Vijayasarathy, L.R. (1993). "Perceptions of Conflict and Success in Information Systems Development Projects," *Journal of Management Information Systems*, Vol. 10, No. 1, pp.123-139.
- Robinson, W.S. (1951). "The Logical Structure of Analytic Induction," *American Sociological Review*, Vol. 16, No. 6, pp. 812-818.
- Rodrigues, A. & Williams, T. (1997). "System Dynamics in Software Project Management: Towards the development of a Formal Integrated Framework," *European Journal of Information Systems*, Vol. 6, No. 1, pp. 51-66.
- Roy, R.H. (1960). "The Development and Future of Operations Research and Systems Engineering." In C.D. Flagle, W.M. Huggins & R.H. Roy (Eds.), *Operations Research and Systems Engineering* (pp. 8-27). Baltimore: The Johns Hopkins Press.
- Royce, W. (1998). *Software Engineering: A Unified Approach*. Boston: Addison-Wesley.
- Rozenblit, J.W. & Kumar, S. (1997). "Toward Synergistic Engineering of Computer Systems," *IEEE Computer*, Vol. 30, No. 2, pp. 126-127.
- Rubin, H.J. & Rubin, I.S. (1995). *Qualitative Interviewing: The Art of Hearing Data*. Thousand Oaks: Sage Publications.
- Ruiz, M., Ramos, I. & Toro, M. (2001). "A Simplified Model of Software Project Dynamics," *The Journal of Systems and Software*, Vol. 59, No. 1, pp. 299-309.
- Runkel, P.J. & Runkel, M. (1984). *A Guide to Usage for Writers and Students in the Social Sciences*. Totowa, NJ: Rowman and Allanheld.

- Sage, A.P. (1995). *Systems Management for Information Technology and Software Engineering*. New York: John Wiley & Sons.
- Sage, A.P. & Palmer, J.D. (1990). *Software Systems Engineering*. New York: John Wiley & Sons.
- Saiedian, H. & Chennupati, K. (1999). "Towards an Evaluative Framework for Software Process Improvement Models," *Journal of Systems and Software*, Vol. 47, No. 2-3, pp. 139-148.
- Schaaf, R. (2005). "What's All This about Systems?" *IEEE Computer*, Vol. 38, No. 6, pp. 104, 102-103.
- Schein, E.H. (1992). *Organizational Culture and Leadership*. (2nd ed.). San Francisco: Jossey-Bass.
- Schwalbe, K. (2002). *Information Technology Project Management* (2nd ed.). Canada: Course Technology Thomson Learning.
- Seaman, Carolyn B. (1999). "Qualitative Methods in Empirical Studies of Software Engineering," *IEEE Transactions on Software Engineering*, Vol. 25, No. 4, pp. 557-572.
- SEI (2002). Capability Maturity Model[®] Integration (CMMISM), Staged Representation (CMU/SEI Technical Report 2002-TR-029). Pittsburgh: Carnegie Mellon University, Software Engineering Institute.
- SEI (2003). *Demonstrating the Impact and Benefits of CMMI[®]: An Update and Preliminary Results* (SEI/CMU Special Report 2003-SR-009). Pittsburgh: Dennis R. Goldenson & Diane L. Gibson.
- SEI (2005). *Capability Maturity Model[®] Integration (CMMI[®]) Overview*. Pittsburgh: Carnegie Mellon University, Software Engineering Institute.
- Seidel, J. & Kelle, U. (1995). "Different Functions of Coding in the Analysis of textual Data." In U. Kelle (Ed.). *Computer-Aided Qualitative Data Analysis: Theory Methods and Practice*. (pp. 52-61). Thousand Oaks: Sage Publications.
- Selye, H. (1964). *From Dream to Discovery: On Being a Scientist*. New York: McGraw-Hill Book Company.
- Sengupta, K. & Abdel-Hamid, T.K. (1996). "The Impact of Unreliable Information on the Management of Software Projects: A Dynamic Decision Perspective," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 26, No. 2, pp. 171-189.
- Shani, A.B. & Sena, J.A. (1994). "Information Technology and the Integration of Change: Sociotechnical System Approach," *Journal of Applied Behavioral Science*, Vol. 30, No. 2, pp. 247-270.
- Shani, A.B., Grant, R.M., Krishnan, R. & Thompson, E. (1992). "Advanced Manufacturing Systems and Organizational Choice: Sociotechnical System Approach," *California Management Review*, Vol. 34, No. 4, pp. 91-111.
- Sheard, S.A. (2001). "Evolution of the Framework's Quagmire," *IEEE Computer*, Vol. 34, No. 7, pp. 96-98.

- Sheard, S.A. (2002). "How Do I Make My Organization Comply with Yet Another New Model?" *Crosstalk*, Vol. 15, No. 2, pp. 15-20.
- Shenhav, A.J. & Dvir, D. (1996). "Toward a Typological Theory of Project Management," *Research Policy*, Vol. 25, No. 4, pp. 607-632.
- Shenhav, A.J., Dvir, D., Milosevic, D., Mulenburg, J., Patanakul, P., Reilly, R., Ryan, M., Sage, A., Sauser, B., Srivannaboon, S., Stefanovic, J. & Thamhain, H. (2005). "Toward a NASA-Specific Project Management Framework," *Engineering Management Journal*, Vol. 17, No. 4, pp. 8-16.
- Sillince, J.A.A. & Mouakket, S. (1997). "Varieties of Political Processes during Systems Development," *Information Systems Research*, Vol. 8, No. 4, pp. 368-397.
- Simon, H.A. (1955). "A Behavioral Model of Rational Choice," *Quarterly Journal of Economics*, Vol. 69, No. 1, pp. 99-118.
- Simon, H.A. (1956). "Rational Choice and the Structure of the Environment," *The Psychological Review*, Vol. 63, No. 2, pp. 129-138.
- Simon, H.A. (1962). "The Architecture of Complexity," *Proceedings of the American Philosophical Society*, Vol. 106, No. 6, pp. 467-482.
- Simon, H.A. (1979). "Rational Decision Making in Business Organizations," *The American Economic Review*, Vol. 69, No. 4, pp. 483-513.
- Singer, J. & Vinson, N.G. (2002). "Ethical Issues in Empirical Studies of Software Engineering," *IEEE Transactions on Software Engineering*, Vol. 28, No. 12, pp. 1171-1180.
- Skyttner, L. (2001). *General Systems Theory: Ideas and Applications*. Singapore: World Scientific.
- Smith, H.W. (1975). *Strategies of Social Research: The Methodological Imagination*. Englewood Cliffs: Prentice Hall.
- Smuts, J. (1926). *Holism and Evolution*. New York: Greenwood Press.
- Snyder, L.J. (1994). "It's All Necessarily So: William Whewell on Scientific Truth," *Studies in History and Philosophy of Science*, Vol. 25, No. 5, pp. 785-807.
- Snyder, L.J. (1997a). "Discoverers' Induction," *Philosophy of Science*, Vol. 64, No. 4, pp. 580-604.
- Snyder, L.J. (1997b). "The Mill-Whewell Debate: Much Ado about Induction," *Perspectives on Science*, Vol. 5, No. 2, pp. 159-198.
- Snyder, L.J. (1999). "Renovating the *Novum Organum*: Bacon, Whewell and Induction," *Studies in History and Philosophy of Science*, Vol. 30, No. 4, pp. 531-557.
- Snyder, L.J. (2006). *Reforming Philosophy: A Victorian Debate on Science and Society*. Chicago: University of Chicago Press.
- Sommer, S.C. & Loch, C.H. (2004). "Selectionism and Learning in Projects with Complexity and Unforeseeable Uncertainty," *Management Science*, Vol. 50, No. 10, pp. 1334-1347.

- Sommerville, I. (2005) *Software Engineering* (7th ed.) New York: Addison-Wesley.
- Stake, R.E. (1995). *The Art of Case Study Research*. Thousand Oaks: Sage Publications.
- Standish (2003). *The 2002 Chaos Research Study*. Yarmouth: The Standish Group.
- Stenbacka, C. (2001). "Qualitative Research Requires Quality Concepts of Its Own," *Management Decision*, Vol. 39, No. 7, pp. 551-555.
- Stevens, S.S. (1946). "On the Theory of Scales of Measurement," *Science*, Vol. 103, No. 2684, pp. 677-680.
- Strauss, A.L. & Corbin, J. (1998). *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. (2nd ed.). Thousand Oaks: Sage Publications.
- Strong, E.W. (1955). "William Whewell and John Stuart Mill: Their Controversy about Scientific Knowledge," *Journal of the History of Ideas*, Vol. 16, No. 2, pp. 209-231.
- Sutherland, J.W. (1973). *A General Systems Philosophy for the Social and Behavioral Sciences*. New York: George Braziller.
- Sushil. (2002). "Physical System Theory: Fundamentals, Recent Developments and Relationships with System Dynamics," *Kybernetes*, Vol. 31, No. 3/4, pp. 496-528.
- Sussman, S.W. & Guinan, J. (1999). "Antidotes for High Complexity and Ambiguity in Software Development," *Information & Management*, Vol. 36, No. 1, pp. 23-35.
- Tait, P. & Vessey, I. (1988). "The Effect of User Involvement on System Success: A Contingency Approach," *MIS Quarterly*, Vol. 12, No. 1, pp. 91-108.
- Thayer, R.H. (1979). Modeling a Software Engineering Project Management System (Doctoral Dissertation, University of California, Santa Barbara, 1979).
- Thayer, R.H. (Ed.). (1997). *Software Engineering Project Management* (2nd ed.). Los Alamitos: IEEE Computer Society Press.
- Thayer, R.H. (2002). "Software System Engineering: A Tutorial," *IEEE Computer*, Vol. 35, No. 4, pp. 68-73.
- Thayer, R.H. & Christensen, C. J. (Eds.). (2002). *Software Engineering, Volume 2: The Supporting Processes* (2nd ed.). Los Alamitos: IEEE Computer Society Press.
- Thayer, R.H. & Dorfman, M. (Eds.). (2002). *Software Engineering, Volume 1: The Development Process* (2nd ed.). Los Alamitos: IEEE Computer Society Press.
- Torgerson, W. (1958). *Theory and Methods of Scaling*. New York: Wiley.
- Turner, R. & Boehm, B. (2003). "People Factors in Software Management: Lessons from Comparing Agile and Plan-Driven Methods," *Crosstalk*, Vol. 16, No. 12, pp. 4-8.
- Tushman, M.L. (1977). "A Political Approach to Organizations: A Review and Rationale," *Academy of Management Review*, Vol. 2, No. 2, pp. 206-216.
- Tversky, A. & Kahneman, D. (1971). "Belief in the Law of Small Numbers," *Psychological Bulletin*, Vol. 76, No. 2, pp. 105-110.

- Tversky, A. & Kahneman, D. (1974). "Judgment under Uncertainty: Heuristics and Biases," *Science*, Vol. 185, No. 4157, pp. 1124-1131.
- United States Department of Health, Education, and Welfare. (1979). *The Belmont Report: Ethical Principles and Guidelines for the Protection of Human Subjects of Research* (DHEW Publication 78-0012). Washington: OPRR Reports.
- United States General Accounting Office, Program Evaluation and Methodology Division. (1990). *Case Study Evaluations* (GAO/PEMD-91-10.1.9). Washington: Government Printing Office.
- Venn, J. (1973). *The Principles of Inductive Logic*. New York: Chelsea Publishing Company (Original work published 1907).
- Wallace, R.H., Stockenburg, J.E. & Charette, R.N. (1987). *A Unified methodology for Developing Systems*. New York: McGraw-Hill Book Company.
- Wallace, L., Keil, M. & Rai, A. (2004). "How Software Project Risk Affects Project Performance: An Investigation of the Dimensions of Risk and an Exploratory Model," *Decision Sciences*, Vol. 35, No. 2, pp. 289-321.
- Walsh, H.T. (1962). "Whewell and Mill on Induction," *Philosophy of Science*, Vol. 29, No. 3, pp. 279-284.
- Walsham, G. (1995). "Interpretive Case Studies in IS Research; Nature and Method," *European Journal of Information Systems*, Vol. 4, No. 2, pp. 74-81.
- Walton, D.C. (2004). "Modeling Organizational Systems: Banathy's Three Lenses Revisited," *Systemic Practice and Action Research*, Vol.17, No. 4, pp. 265-284.
- Weaver, W. (1948). "Science and Complexity," *American Scientist*, Vol. 36, No. 4, pp. 536-544.
- Weick, K.E. (1974). "Middle Range Theories of Social Systems," *Behavioral Science*, Vol. 19, No. 6, pp. 357-367.
- Weick, K.E. (1989). "Theory Construction as Disciplined Imagination," *Academy of Management Review*, Vol. 14, No. 4, pp. 516-531.
- Weick, K.E. (1995). "What Theory is Not, Theorizing Is," *Administrative Science Quarterly*, Vol. 40, No. 3, pp. 385-390.
- Weitzman, E.A. & Miles, M.B. (1995). *Computer Programs for Qualitative Data Analysis: A Software Sourcebook*. Thousand Oaks: Sage Publications.
- Wernick, P. & Hall, T. (2004). "Can Thomas Kuhn's Paradigms Help Us Understand Software Engineering?" *European Journal of Information Systems*, Vol. 13, No. 3, pp. 235-243.
- West, M. (2004). "Applying Systems Thinking to Process Improvement," *Crosstalk*, Vol. 17, No. 3, pp. 26-29.
- Whetten, D.A. (1989). "What Constitutes a Theoretical Contribution," *Academy of Management Review*, Vol. 14, No. 4, pp. 480-495.

- Whewell, W. (1840). *The Philosophy of the Inductive Sciences, Founded Upon Their History, Volumes I and II*. London.
- Whewell, W. (1847) *The Philosophy of the Inductive Sciences, Founded Upon Their History, Volumes I and II*, (2nd ed.). London.
- Whewell, W. (1849). *Of Induction, with Special Reference to Mr. J. Stuart Mill's System of Logic*. London: John W. Parker.
- Whewell, W. (1857). *The History of the Inductive Sciences, from the Earliest to the Present Time*. (3rd ed.). London: John Parker.
- Whewell, W. (1858). *Novum Organum Renovatum*. (3rd ed.). London: John W. Parker and Son.
- White, B. (2000). *Dissertation Skills for Business and Management Students*. London: Thomson.
- White, H.C. (1970). *Chains of Opportunity*. Cambridge: Harvard University Press.
- Whittemore, R., Chase, S.K. & Mandle, C.L. (2001). "Validity in Qualitative Research," *Qualitative Health Research*, Vol. 11, No. 4, pp. 522-537.
- Wilson, D.N., Hall, T. & Baddoo, N. (2001). "A Framework for Evaluation and Prediction of Software Process Improvement Success," *Journal of Systems and Software*, Vol. 59, No. 2, pp. 135-142.
- Wilson, E.O. (1998). *Consilience: The Unity of Knowledge*. New York: Alfred A. Knopf.
- Yin, R.K. (2003). *Case Study Research: Design and Methods*. (3rd ed.). Thousand Oaks: Sage Publications.
- Yolles, M. (2004). "Implications for Beer's Ontological System/Metasystem Dichotomy," *Kybernetes*, Vol. 33, No. 3/4, pp. 726-764.
- Znaniecki, F. (1934). *The Method of Sociology*. New York: Farrar & Rinehart.
- Zelditch, M. (1962). "Some Methodological Problems of Field Studies," *American Journal of Sociology*, Vol. 67, No. 5, pp. 566-576.
- Zhong, X., Madhavji, N.H. & El Emam, K. (2000) "Critical Factors Affecting Personal Software Processes," *IEEE Software*, Vol. 17, No. 6, pp. 76-83.
- Zhu, Z. (2000). "WSR: A Systems Approach for Information Systems Development," *Systems Research and Behavioral Science*, Vol. 17, No. 2, pp. 183-203.

APPENDIX A: ODU IRB RESEARCH EXEMPTION

No.: 06-039

**OLD DOMINION UNIVERSITY
HUMAN SUBJECTS INSTITUTIONAL REVIEW BOARD
NOTIFICATION OF EXEMPT RESEARCH**


To: Charles Keating
Responsible Project Investigator

DATE: April 20, 2006
IRB Decision Date


Re: The relationship between a structured systemic framework for software development and software development project performance
Name of Project

Please be informed that your research proposal has been considered by the Institutional Review Board and was found to be EXEMPT for the following reason(s):

survey research that carries no risk, 45CFR46.101(b)2



IRB Chairperson's Signature



date

Contact the IRB for clarification of the terms of your research, or if you wish to make ANY change to your research protocol that might alter its exempt status.

APPENDIX B: GUIDELINES FOR THE OUTSIDE EXPERT

ROLE OF THE EXPERT REVIEWER

The Purpose of the Review

The review conducted by the expert is a one-time feedback loop to verify that the literature review has captured all of the relevant information. The observed and collected facts will serve as the source of empirical data for the inductive development of the framework; providing an appropriate range of ideas, concepts, and theories. The observation and collection of empirical facts has a direct affect on the validity of the inductively developed framework, which “. . . depends primarily on the quality of the data base from which the inductive inferences were derived.” (Sutherland, 1973, p. 168)

The use of an expert, outside of the researcher, is intended to decrease research risk by ensuring that the information selected by the researcher is adequate enough to provide a firm foundation for the induction. The verification guidelines for the review conducted by the outside expert are based on his training, education, experience, and personal expertise in systems and software development. Electronic copies of all articles cited were provided to the expert on an accompanying CD-ROM.

EMPIRICAL FACTS FOR THE RESEARCH

Schema for the Literature Review

The multi-disciplinary nature of software development project management requires the inclusion of a variety of scholarly literature from the management, information systems, software, and systems fields of study. The literature search included appropriate scholarly journals in the fields associated with the research purpose

and primary research questions. As stated, the sources included in the schema were from a wide variety of disciplines and include the scholarly journals listed in Exhibit 1.

Discipline	Journal Title	ISSN	Article Retrieval Source
Dissertations	<i>Doctoral Dissertations</i>	N/A	Digital Dissertations
Management	<i>International Journal of Project Management</i>	0263-7863	Science Direct
Management	<i>Journal of Operations Management</i>	0272-6963	Science Direct
Management	<i>Engineering Management Journal</i>	1042-9247	ABI/INFORM Global (Proquest)
Management	<i>Project Management Journal</i>	8756-9728	ABI/INFORM Global (Proquest)
Management	<i>European Journal of Operational Research</i>	0377-2217	Science Direct
Management	<i>European Management Journal</i>	0263-2373	Science Direct
Management	<i>International Journal of Operations & Production Management</i>	0144-3577	ABI/INFORM Global (Proquest)
Management	<i>Journal of General Management</i>	0306-3070	Business Source Premier (EBSCO)
Management	<i>Harvard Business Review</i>	0017-8012	Business Source Premier (EBSCO)
Management	<i>Management Science</i>	0025-1909	Business Source Premier (EBSCO)
Software	<i>Communications of the ACM</i>	0001-0782	ACM Digital Library
Software	<i>Journal of the ACM</i>	0004-5411	ACM Digital Library
Software	<i>IEEE Computer</i>	0018-9162	IEEE Digital Library
Software	<i>IEEE Transactions on Software Engineering</i>	0098-5589	IEEE Digital Library
Software	<i>IEEE Software</i>	0740-7459	IEEE Digital Library
Information Science	<i>Decision Sciences</i>	0011-7315	ABI/INFORM Global (Proquest)
Information Science	<i>Decision Support Systems</i>	0167-9236	Science Direct
Information Science	<i>MIS Quarterly</i>	0276-7783	Business Source Premier (EBSCO)
Information Science	<i>Information and Management</i>	0378-7206	Science Direct
Information Science	<i>Journal of Management Information Systems</i>	0742-1222	Business Source Premier (EBSCO)
Information Science	<i>Information Systems Research</i>	1047-7047	Business Source Premier (EBSCO)
Information Science	<i>European Journal of Information Systems (old Journal of Applied Systems Analysis 1969-1991)</i>	0960-085X	ABI/INFORM Global (Proquest)
Systems	<i>Journal of the Operational Research Society</i>	0160-5682	JSTOR
Systems	<i>Journal of Systems and Software</i>	0164-1212	Science Direct
Systems	<i>Kybernetes: The International Journal of Systems & Cybernetics</i>	0368-492X	Emerald Fulltext
Systems	<i>Systems Research and Behavioral Science</i>	1092-7026	ABI/INFORM Global (Proquest)
Systems	<i>Systemic Practice and Action Research</i>	1094-429X	ABI/INFORM Global (Proquest)
Systems	<i>Crosstalk</i>	0000-0000	www.stsc.af.mil/crosstalk

Exhibit 1: Scholarly Journals in Literature Review

A clear distinction has been made between published literature that is founded on empirical research and that which has been published with no empirical basis, with the latter excluded from the review.

The scholarly journals selected for the literature review were included to describe the theoretical perspectives and previous research findings related to the research purpose. Exhibit 1 includes the primary scholarly journals in management, software, information systems, and systems. Journal articles related to the research purpose were classified within four areas:

1. Systems principles,
2. Systemic improvement frameworks for software development,
3. Application of systems principles to software development, and
4. Software development project performance.

A scholarly review and a concise report of the findings and themes present in the literature were conducted. Exhibit 2 depicts the schema for the literature review and how the wide body of knowledge was narrowed to support the development of a generalizable assessment framework for software systems development. Exhibit 2 also shows how the 245 screened abstracts were distributed among the four focus areas. It also shows how the literature was synthesized to a document database of 34 applicable scholarly articles.

Synthesis of the Literature

A review of literature on systems principles, systemic improvement frameworks for software development, application of systems principles to software development, and software development project performance was conducted. It focused on empirical studies that could contribute to the research and provide a solid foundation for the

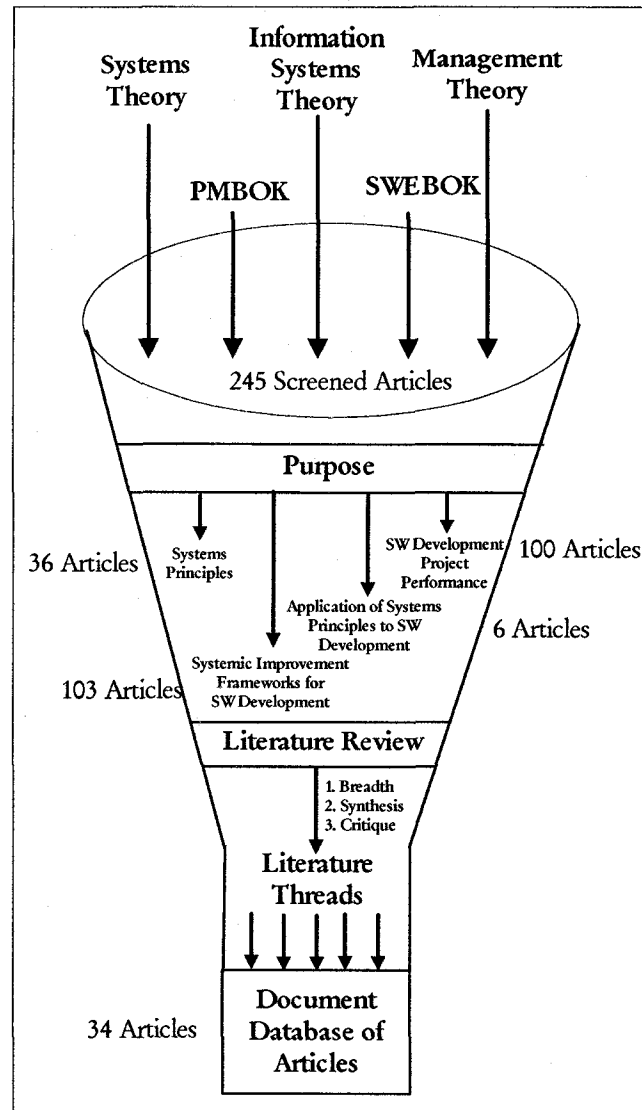


Exhibit 2: Schema for Literature Review

development of the theoretical framework for software development. Exhibit 3 presents the 34 citations, each have some bearing on the research and are intended to relate theory and practice to the principal research questions.

Exhibit 3 also shows the gap in the literature surrounding the application of systems principles to software development which serves as the focal point for the research. The purpose of the research is to develop and apply a systems-based framework for the analysis of software development project performance. The framework will provide the conceptual basis for understanding the context surrounding

Literature	Systems Principles	Systemic Improvement Frameworks for SW Development	Application of Systems Principles to SW Development	SW Development Project Performance
West (2004)	X			
Bohr (1928, 1937)	X			
Simon (1955, 1956)	X			
Ackoff (1979a, 1979b); Keating, Kauffmann & Dryer (2001)	X			
Hitch (1953)	X			
Ashby (1956); Conant & Ashby (1970)	X			
Beer (1984), Espejo (2004) and Yolles (2004)	X			
Walton (2004); Ramsay, Boardman & Cole (1996)	X			
Shani, Grant, Krishnan & Thompson (1992); Shani & Sena (1994); Jacobs, Keating & Fernandez (2000)	X			
CMM [®] , CMMI [®]		X		
ISO/IEC 12207		X		
Boloix & Robillard (1995)		X		
Iivari, Hirschheim & Klein (1998)		X		
Hirschheim, Klein & Lyytinen (1996)		X		
Jiang, Klein, Hwang, Huang & Hung (2004)		X		
Dyba (2005)		X		
Bai & Lindberg (1999)			X	
Bennetts, Wood-Harper & Mills (2000)			X	
Banker & Kemerer (1992)				X
Keil & Robey (1999); Keil, Mann & Rai (2000)				X
Aladwani (2002)				X
Wallace, Keil & Rai (2004)				X
Jones (2004)				X

Exhibit 3: Literature Relationship to Research Purpose

software development projects, but will also support the development of formal methodologies that can be used by software practitioners to improve software development project performance.

The strength of the framework will be derived from being grounded in the theoretical constructs derived from the application of systems theory. Development of the framework will use *Discoverers' Induction*, with the categories, attributes, relationships, and dimensions of the framework drawn directly from the 34 scholarly

articles in Exhibit 3. Five important threads have been drawn from the synthesis of the literature.

1. A number of systems-based principles and concepts exist in the literature that can be applied to the research questions.
2. Systems-based methods and models exist that may be adequate to holistically describe the software development process.
3. Few existing software development frameworks and/or methodologies address the overall development process holistically.
4. There has been limited application of systems principles to the problems associated with software development.
5. The literature on software development project performance does not address the root causes of poor performance.

Critique of the Findings

The review of the body of literature shows that a gap exists in the application of systems principles to software development and the need for additional empirical research. The gap shows a failure to treat software development projects as an organized or complex whole; a system. The software engineering community has been unable to coherently integrate their knowledge of the individual software development and management processes (sub-systems) in order to better understand the overall socio-technical system in which each of the development and management processes exists. Notable exceptions are the work of Abdel-Hamid (1984, 1988, 1992, 1993, 1996), Abdel-Hamid & Madnick (1989, 1990, 1991), Thayer (1979, 1997, 2002), Thayer & Christensen (2002), Thayer & Dorfman (2002), and Sengupta & Abdel-Hamid (1996) who have approached the management of software development projects from a holistic, systemic perspective. An early text on software engineering includes the following statement (Jensen & Tonies, 1978):

There is much attention on individual phases and functions of the software development sequence, but little on the whole life cycle as an integral, continuous process – a process that can and should be optimized. (p. 25)

They go on to state:

A systems treatment of the whole process from conceptual stage through product installation and operation is needed. (p. 25)

Since these statements were made limited research has been conducted on the integrated software development life-cycle process. In fact, the widely accepted literature on software process improvement provides conflicting advice. Of particular concern is the widespread adoption and implementation of the Software Engineering Institute (SEI) Capability Maturity Model (CMM®) and the Integrated CMM® (CMMI®). The SEI CMM® and CMMI® have been used as frameworks for assessing an organization's ability to produce software as well as a *de facto* software development standard. CMM® and CMMI® assessments routinely evaluate software development projects and organizations by reviewing the individual processes that are described in the CMM® or CMMI®, without regard for the integrated process. The CMM® and CMMI® and their associated assessment methodologies, run contrary to the Systems Principle of Suboptimization, where optimization of the individual software development and management processes occur at the expense of the larger software development project (system).

A new holistic, systemic view may reveal a better way to look at performance. The application of basic systems principles, within a structured systemic framework for software development, may provide insight into the failure to achieve overall software development project (system) improvement despite improvements within a number of the individual software development processes (sub-systems).

Observation and Collection of Facts for the Induction

The detailed literature review conducted in Chapter 2 synthesized the real-world facts relevant to software development project performance to the list presented in Exhibit 4.

Five Predominant Threads	References in Extant Literature
1. A number of systems-based principles and concepts exist in the literature that can be applied to the research questions.	<ul style="list-style-type: none"> • West (2004)
✓ Principle of complementarity,	• Bohr (1928)
✓ Principle of satisficing,	• Simon (1955, 1956)
✓ Principle of sub-optimization,	• Hitch (1953)
✓ Principle of minimum critical specification	• Cherns (1976, 1987)
✓ Concept of context	<ul style="list-style-type: none"> • Ackoff (1979a, 1979b) • Keating, Kauffmann & Dryer (2001)
2. Systems-based methods and models exist that may be adequate to holistically describe the software development process.	
✓ The Viable Systems Model	<ul style="list-style-type: none"> • Beer (1984) • Espejo (2004) • Yolles (2004)
✓ Banathy's three-lens model	• Walton (2004)
✓ Sociotechnical Systems	<ul style="list-style-type: none"> • Shani, Grant, Krishnan & Thompson (1992); Shani & Sena (1994); Jacobs, Keating & Fernandez (2000)
✓ Other methods	• Ramsay, Boardman & Cole (1996)
3. Few existing software development frameworks and/or methodologies that address the overall development process holistically.	
✓ Frameworks	<ul style="list-style-type: none"> • Sheard (2001, 2002) • CMM[®], CMMI[®] • ISO/IEC 12207 • Humphrey (1995, 1996a, 1996b, 2000) • Boloix & Robillard (1995)
✓ Methodologies	<ul style="list-style-type: none"> • Iivari, Hirschheim & Klein (1998) • Hirschheim, Klein & Lyytinen (1996) • Jiang, Klein, Hwang, Huang & Hung (2004) • Dyba (2005)
4. There has been limited application of systems principles to the problems associated with software development.	<ul style="list-style-type: none"> • Bai & Lindberg (1999) • Bennetts, Wood-Harper & Mills (2000)
5. The literature on software development project performance does not address the root causes of poor performance.	<ul style="list-style-type: none"> • Banker & Kemerer (1992) • Keil & Robey (1999) • Keil, Mann & Rai (2000) • Aladwani (2002) • Wallace, Keil & Rai (2004) • Jones (2004)

Exhibit 4: Decomposed Facts for the Inductive Process

Outcome of the Review

The researcher anticipates that the expert will, upon completion of his review, (1) provide comments on selected articles, and (2) recommend additional scholarly articles from the literature that will add depth and provide increased understanding useful in the development of the framework.

The remaining pages of this Appendix have space for the expert reviewer to comment on selected articles and to recommend supplementary readings. The completed Appendix will become research data and the recommendations will be included as additional extant literature for the induction.

Thread Area	Scholarly Article	Expert Reviewer Comments
1. A number of systems-based principles and concepts exist in the literature that can be applied to the research questions.	West (2004)	
✓ Principle of complementarity,	Bohr (1928)	
✓ Principle of satisficing,	Simon (1955, 1956)	
✓ Principle of sub-optimization,	Hitch (1953)	
✓ Principle of minimum critical specification	Cherns (1976, 1987)	
✓ Concept of context (1)	Ackoff (1979a, 1979b)	
✓ Concept of context (2)	Keating, Kauffmann & Dryer (2001)	
2. Systems-based methods and models exist that may be adequate to holistically describe the software development process.		
✓ The Viable Systems Model (1)	Beer (1984)	
✓ The Viable Systems Model (2)	Espejo (2004)	
✓ The Viable Systems Model (3)	Yolles (2004)	
✓ Banathy's three-lens model	Walton (2004)	
✓ Other methods (1)	Nakamori & Sawaragi (2000)	
✓ Other methods (2)	Ramsay, Boardman & Cole (1996)	
3. Few existing software development frameworks and/or methodologies that address the overall development process holistically		
✓ Frameworks	Sheard (2001, 2002)	
✓ Frameworks	CMM [®] , CMMI [®]	
✓ Frameworks	ISO/IEC 12207	
✓ Frameworks	ISO/IEC 90003	
✓ Frameworks	ISO/IEC 15504	
✓ Frameworks	Humphrey (1995, 1996a, 1996b, 2000)	
✓ Frameworks	Boloix & Robillard (1995)	

Thread Area	Scholarly Article	Expert Reviewer Comments
✓ Methodologies	Iivari, Hirschheim & Klein (1998)	
✓ Methodologies	Hirschheim, Klein & Lyytinen (1996)	
✓ Methodologies	Jiang, Klein, Hwang, Huang & Hung (2004)	
✓ Methodologies	Dyba (2005)	
✓ Methodologies	Turner & Boehm (2003)	
4. There has been limited application of systems principles to the problems associated with software development.		
	Bai & Lindberg (1999)	
	Bennetts, Wood-Harper & Mills (2000)	
5. The literature on software development project performance does not address the root causes of poor performance		
	Banker & Kemerer (1992)	
	Keil & Robey (1999)	
	Keil, Mann & Rai (2000)	
	Aladwani (2002)	
	Wallace, Keil & Rai (2004)	
	Jones (2004)	
Additional Scholarly Article 1		
Additional Scholarly Article 2		
Additional Scholarly Article 3		
Additional Scholarly Article 4		

APPENDIX C: GUIDELINES FOR THE PANEL OF EXPERTS

ROLE OF THE PANEL OF EXPERTS

The purpose of the verification is to ensure that the completed framework “. . . *looks like* it measures what it was intended to measure” (Nunnally, 1967, p. 99) prior to the formal validation with the real-world case studies. By occurring prior to the formal validation it allows the researcher to incorporate the comments of experts prior to the case study validation, increasing the validity of the inductive process, the stability of the framework, and the external validity and transferability of the research.

THEORETICAL BACKGROUND FOR THE VERIFICATION

The post-development external verification process used by the panel of experts is based on the work of Ahire & Devaraj (2001) who recommend using both content and face validation criteria. Exhibit 5 provides expanded definitions of both content and face validation; the two validity checks most commonly used at the completion of the development phase for measurement instruments.

Three key features are addressed when evaluating a theoretical framework.

1. Boundaries of the Framework

The 1st feature addresses the boundaries of the framework and where the framework is expected to be effective. This is called the domain and is defined as “. . . the territory over which we can make truth statements about the framework, and therefore, about the values of the units composing the framework.” (Dubin, 1978, pp. 134-135) “There is an inverse relationship between the number of boundary-determining criteria employed in a

model and the size of the domain owned by the model.” (Dubin, 1978, p. 134) This must be

Validity Check	Definition
Content Validity	<ul style="list-style-type: none"> • “The degree to which an empirical measurement reflects a specific domain of content (Carmines & Zeller, 1979, p. 20)” • “The <i>representativeness</i> or <i>sampling adequacy</i> of the content – the substance, the matter, the topic – of a measuring instrument (Kerlinger & Lee, 2000, p. 667)” “Content validation, then, is basically judgmental. The items of a test must be studied, each item being weighed for its presumed representativeness of the universe. This means that each item must be judged for its presumed relevance to the property being measured, which is no easy task. Usually other <i>competent</i> judges should judge the content of the items. The universe of the content must, if possible, be clearly defined; that is, the judges must be furnished with specific directions for making judgments, as well as with the specification of what they are judging (Kerlinger & Lee, 2000, p. 668).”
Face Validity	<ul style="list-style-type: none"> • “Concerns the extent to which an instrument <i>looks like</i> it measures what it is intended to measure (Nunnally, 1967, p. 99).” “Face validity concerns judgments about an instrument after it is constructed. . . . Face validity can be considered one aspect of content validity, which concerns an inspection of the final product to make sure nothing went wrong in transforming plans into a completed instrument (Nunnally, 1967, p. 99).” • “Face validity is not validity in the technical sense. It refers to what the test appears to measure. Trained or untrained individuals would look at the test and decide whether or not the test measures what it was supposed to measure. There is no quantification of the judgment or any index of agreement that is computed between judges (Kerlinger & Lee, 2000, p. 668).”

Exhibit 5: Post-Development Validity Checks for Measurement Instruments

related to the stated intention to work with *theories of the middle range* (Merton, 1968).

“The sense or meaning that can be given to the term *theories of the middle range* is that they are models having not too few and not too many boundary-determining criteria.”

(Dubin, 1978, p. 135) The boundary conditions may include the following:

- a theoretical strategy upon which the framework is based
- a formal method for constructing the framework
- a position on the theoretical continuum
- the real-world domain within which it will be applied

2. Utility of the Framework

The 2nd feature is concerned with the utility of the framework. Utility addresses the question *what makes this framework useful?* This is what Maxwell (1962) calls the conditions of adequacy of the framework. In the most basic sense the framework's role is to report, explain and predict the facts concerning the phenomena under consideration. Maxwell provides an elegant statement when he states: "A framework allows us to express a greater number and a larger variety of *observational facts* – *and this is crucial* – *to explain these facts* (1962, p. 136)." Bacharach (1989) tabulates the characteristics of *utility* presented in Exhibit 6.

Characteristics	Utility
Variables	<i>Variable scope</i> : Variables must sufficiently although parsimoniously tap the domain of the constructs in question.
Constructs	<i>Construct scope</i> : Constructs must sufficiently although parsimoniously tap the domain of the phenomenon in question.
Explanatory potential	<i>Explanatory potential</i> : Establishes the substantive meaning of constructs, variables, and their linkages.
Predictive adequacy	<i>Predictive adequacy</i> : Validates substantive meaning by comparing it to empirical evidence.

Exhibit 6: Utility Characteristics for the Framework

3. Pragmatic Factors and the Framework

The 3rd feature addresses the *usefulness* of the framework. Usefulness must answer the question *why is this framework more useful than another?* The question addresses the concerns that arise when more than one framework exists. Bacharach (1989), Maxwell (1962), and Whetten (1989) answer the question by citing a number of pragmatic factors that affect frameworks in Exhibit 7.

Pragmatic Factors	Elements
Logic underlying the Framework	<p>The framework must convince others that the propositions make sense.</p> <ul style="list-style-type: none"> • Useful guide for research. • Provides a framework for interpreting patterns, or discrepancies in empirical observations. • Explains <i>how</i>, <i>when</i>, and <i>why</i> certain relationships exist in the empirical data.
Simplicity	Ease of comprehension, communication, computations and other inferential manipulations.
Aesthetic Considerations	Idiosyncratic tastes and a language relevant for users of the framework.

Exhibit 7: Pragmatic Factors Affecting Frameworks

CRITERIA FOR EVALUATING THE FRAMEWORK

The panel of experts will use the following criteria when evaluating the framework. The criteria have been formatted as specific questions for the panel of experts to address.

Boundaries of the Framework. The framework will be an approximation of theory on the theoretical continuum; a theory of the middle range. The boundaries are as follows:

1. Has the framework been developed using an inductive method?
2. Has the framework development been based upon perspectives and solutions founded on systems principles?
3. Has the framework development used a generalizing theoretical strategy based on empirical data and ensures generalizability and transportability?
4. Is the framework applicable to the full-range of worldwide software development projects and remain unconstrained by the rapid evolution of associated development mechanics (languages, analysis and design methods, applications, etc.)?

5. Does the framework apply to software development projects of all durations?
6. Can the framework address software development projects contracted with either government or commercial customers?

Utility of the Framework. The framework's purpose is to report, explain and predict the facts concerning software development projects. The characteristics for evaluation are:

7. Do the measurement objects (i.e. variables) parsimoniously tap the domain of the constructs that they support?
8. Do the constructs (functions, structure, and environment) sufficiently and parsimoniously tap the software development project domain?
9. Has the framework established a substantive meaning and linkage between the measurement objects (i.e. variables) and the constructs?
10. Does the framework use empirical evidence (from the measurement objects) to validate predictive meanings?

Pragmatic Factors of the Framework. The framework must answer the question *why is this framework more useful than another?* The characteristics for evaluation are:

11. Can the framework interpret patterns or discrepancies in empirical observations (i.e. measurement objects)?
12. Is the framework a useful guide for continued research?
13. Can the framework be easily comprehended and communicated by the software engineering profession?
14. Does the framework involve simple computations and other inferential manipulations?
15. Is the framework presented in a language relevant to the software engineering profession?

ELICITATION METHOD

The elicitation method will be a modified *Delphi* situation in which each panel expert, isolated from one another and the researcher provides judgments on the 15 verification criteria from the previous section. The judgments will be made against the framework as presented in the section from Chapter 5 entitled *Inductive Development of the Framework for Software Development Project Performance*. The mode of response will have each expert panelist complete the verification form at the end of this section. The completed verification forms from the panel of experts will become research data with recommendations serving as sources of change for the framework. Because the panel of experts will be of limited size, formal statistical measures (Cohen, 1960; Lawshe, 1975) required to correlate the judgments of the panel will not be required.

OUTCOME OF THE REVIEW

The verification guidelines for the review conducted by the panel of experts are contained in the following section. The researcher anticipates that the panel of experts, isolated from one another and the researcher, will provide judgments about and recommendations to the framework that will add clarity to the study. The additional understanding gathered from the panelists ensures the plausibility of the framework. The specific recommendations of the panelist's will become research data and serve as sources of change for the framework.

Verification Criteria	Panelist's Comments
1. Has the inductive method used to develop the framework clearly related the empirical data used for the induction to the development of the theoretical concepts and resulting constructs?	
2. Has the framework development been based upon perspectives and solutions founded on systems principles?	
3. Has the framework development used a generalizing theoretical strategy based on empirical data and ensures generalizability and transportability?	
4. Is the framework applicable to the full-range of worldwide software development projects and remain unconstrained by the rapid evolution of associated development mechanics (languages, analysis and design methods, applications, etc.)?	
5. Does the framework apply to software development projects of all durations?	
6. Can the framework address software development projects contracted with either government or commercial customers?	
7. Do the measurement objects (i.e. variables) parsimoniously tap the domain of the constructs that they support?	
8. Do the constructs (functions, structure, and environment) sufficiently and parsimoniously tap the software development project domain?	
9. Has the framework established a substantive meaning and linkage between the measurement objects (i.e. variables) and the constructs?	

Verification Criteria	Panelist's Comments
10. Does the framework use empirical evidence (from the measurement objects) to validate predictive meanings?	
11. Can the framework interpret patterns or discrepancies in empirical observations (i.e. measurement objects)?	
12. Is the framework a useful guide for continued research?	
13. Can the framework be easily comprehended and communicated by the software engineering profession?	
14. Does the framework involve simple computations and other inferential manipulations?	
15. Is the framework presented in a language relevant to the software engineering profession?	
Additional Comment Number 1	
Additional Comment Number 2	

APPENDIX D: LITERATURE VERIFICATION COMMENTS FROM THE OUTSIDE EXPERT

Thread Area	Scholarly Article	Expert Reviewer Comments
Area 1. A number of systems-based principles and concepts exist in the literature that can be applied to the research questions.	West (2004)	Are there other principles that you haven't chosen to include – e.g., The Principle of Minimum Dissipation or the Law of Requisite Variety?
Area 2. Systems-based methods and models exist that may be adequate to holistically describe the software development process. The Viable Systems Model (1)	Beer (1984)	You may wish to cite Beer's first major work, <i>Cybernetics and Management</i> , English Universities Press, 1959. His VSM model is an outgrowth of a lot of previous work.
Area 3. Few existing software development frameworks and/or methodologies that address the overall development process holistically: Frameworks	CMM [®] , CMMI [®]	CMMI is also being touted as a systems approach – although it is not an integrated one.
Area 3. Few existing software development frameworks and/or methodologies that address the overall development process holistically: Frameworks	ISO/IEC 12207	Note: ISO/IEC 12207 is moving much closer to a systems engineering approach to software systems. Be aware of that.
Area 3. Few existing software development frameworks and/or methodologies that address the overall development process holistically: Methodologies	Turner & Boehm (2003)	You may want to mention the recent “agile” software development movement that does take a bounded systems view to development vs. the more open-ended approach of traditional software development practices. See Jim Highsmith's work in this area.
Additional Scholarly Article 1	Weinberg (1988) <i>Rethinking Systems Analysis & Design</i>	You may want to take a look at Gerry Weinberg's work. He has advocated a systems approach to software development.
Additional Scholarly Article 2	<i>Systems Engineering</i> (Sage 1992) and <i>Software Systems Engineering</i> (Sage and Palmer, 1990)	I think you have to mention the body of work created by Andrew Sage who was one of the first to really try to bring systems and software engineering together in some meaningful, integrated way.

Thread Area	Scholarly Article	Expert Reviewer Comments
Additional Scholarly Article 3	Systems Engineering: Fundamental Limitations by A. Sage, Proc. Of the IEEE, Vol. 69, No. 3 May – June 1986	May be a good paper to review to see what systems engineering cannot do – another point that you need to mention somewhere.
Additional Scholarly Article 4	<i>General Systems Theory</i> (Ludwig von Bertalanffy, 1968)	Bertalanffy is not well known in the US, but some say is the founder of systems theory. You have him listed in figure 2, but I think that he is more than just a link into general systems theory/
General Comment		You need to define very clearly what is meant by software development project performance – here it is implied to mean cost, schedule and technical performance AS SPECIFIED during development – not operations. You need to be very clear as to what is and is not considered in your work. Be careful to point out that you are not considering system performance in operation, and that you understand that a system may be developed in budget, schedule, and as specified, but still may not operate effectively once fielded.

APPENDIX E: FRAMEWORK VERIFICATION COMMENTS FROM THE PANEL OF EXPERTS

Evaluation Criteria	Comments from Panel of Experts
Boundaries of the Framework 1. Has the inductive method used to develop the framework clearly related the empirical data used for the induction to the development of the theoretical concepts and resulting constructs?	<p><u>Panelist 1</u>: Yes – well constructed links are evident. <u>Panelist 2</u>: The method used provides a clear connection between the source data and the five hierarchical elements of framework structure – open-coded nodes, subcategories, categories, concepts and the theory. While the relationships are articulated in the description of the framework development, it is recommended that the researcher avoid phrases such as “... themes or subcategories that the researcher <i>felt had significance for the construction of the framework</i>...” and “...the effort and <i>intuition</i> required...” which may suggest a higher element of subjectivity than is suggested in the description of the process. <u>Panelist 3</u>: The inductive approach seemed clearly and comprehensively developed and pursued. Inductive process elements such as the use of NVivo and conventional schemes such as open and axial coding, coupled with the use of accepted, ISO/IEC and other standards made a powerful case that necessary linkages had been made from data through all key, framework elements.</p>
2. Has the framework development been based upon perspectives and solutions founded on systems principles?	<p><u>Panelist 1</u>: Yes – the framework appears sound from a systems perspective. <u>Panelist 2</u>: Systems principles were clearly foundational in the construction of the framework as developed and defined. <u>Panelist 3</u>: The reference material identified and the explanations given for its use in and pertinence to the framework made clear the fundamental role that systems principles played in framework development.</p>
3. Has the framework development used a generalizing theoretical strategy based on empirical data and ensures generalizability and transportability?	<p><u>Panelist 1</u>: Yes – the framework has been generalized using the appropriate approach <u>Panelist 2</u>: The basis for the development of the framework for SW development project performance included a broad sampling of the literature within the field under study and the methodology utilized in analysis of the empirical data ensured that, to the greatest degree possible, the results were generalizable and transportable. <u>Panelist 3</u>: Framework structure seems faithfully aligned with systems theory precepts such as those espoused by von Bertalanffy, therefore ensuring its generalizability. The development process described in provided material reflects a strongly purposeful effort to afford generalizability and transportability without which the framework would seem useless on functional grounds. Even the application of ordinal scales to the lowest level of framework entities promoted generalizability and transportability.</p>
4. Is the framework applicable to the full-range of worldwide software development projects and remain unconstrained by the rapid evolution of associated development mechanics (languages, analysis and design methods, applications, etc.)?	<p><u>Panelist 1</u>: Partially – this reviewer believes that the framework developed is most effective for larger scale (1 million lines of code and above) projects that require significant management/administrative overheads (in the range of 15% to 25% of cost). For smaller, agile based developments, this reviewer does not believe that the framework would be as applicable. In addition, while the framework is applicable to current accepted standards of practice, there are new techniques on the horizon such as lean development of software that may not fit as well across the framework. Neither of these two limitations invalidates the general approach taken, however.</p>

Evaluation Criteria	Comments from Panel of Experts
4. (continued)	<u>Panelist 2:</u> The framework developed in this research is broadly applicable across the full spectrum of software development. It is not constrained to development efforts within any particular functional area or development environment. The general nature of the framework ensures that it will remain broadly relevant and applicable in software development for the foreseeable future regardless of innovations in tools and methods applied. <u>Panelist 3:</u> Per the arguments made for (3), the framework seems independent of software development mechanics.
5. Does the framework apply to software development projects of all durations?	<u>Panelist 1:</u> Partially - as commented above, the longer the project (a minimum of 12 – 18 months in duration) the more the framework appears to this reviewer to be applicable. <u>Panelist 2:</u> The framework is applicable to projects of any size, however, the return on investment from applying the framework on small/short-term projects is questionable. <u>Panelist 3:</u> Temporal considerations – including those that might be pertinent to issues regarding project duration – seem woven throughout framework elements. For example, framework elements address temporally pertinent, Life Cycle Support and Life Cycle Documentation factors, while temporal concerns more precisely associated with development processes per se were evident in framework elements such as cost, schedule pace, and material and capital fund related measures.
6. Can the framework address software development projects contracted with either government or commercial customers?	<u>Panelist 1:</u> Yes – as long as the different success factors are explicitly accounted for. Commercial projects are measured generally on a financial ROI or on a value returned to the shareholders. Government projects generally have non-financially based measures of success and criteria for their initiation. Furthermore, government projects tend to require greater process compliance with standards (CMMI & 12207) than commercial firms do. <u>Panelist 2:</u> The framework is customer/user independent. It has applicability in both the public and private sector, but possibly for different reasons. In the case of government development efforts, the framework could be used to give government program managers a sense of the level of performance for a set of development efforts. In the private sector, where profitability is the key metric, the framework could be used to differentiate which efforts were providing the greatest return for the development dollar. <u>Panelist 3:</u> The framework plainly addressed each of those customers, and probably in terms general enough to justify the identical measurement criteria associated with the different two distinct stakeholders.
Utility of the Framework 7. Do the measurement objects (i.e. variables) parsimoniously tap the domain of the constructs that they support?	<u>Panelist 1:</u> Yes – measurement objects tap the domain constructs they support. There may be disagreements about some of the specific definitions (e.g., Table 13, INTL Measurement Criteria; it could be argued that political events should be captured in this table as well). However, given the data analysis, the inductive approach taken, the need to limit the size of the measures, the identified measurement objects seem reasonable. <u>Panelist 2:</u> The framework as constructed ensures parsimony in how the measurement objects were developed. The framework appears to provide a set of necessary and sufficient measures of project performance. <u>Panelist 3:</u> Perhaps, in fact probably so, though I did come away from the entire review with a feeling that the framework's application might be quite manpower intensive. That's not to say a manpower-intensive framework isn't required to do the job (i.e., to have the utility – your definition – you say it should), but I cannot help but wonder if fewer variables might not be adequate for the task; a question you could possibly answer by applying sensitivity analysis to the two case studies you anticipate in your second phase of research.

Evaluation Criteria	Comments from Panel of Experts
8. Do the constructs (functions, structure, and environment) sufficiently and parsimoniously tap the software development project domain?	<i>Panelist 1:</i> Yes – these three constructs provide feasible and adequate coverage. <i>Panelist 2:</i> The analysis and coding process employed in the methodology ensured parsimony in the development of the constructs. The aggregation of concepts from the data into higher level concepts ensured the simplest conceptual construction. <i>Panelist 3:</i> I feel better about the balance of sufficiency and parsimony you’ve struck here than I do about the same balance (or imbalance) reflected at the variable level.
9. Has the framework established a substantive meaning and linkage between the measurement objects (i.e. variables) and the constructs?	<i>Panelist 1:</i> Yes – they are meaningful and material. <i>Panelist 2:</i> The linkage between the constructs and the measurement objects is very clearly articulated in the description of the framework and collectively provide a comprehensive depiction of all of the elements of software development projects that collectively contribute to overall performance. <i>Panelist 3:</i> Clearly.
10. Does the framework use empirical evidence (from the measurement objects) to validate predictive meanings?	<i>Panelist 1:</i> As noted previously, there is some necessary judgment needed here on part of the researcher to choose which are the most appropriate. In this reviewer’s opinion, the framework uses sufficient empirical evidence to validate predictive meanings. <i>Panelist 2:</i> The reflexivity between the constructs and the measurement objects provides the necessary predictive adequacy for the framework. The researcher ensured that these relationships were well supported by the empirical data collected during the research. <i>Panelist 3:</i> I believe it may well, though I have one major concern about the measurement criteria that could affect framework validity: while I understand all Keil & Robey referenced justifications you’ve invoked to establish certain of your resource measurement criteria, I do not agree believe that two of them cover all the bases they should. Your first (owners and shareholder boards) and second (external management) factors related to resources key only upon the goodness that might be associated with owners, stakeholder boards, and external managers staying out of the way of project development processes. Might there not be some goodness in the involvement of those communities? If there is – and I’d be mighty reticent to say there’s not, if I were you – your criteria seems in no way able capture it. Did I miss something?
<p>Pragmatic Factors of the Framework</p> <p>11. Can the framework interpret patterns or discrepancies in empirical observations (i.e. measurement objects)?</p>	<i>Panelist 1:</i> The framework provides a basis for pattern interpretation and observed discrepancies. However, it is not entirely clear how well patterns identified will be linked to sources of risks or problems. Linkages/interconnections among constructs that can show cause – effects are suspect. The framework may be very good at showing recurring symptomatic issues, but not systemic causative risk/problem issues in the case studies. <i>Panelist 2:</i> The framework does provide the ability to interpret patterns or discrepancies in empirical observations. The relationships between the framework, the constructs, the measurement concepts, and the measurement attributes allow a user of the framework to draw higher-level conclusions about project performance than would not be readily attainable from empirical observation alone... without the framework. <i>Panelist 3:</i> I think it would do a good job of that.

Evaluation Criteria	Comments from Panel of Experts
12. Is the framework a useful guide for continued research?	<i>Panelist 1:</i> Definitely yes. <i>Panelist 2:</i> The framework provides a worthwhile piece of constructive research that makes a valuable contribution to the general project management body of knowledge in the assessment/evaluation of project performance, with specific contribution in the area of software engineering/development project performance (an area where such an evaluation methodology is needed). <i>Panelist 3:</i> The framework is obviously well-conceived and quite clearly explained, though – and, here again, maybe I missed something – the one possible weak point of conception I noted rests with a lack of rationale for many of your variable measures and associated descriptors. Those seem to have been products of the researcher’s imagination, a tool you do state to have used, and convincingly so, in general terms early in chapter 5, but never specifically addressed as justification for any measures. If some of your measures and descriptors are researcher-generated, they alone would certainly beg for further research. On a grander scale, I’m sure we’d all concede that the framework could afford a level of exercise exceeding the two case studies you plan with your balance of work.
13. Can the framework be easily comprehended and communicated by the software engineering profession?	<i>Panelist 1:</i> This reviewer believes so, although systems engineers would more likely understand it than software engineers. <i>Panelist 2:</i> The logical flow of the framework and method used for measurement and assessment should be easily understood by those who work within the software engineering/development domain. <i>Panelist 3:</i> “Comprehended by”...yes, absolutely. “Communicated by”...if that’s what you truly meant, the answer would depend on software professionals’ level of expertise with your framework’s foundational elements of systems science and the tools of inductive reasoning you used to emplace systems principles et al within the framework...and you could justifiably forecast that answer, then, to be “no.” If, on the other hand, what you meant was “Communicated to the software engineering profession,” then I’d say you’re squarely back in “yes” –land.
14. Does the framework involve simple computations and other inferential manipulations?	<i>Panelist 1:</i> Yes – there is nothing computationally complex (although it could easily made so!!) <i>Panelist 2:</i> The framework provides an approach that is not overly complicated in the level of mathematical understanding required for its use. <i>Panelist 3:</i> The measure values are simple, seem appropriate (don’t forget my comment for (12), however), and are simple to manipulate, particularly since the researcher has taken care to construct scales of good and bad things in commensurable fashion, i.e., the “baddest” evaluations of bad things garner fewest points on applicable scales, whereas “bestest” evaluations of good things garner most points on scales applicable to good things...so everyone lives happily ever after, with goodness always on the right and badness always on the left (a quite unsettling circumstance for us southpaws, by the way!), with bad stuff gaining few points and good stuff gaining many points!
15. Is the framework presented in a language relevant to the software engineering profession?	<i>Panelist 1:</i> Yes – but again, the systems engineering community may find it of more value. <i>Panelist 2:</i> Because it is based on a broad spectrum of literature, much of which is drawn specifically from the software engineering domain, the framework is presented in language that is both relevant to and easily integrated by those working in the software engineering/development domain. <i>Panelist 3:</i> I believe that the software engineering community would: readily identify with language used to describe framework variables; likely readily concede an understanding of your highest construct level components of “functions,” “structure,” and “environment” without too much fuss, and have the most difficulty understanding – and therefore, implementing – your 15 constructs. Bottom line answer, though, would have to be “yes”...they’d figure out what they needed...probably!

Evaluation Criteria	Comments from Panel of Experts
Additional Comment 1	<i>Panelist 1:</i> There is an issue that needs to be addressed in terms of the research's limitations. The focus is on software development only, and success is defined as being within cost/schedule/technical requirements. However, this presumes that the original estimates of these three parameters are correct measures of success. In practice, especially in larger developments, original estimates are always incorrect. In fact, success in government IT projects is deemed by OMB to be within 10% of the original estimates. The framework does not address poor estimation practice – is this considered to be part of development, or something else? For instance, external or internal forces (competitor pressure, organizational politics, etc.) may influence an estimate greatly. Furthermore, external (or internal events) may require a re-base lining of these parameters. So, if there is a change, is the project a failure? In addition, real success is defined by the software system in operation. A system may meet its cost, schedule, etc. but fail miserably in operation. The issue of development success and what it means needs to be discussed fully.
Additional Comment 2	<i>Panelist 1:</i> Another issue is that software development fails from basically two reasons – the development is OBE which make it no longer needed, or the cost of rework overwhelms the resources available (see Charette, "Why Software Fails," IEEE Spectrum, September 2005). Rework is the best general measure for understanding that a project is getting into trouble. It is also a precursor for escalation of issues to senior management.
Additional Comment 3	<i>Panelist 1:</i> There is an underlying assumption that CMMI and 12207 define acceptable software development approaches, yet this is open to debate in the community. Compliance with these standards does not ensure performance. If the development is not within an organization's competence, a high level CMMI rating may be a total mis-characterization of the likelihood of development success. Context is very important to whether a project can be successful. Furthermore, CMMI and 12207 are most applicable for large scale-developments (a vestige of their history). No single project (or project sponsor) could ever afford to do everything that is called for in CMMI and 12207 – therefore some trade-offs are always necessary. A major benefit of the framework will be if it can predict whether some of these trade-offs (i.e., what wasn't done) lead to failure more than others. Given the limitation of the number of case studies to be examined, this may not be possible now. However, it makes for an interesting future research issue.
Additional Comment 4	<i>Panelist 1:</i> First, there is an article in CIO magazine about 25 IT horror stories. You may want to look at them and do a quick bounce off the framework.
Additional Comment 5	<i>Panelist 1:</i> I am working on a paper on why DoD Projects succeed. In almost every instance, they do because they break the "rules." You may want - for protection sake - to have a small discussion on project success. Absence of modes of failure does not mean success - just as the CMMI and 12207 are not based on case studies of success, but on case studies of failure. This I think is critical, and is a limitation of the framework. You can put it down for future research.
Additional Comment 6	<i>Panelist 2:</i> This framework provides a much needed bridge between the detailed empirical observations that are routinely tracked and measured in software development projects and the overarching performance criteria that can be equated with "measures of success" for a given software project. The empirical measurements have been used for a long time to gauge project performance, but do not adequately provide the predictive assessment that this framework will provide. The individual measures describe what is happening... the framework answers the question - "So what?"

APPENDIX F: DIA BHS CASE STUDY

1. INTRODUCTION

This case study will present the facts surrounding the design and implementation of the automated baggage handling system at the Denver International Airport. The 1st part of the study will provide background material essential in understanding the decision to adopt a complex automated baggage handling system for the world's largest and newest airport. The 2nd part of the study will review the scope of the baggage handling system and its supporting software system. The 3rd part of the case study will review the outcome of the baggage handling system design and implementation at DIA. The 4th and final part of the case study will evaluate the design and implementation of the baggage handling system using the Function-Structure-Environment (FSE) Framework.

2. BACKGROUND

The Denver International Airport (DIA) was the first new airport to be designed and built in the United States since 1974 (Kerzner, 2000). The new airport was to replace Denver's Stapleton International Airport and was the outgrowth of 20 years of political maneuvering by stakeholders in the Denver metropolitan area. The airport was to satisfy Denver's air transportation needs for 50-60 years and include all-weather operations. The City and County of Denver joined with a joint-venture engineering, architecture and airport-design firm to manage the airport design and construction. The undertaking was a true mega-project. The design encompassed 53 square miles, an area twice the size of Manhattan, and included 6 runways and 120 gates. In 1989 the opening was targeted for October 1993 with a projected cost of \$5 Billion. The City of Denver contributed \$3.8B in bonds, the Federal Aviation Administration (FAA) \$685M, and the Airlines \$400M for

facilities and equipment. A key element of the new design was United Airlines' automated baggage handling system.

The background for the Case Study involves two key elements: the political environment and the design of the airport. Both of these elements are essential in understanding why the airport chose to select a complex automated baggage handling system as an essential feature of the new airport design.

Political Environment

The political environment that influenced the decision makers involved with the DIA mega-project was a contributing factor in the decision to use an automated baggage handling system. E.J. Feldman, a scholar of airport planning, made the following statement (Szyliowicz & Goetz, 1995):

The fate of megaprojects is determined not only by difficulties in forecasting but by such political factors as the nature of bureaucracies, the role of citizens, and how financing and administration of these projects proceed. (p. 348)

Feldman's observation was particularly relevant in the DIA case and is characterized by three elements.

The 1st element addresses the role of the bureaucracy. The need for additional air transportation facilities was recognized by everyone in the Denver metropolitan area. Rapid population growth and increased air travel had propelled Denver at a pace which had eclipsed the capacity of the existing airport, Stapleton International Airport (SIA) in Adams County. Expansion of SIA was severely constrained because of its location. A regional commission, the Denver Regional Council of Governments (DRCOG) was tasked with recommending a course of action. In October of 1983 the DRCOG voted to expand SIA, despite the fact that Adams County officials and voters vehemently opposed

the expansion of SIA. In November of 1983 Federico Peña was elected Mayor of Denver and conducted a thorough review of the DRCOG decision. Negotiations between the City of Denver and Adams County officials started in February 1984 and by January 1985 Adams County agreed to cede 50 square miles of uninhabited land east of Denver for a new airport, and SIA would be closed. However, the entire deal was subject to approval by the voters of Adams County scheduled for the election in May 1988.

The 2nd element addresses the roles of the citizens and their elected leaders. Proponents of the new airport, the majority of who were not citizens of Adams County, mounted an impressive political campaign in support of the new airport. Led by the Governor and the entire Colorado congressional delegation and a \$1.5M warchest, the campaign successfully turned back the opposition and the proposition passed by a vote of 56% to 44%. The new airport location and closure of SIA had been approved. The Denver politicians realized that the expansion would be a financial challenge and did not feel comfortable committing huge sums of money for the largest public works project in Denver history without the support of their electorate. The bonds required to support the design and construction of the airport were placed on the ballot. The political establishment and the business community mounted a huge campaign and won 66% of the vote. The bond issue for the new airport had been approved by the citizens of Denver.

The 3rd element addresses the funding. The City of Denver was faced with a bond issue of over \$3B. For the bonds to be able to sell, the project had to be financially viable, which required the support of the two major airlines with existing hubs at SIA; Continental and United. Neither Continental nor United were excited about the new

airport because of increased operating costs and overly optimistic passenger projections being used by the air port planners. The City was required to entice both airlines. In April 1990 Continental, the 2nd largest hubbing airline, agreed to lease 25 gates for 5 years in return for \$58M in concessions and design changes from the City. United reached an similar agreement with the City in June 1991, leasing 45 gates in return for concessions totaling \$204M and even more extensive design changes.

The City now had a financially viable plan for the new DIA. However, the design changes agreed to as part of the political process would prove to be disastrous.

Airport Design

The design methodology for the airport followed the FAA master planning prescription, which was based upon the rational comprehensive approach. (Goetz & Szyliowicz, 1997) The rational comprehensive approach included weaknesses in its ability to properly forecast air transportation demand projections, which proved to be a contributing, and possibly initiating factor in the series of financially disastrous delays in opening the airport. The size of the airport was based on the FAA passenger forecasts. In order to pay-back the bonds the City of Denver (hereafter referred to as *the City*) required a predetermined number of the airport gates to be leased by the participating airlines.

Neither of the airport's major hub carriers (Continental and United) were enthusiastic about the new airport. This was based on the potentially high-operating costs envisioned by the carriers. The City, pressed by the need to ensure its \$3.8B in bonds remained viable, was forced to negotiate, from a position of weakness, with the airlines. A number of major concessions were made that directly affected the design of the airport.

Continental acted first and received \$58M in concessions, one of which was the lease for Concourse A, closest to the main terminal. United acted next and received \$204M in concessions and access to the less desirable Concourse B. For accepting the more remote concourse the City conceded to United's demand for a major design change, a fully automated baggage handling system. United's goal was to ensure the transfer of passenger's bags in time for them to make connecting flights.

"The City had already explored the feasibility of installing an airport wide automated baggage system. In August 1990, a study commissioned by the City indicated that the highly complex and technically difficult state-of-the-art automated baggage system necessary for an airport of that size could probably not be built and tested in time to meet the scheduled opening date of October 1993." (GAO, 1995, p. 3) As a result of the study the City's initial design for the airport included a conventional tug-and-cart baggage system. This decision was not unusual, as most major airport designs in the United States relied upon the individual airlines to build their own baggage handling systems. (Rifkin, 1994)

However, within two weeks of United's June 1991 agreement the planners decided to change the design and incorporate a single airport-wide baggage handling system opting to use an automated baggage handling system for the entire airport (Russell, 1994). It is important to note that this decision was made fully 2 years into the overall airport design.

3. BAGGAGE HANDLING SYSTEM DESIGN

The City requested proposals from 16 firms (both domestic and foreign) and received only 3 responses. The consulting firm of Breier, Neidle and Patrone, "...

recommended against all three submitted designs on the grounds that the configurations would not meet the airport's needs." (Montealegre & Keil, 2000, p. 423) Boeing Aviation Equipment (BAE) Systems, which was designing United's automated system for Concourse B, was one of the 13 companies contacted by the City that chose not to respond. "The proposal represented a system of much greater size and complexity than anything BAE had built before." (Montealegre & Keil, 2000, p. 423)

Undeterred by their earlier analysis and the consultant's recommendation, the City once again approached BAE about designing and implementing an automated airport-wide system. The city wanted BAE, as BAE was considered the top baggage system company in the world having spent 26 years building baggage handling systems for individual airlines at airports in Atlanta, San Francisco, Dallas-Fort Worth and JFK. (Rifkin, 1994) BAE President Gene Di Fonso reconsidered the City's proposal.

BAE presented the City of Denver with a proposal to develop the "most complex automated baggage system ever built," according to Di Fonso. It was to be effective in delivering bags to and from passengers, and efficient in terms of operating reliability, maintainability, and future flexibility. The system was to be capable of directing bags (including suitcases of all sizes, skis, and golf clubs) from the main terminal through a tunnel into a remote concourse and directly to a gate. Such efficient delivery would save precious ground time, reduce close-out time for hub operations, and cut time-consuming manual baggage sorting and handling. (Applegate, Montealegre, Knoop & Nelson, 1996, p. 9)

The complexity of the overall system was enormous. System complexity was a function of the (1) physical distances, (2) volume of baggage, (3) transfer time requirement, (4) distributed architecture, (5) routing algorithms, (6) mechanical devices, and (6) high-technology components. For example, the system would:

Deliver bags automatically from check-in to the gate. Luggage with bar-coded tags would be placed into individual tele-carts and electronically scanned to identify their appropriate destinations and be routed to the correct gate. The system's scale and complexity is illustrated by the need

for hundreds of individual tele-carts, 17 miles of track, over 150 computers and servers, dozens of bar-code and radio frequency readers, and thousands of switching software programs, electric motors and photocells. (Goetz & Szyliowicz, 1997, pp. 270-271)

In addition, the technology being used by BAE was known to be difficult to implement because of the extensive amount of full-scale operational testing required.

The Frankfurt airport planners had adopted a smaller version of this system and it took them 15 years to build and test it before it worked properly. The Munich airport declined to use such a system. (Goetz & Szyliowicz, 1997, pp. 270-271)

A significant portion of the complexity was to be handled by the computer software. The software was responsible for tracking passenger baggage and managing and directing the actions of the baggage handling system. The software was the brain responsible for the “. . . central nervous system of some 100 computers networked to one another and to 5,000 electric eyes, 400 radio receivers and 56 bar-code scanners orchestrates the safe and timely arrival of every valise and ski bag.” (Gibbs, 1994, p. 86) The software had immediate challenges:

1. Managing a complex network of interacting, fully loaded queues efficiently for any single scenario is complicated. Managing these flows under all the realistic scenarios is exponentially more difficult. Learning how to do this appears to be a major, long-term research project. (de Neufville, 1994, p. 231)
2. Managing the information accurately is also difficult. The database needs to track tens of thousands of bags, going to hundreds of destinations, all in real time. The problem is further complicated at Denver because it uses a distributed system of about 150 computers. (de Neufville, 1994, p. 234)

3. The software must, in addition to the usual error checking codes that guard against electrical disturbances in the communications, have multiple levels of redundancy and be able to recover from errors very rapidly. (de Neufville, 1994, p. 234)
4. The software required the writing of millions of lines of computer code which were necessary to direct bags safely and correctly to their destination. (Goetz & Szyliowicz, 1997, p. 271)
5. The software was required to interface with each of the airlines' reservations systems. This caused BAE to enter a programming area where it had no expertise. "BAE ran into a raft of programming nightmares. One was writing the code for establishing and maintaining communications with the airlines' reservations systems, especially United's Apollo computers." (Rifkin, 1994, p. 113)

The brain in the system was actually a complicated BAE software program called the Empty Car Management System. The program was designed to dispatch baggage cars to any point that required them. The complex program was written by 20 BAE programmers over a two year period. Baggage, fitted with bar-coded tags or miniature photocells, were tracked by an array of lasers located along the input conveyors. The baggage location was transmitted to the central processor which directed located the proper flight. The bag was dropped into a cart and directed to the proper flight. Each cart had a radio frequency identification device (RFID) used by a series of radio transponders located along the tracks to monitor the movement and location of the cart. The baggage location was constantly updated in the central computer as it moves through

the system. This was complex, real-time software on the leading edge of the technology available in 1992.

The City accepted the design and in April 1992 awarded BAE a \$175.6M contract to build the entire airport baggage handling system, with no corresponding change to the original opening of the airport scheduled for October 1993. BAE accepted the schedule but Di Fonso placed the following conditions on acceptance of the contract.

The design was not to be changed beyond a given date and there would be a number of freeze dates for mechanical design, software design, permanent power requirements and the like. The contract made it obvious that both signatory parties were very concerned about the ability to complete. The provisions dealt mostly with all-around access, timely completion of certain areas, provision of permanent power, provision of computer rooms. All these elements were delineated as milestones.
(Applegate et al, 1996, p. 10)

4. BAGGAGE HANDLING SYSTEM IMPLEMENTATION

The implementation of the system was beleaguered with problems. Three problem areas stood out: (1) the organizational relationship between BAE and the City, (2) the ever-changing design, and (3) the insufficient time to test the system.

Organizational Relationship

“Design of the United baggage system was frozen on May 15, 1992, when the PMT assumed managerial responsibility for the integrated baggage system. The direct relationship with BAE was delegated to Working Area 4, which also had responsibility for building design efforts such as the people-mover, airside concourse building, passenger bridge main landside building complex and parking garage, and various other smaller structures.” (Applegate et al, 1996, p. 10)

This organizational relationship dictated by the Project management team (PMT) was foreign to BAE. BAE was accustomed to working within a management structure

that created an area of responsibility for systems that encompassed the entire airport, much like the baggage handling system. BAE was now required report to and work with a manager responsible for a single area with no overarching airport-wide view or responsibility. BAE had to modify their methods to make up for the inadequate PMT organization. Two major organizational changes occurred in the first six months:

1. In May 1992 the head of the DIA project resigned.
2. In October 1992 the chief airport engineer died

The new airport chief engineer's authority was challenged by the City requiring her to get approval for just about all decisions. BAE, forced far down in the management hierarchy had to live with the City's inability to make timely decisions and with a supervising manager that did not have experience with airport baggage handling systems. The President of BAE, Gene Di Fonso, acted as the project manager for the first two years of BAE's involvement at DIA.

"The relationship with the management team was very poor," recalled Di Fonso. The management team had no prior baggage handling competence or experience. This was treated as a major public works project. The management team treated the baggage system as similar to pouring concrete or putting in air-conditioning ducts. When we would make our complaints about delays and access and so forth, other contractors would argue their position. The standard answer was, "Go work it out among yourselves." . . . With contractors basically on their own, this led almost to anarchy. Everyone was doing his or her own thing. (Applegate et al, 1996, p. 13)

There were other perspectives of BAE's management. A highly experienced project manager from Stone & Webster, consulting for the PMT, made the following comments about BAE:

This contractor simply did not respond to the obvious incredible workload they were faced with. Their inexperienced project management vastly underestimated their task. Their work ethic was deplorable. (Applegate et al, 1996, p. 13)

Baggage Handling System Design Changes

The BHS design was also besieged with change requests.

The airlines began requesting changes to the system's design even though the mechanical and software designs were supposed to be frozen. "Six months prior to opening the airport," Di Fonso recalled, "we were still moving equipment around, changing controls, changing software design." (Applegate et al, 1996, p. 12)

"Denver's airport planners saddled BAE with \$20 million worth of changes to the design of its baggage handling system long after construction had begun." (Gibbs, 1994, p. 90)

Examples of the changes include the following:

- United eliminated one of the two loops of track to save \$20M, causing significant system redesign.
- United requested a change to the baggage size to accommodate additional ski equipment, adding \$1.61M to the cost of the system, changing the design, and adding additional system components.
- Continental requested additional baggage sorting in their West basement adding \$4.67M, changing the design
- A maintenance track was required, adding \$.9M, and major redesign

Additional problems surfaced. The City was unable to provide stable electrical power to the BHS. The system's motors and circuitry were extremely sensitive to power fluctuations and would routinely trip. The City contracted for electrical filters to correct the situation. Because of contract problems the filters did not arrive until March of 1994, delaying important testing. A particularly vexing problem related to personnel. Because this was a government project subject to City hiring laws, the City required a percentage of the jobs to be contracted with minority-owned companies. BAE was prohibited from

using its own personnel to perform system maintenance, a function that the City had contracted out in order to meet its minority firm quotas. All of these changes added to the system cost, now estimated at \$234M. (GAO, 1995)

Baggage Handling System Testing

Construction problems delayed the opening of the airport two times; from the original date of October 1993 to December 1993 to March 1994. The third delay, from March to May 1994, was solely as a result of problems in getting the baggage system to work properly. (GAO, 1995) BAE was trying to meet its commitment. In late April 1994 BAE was preparing for its first full operational test of the system. The City, without notifying BAE, invited reporters to observe the first test of the system. The reporters witnessed a debacle.

So many problems were discovered that testing had to be halted. Reporters saw piles of discarded clothes and other personal items lying beneath the telecar's tracks. After the test, Mayor Webb delayed the airport's opening for an indefinite period of time. "Clearly, the automated baggage system now underway at DIA is not yet at a level that meets the requirements of the city, the airlines, or the traveling public," the mayor stated. "There is only one thing worse than not opening DIA...[and] that is opening the airport and then having to shut it down because the [baggage] system doesn't work. (Montealegre & Keil, 2000, p. 424)

While this element of the test was embarrassing the overall 600 bag test results were not far from the specifications. "Outbound (terminal to plane), the sort accuracy was 94 percent and inbound accuracy was 98 percent. The system had a zero downtime for both inbound and outbound testing. The specification requirements called for 99.5 percent accuracy." (Kerzner, 1998, p. 624-625)

However, in May, the City, under mounting pressure from the public, the media, its political supporters, and the bond market hired an outside consultant to assess the state of the automated BHS. It is important to note, this is the first independent, outside

assessment of the automated BHS. The consultant, Logplan, issued their 11 page report which characterized the BAE system as follows:

"Highly advanced" and "theoretically" capable of living up to its promised "capacities, services, and performances," but acknowledged that software and mechanical problems "make it most improbable to achieve a stable and reliable operation. (Montealegre & Keil, 2000, p. 426)

The report also recommended the construction of a backup BHS, a conventional pull and tug system, much like that originally envisioned for the airport in 1991. The City contracted with Rapistan-Demag, a Michigan-based firm recommended by Logplan, to design and implement the conventional baggage handling system.

By November 1994 the baggage system was working, but only in segments "Frustration still existed in not being able to get the whole system to work at the same time. The problem appeared to be with the software required to get computers to talk to computers." (Kerzner, 1998, p. 630) Lawsuits and counterclaims were filed by the City and BAE.

In the end the airport managed to open in February 1995 with a conventional baggage handling system. The airport was 16 months late and close to \$2B over budget. (Montealegre & Keil, 2000)

5. BAGGAGE HANDLING SYSTEM EVALUATION

In this section the DIA automated BHS has been evaluated against the systemically-based FSE Framework. The framework has three elements; Function, Structure, and Environment. The evaluation matched the empirical facts concerning the automated BHS against each framework element. The evaluation was conducted with the assistance of a qualitative software program called NVivo. The NVivo program served as the database for the empirical evidence used in the case study analysis. Each of the 17

journal articles in the reference section and the results of the interview questionnaires were entered into the NVivo tool. The hierarchical structure of the FSE Framework was also entered into NVivo where it became a *node tree*. The hierarchical node tree would serve as the collection point for the empirical facts *coded* from the journal articles.

The researcher manually *coded* each of the 17 journal articles and questionnaires. Coding is the term used for a specific type of analysis; the analysis that involved differentiating and combining data associated with the phenomena under investigation. In this case the phenomenon under investigation was the performance of the DIA BHS software development project; and the data was the empirical evidence in the journal articles and questionnaires. The researcher reviewed all of the empirical evidence and selected relevant *chunks* of varying size – words, phrases, sentences, or whole paragraphs, connected to a specific measurement object in the framework. The *chunks of information* were called free nodes. 385 free nodes were coded from the journal articles (177 nodes) and questionnaires (208 nodes).

The hierarchical node tree (the FSE framework) served as the collection point for the free nodes (the *coded* empirical facts). The node tree had 60 collection points, which corresponded with each of the 60 FSE Framework measurement objects. The 385 free nodes were moved to relevant positions on the hierarchical node tree. The completed node tree served as the starting point for evaluation of the DIA BHS software development project.

Function Element

The functions element of the FSE framework had five areas that evaluated 26 measurement objects. The areas address software development, improvement & training

processes, infrastructure, life cycle support processes, and management worth a total of 50 points. Each area will be evaluated separately.

1. Development: The development area addressed four measurement objects:

a. Requirements Management (REQM)

Requirements Management (REQM): The process to manage the requirements of the project's products and product components and to identify inconsistencies between those requirements and the project's plans and work products.	
Source of Empirical Evidence	Empirical Evidence
Hickerson (2006)	Changes after the so-called 'freezes' had taken place. Each airline began to submit more changes after the system freezes had supposedly taken place.
Donaldson (2002)	Design of the UA baggage handling system was frozen on 15 May 1992 when PMT assumed responsibility for the integrated baggage system.
Applegate, Montealegre, Knoop & Nelson (1996)	Design of the United baggage system was frozen on May 15, 1992, when the PMT assumed managerial responsibility for the integrated baggage system.
Applegate, Montealegre, Knoop & Nelson (1996)	One of the biggest problems we had was keeping track of all the changes.
Applegate, Montealegre, Knoop & Nelson (1996)	The design was not to be changed beyond a given date and there would be a number of freeze dates for mechanical design, software design, permanent power requirements and the like. The contract made it obvious that both signatory parties were very concerned about the ability to complete. The provisions dealt mostly with all-around access, timely completion of certain areas, provision of permanent power, provision of computer rooms. All these elements were delineated as milestones.
Survey respondent 31828	9. Did the project have a formal requirements management (REQM) process to manage the requirements of the project's products and product components and to identify inconsistencies between those requirements and the project's plans and work products? No
Survey respondent 31954	9. Did the project have a formal requirements management (REQM) process to manage the requirements of the project's products and product components and to identify inconsistencies between those requirements and the project's plans and work products? Response: Yes
Survey respondent 31954	10. How many of the following process elements were performed in accomplishing the requirements management (REQM) process? Response: The REQM transformation process included a formal method for feedback to make adjustments and changes based on information coming to and from the process.
Survey respondent 31960	9. Did the project have a formal requirements management (REQM) process to manage the requirements of the project's products and product components and to identify inconsistencies between those requirements and the project's plans and work products? Response: Yes
Survey respondent 31960	10. How many of the following process elements were performed in accomplishing the requirements management (REQM) process. Response: The REQM transformation process included a formal method for feedback to make adjustments and changes based on information coming to and from the process.
Score	Overall Evaluation
0.5	The software development project performed one REQM process.

Exhibit 8: REQM Evaluation

b. Requirements Development (RD)

Requirements Development (RD): The process which produced and analyzed customer, product, and product-component requirements.	
Source of Empirical Evidence	Empirical Evidence
Donaldson (2002)	Also in January 1993, maintenance tracks were added to permit the Telecars to be serviced without having to lift them off the main tracks. This cost an additional \$912,000.
GAO (1994)	BAE Automated Systems Incorporated was awarded a \$193 million contract to design, build, and test an automated baggage handling system. This system was designed to move baggage from the check-in areas to the aircraft within 20 minutes. However, the baggage handling system installed at the new airport has had many problems and does not yet work satisfactorily.
Gibbs (1994)	Denver's airport planners saddled BAE with \$20 million worth of changes to the design of its baggage system long after construction had begun.
Applegate, Montealegre, Knoop & Nelson (1996)	In 1992, two years into construction, the project's top managers recommended inclusion of an airport-wide integrated baggage-handling system that could dramatically improve the efficiency of luggage delivery. Originally contracted by United Airlines to cover its operations, the system was to be expanded to serve the entire airport. It was expected that the integrated system would improve ground time efficiency, reduce close-out time for hub operations, and decrease time-consuming manual baggage sorting and handling.
Donaldson (2002)	In August 1992 (as an example) UA altered plans for a transfer system for bags changing planes, requesting that BAE eliminate an entire loop of track from Concourse B. (they would operate with one loop rather than two). This saved approximately \$20 million, but required a system redesign.
Donaldson (2002)	In January 1994, UA requested alterations to its odd-size baggage inputs. This cost an additional; \$432,000
Donaldson (2002)	Still in August 1992, additional ski-claim devices and odd-size baggage elevators added in four of the six sections of the terminal, added \$1.6 million to the cost of the system.
Applegate, Montealegre, Knoop & Nelson (1996)	The initial project design did not incorporate an airport-wide baggage system; the airport expected the individual airlines to build their own systems as in most other American airports.
Montealegre & Keil (2000)	To add to the difficulty of measuring progress, IT projects are dynamic and tend to have volatile requirements (Abdel-Hamid and Madnick 1991; Zmud 1980) that cause project scope to change frequently.
Applegate, Montealegre, Knoop & Nelson (1996)	To further complicate matters, the airlines began requesting changes to the system's design even though the mechanical and software designs were supposed to be frozen. "Six months prior to opening the airport," Di Fonso recalled, "we were still moving equipment around, changing controls, changing software design."
Rifkin (1994)	United, wanting to reduce its share of the costs, decided to remove an entire loop from its own ambitious design for Concourse B. Rather than have two complete loops of track, United would have one. That cut the contract by \$20 million and required a redesign of the system.
Kerzner (2000)	The designer of the baggage handling system was asked to reexamine the number of bags per minute that the BAE system was required to accommodate as per the specifications. The contract called for departing luggage to Concourse A to be delivered at a peak rate of 90 bags per minute. The designer estimated peak demand at 25 bags per minute. Luggage from Concourse A was contracted for at 223 bags per minute but again, the designer calculated peak demand at a lower rate of 44 bags per minute.
Survey respondent 31828	11. Did the project have a formal Requirements Development (RD) process

	which produced and analyzed customer, product, and product-component requirements? No
Survey respondent 31954	11. Did the project have a formal Requirements Development (RD) process which produced and analyzed customer, product, and product-component requirements? Response: Yes
Survey respondent 31954	12. How many of the following process elements were performed in accomplishing the requirements development (RD) process? Response: The RD transformation process included a formal method for feedback to make adjustments and changes based on information coming to and from the process.
Survey respondent 31960	11. Did the project have a formal Requirements Development (RD) process which produced and analyzed customer product and product-component requirements? Response: Yes
Survey respondent 31960	12. How many of the following process elements were performed in accomplishing the requirements development (RD) process? Response 1: The RD transformation process formally designated, received, screened and processed inputs. Response 2: The RD transformation process formally assessed and processed outputs. Response 3: The RD transformation process included a formal method for feedback to make adjustments and changes based on information coming to and from the process.
Score	Overall Evaluation
0.5	The software development project performed one RD process.

Exhibit 9: RD Evaluation

c. Technical Solution (TS)

Technical Solution (TS): The process to design, develop, and implement solutions to requirements.	
Source of Empirical Evidence	Empirical Evidence
Auguston (1994)	Industry has a habit of making the mechanical choices first and then thinking that they can add complexity later through the software and controls, says Dave Tapley, a controls expert with more than 25 years experience in the design and integration of highly automated systems.
Auguston (1994)	It's a typical decision analysis problem," says de Neufville. "You can spend \$200 million on a system without simulation and be uncertain that it will work. Or for less than a million upfront, you can do a simulation and increase your chances of success." In order to anticipate and eliminate any of the various conditions that can make a complex system function unreliably, de Neufville stresses that it is necessary to perform very detailed simulations over a wide range of situations. "A cursory model does not expose the weak links in the system, which is the whole point of doing a simulation in the first place."
Auguston (1994)	These are major research undertakings," claims de Neufville. "Normally, simulating a system like the one at the Denver Airport would take months and months of effort. But then it's a whole lot cheaper to find out what your problems are on paper before you start cutting any metal."
Auguston (1994)	What the layout doesn't reflect, however, is the series of complicated and highly variable vehicle routings that must be managed by the system's control logic software.
Montealegre & Keil (2000)	In April 1992, after viewing a prototype of the proposed system, Denver officials awarded BAE a \$175.6 million contract to build the automated baggage system for the entire airport.
Survey respondent 31828	13. Did the project have a formal Technical Solution (TS) process to design, develop, and implement solutions to requirements? No
Survey respondent 31954	13. Did the project have a formal Technical Solution (TS) process to design, develop, and implement solutions to requirements? Response: Yes.

Survey respondent 31954	14. How many of the following process elements were performed in accomplishing the Technical Solution (TS) process? Response 1: The TS transformation process formally assessed and processed outputs. Response 2: The TS transformation process included a formal method for feedback to make adjustments and changes based on information coming to and from the process.
Survey respondent 31960	13. Did the project have a formal Technical Solution (TS) process to design, develop, and implement solutions to requirements? Response: Yes
Survey respondent 31960	14. How many of the following process elements were performed in accomplishing the Technical Solution (TS) process? Response 1: The TS transformation process formally designated, received, screened and processed inputs. Response 2: The TS transformation process formally assessed and processed outputs. Response 3: The TS transformation process included a formal method for feedback to make adjustments and changes based on information coming to and from the process.
Score	Overall Evaluation
1.0	The software development project performed two TS processes.

Exhibit 10: TS Evaluation**d. Product Integration (PI)**

Product Integration (PI): The process that assembled the product from the product components, ensured that the product, as integrated, functioned properly, and delivered the product.	
Source of Empirical Evidence	Empirical Evidence
Rifkin (1994)	Di Fonso claims that the complicated system is essentially ready, both mechanically and digitally; it simply needs more time to be tested and debugged. It wasn't until the end of April that BAE had its first opportunity to run the entire system as unit. Not surprisingly for a system this complex, it had many bugs.
Gibbs (1994)	errors in the software that controls its automated baggage system. Scheduled for takeoff by last Halloween, the airport's grand opening was postponed until December to allow BAE Automated Systems time to flush the gremlins out of its \$193-million system.
De Neufville (1994)	Getting a control system to deal effectively with the line balancing problem requires extensive testing for the specific loads prevailing at a site.
Rifkin (1994)	While the Munich airport was being built, the project's technical advisors told their Denver counterparts that they had spent two years testing the system. In addition, they said that the system was up and running 24 hours a day for six months before the airport opened.
Survey respondent 31828	15. Did the project have a formal Product Integration (PI) process that assembled the product from the product components, ensured that the product, as integrated, functioned properly, and delivered the product? Yes
Survey respondent 31828	16. How many of the following process elements were performed in accomplishing the Product Integration (PI) process? Response 1: The PI transformation process formally designated, received, screened and processed inputs. Response 2: The PI transformation process formally assessed and processed outputs.
Survey respondent 31954	15. Did the project have a formal Product Integration (PI) process that assembled the product from the product components, ensured that the product, as integrated, functioned properly, and delivered the product? Response: Yes.
Survey respondent 31954	16. How many of the following process elements were performed in accomplishing the Product Integration (PI) process? Response: The PI transformation process included a formal method for feedback to make adjustments and changes based on information coming to and from the process.

Survey respondent 31960	15. Did the project have a formal Product Integration (PI) process that assembled the product from the product components, ensured that the product, as integrated, functioned properly, and delivered the product? Response: Yes
Survey respondent 31960	16. How many of the following process elements were performed in accomplishing the Product Integration (PI) process? Response 1: The PI transformation process formally designated, received, screened and processed inputs. Response 2: The PI transformation process formally assessed and processed outputs. Response 3: The PI transformation process included a formal method for feedback to make adjustments and changes based on information coming to and from the process.
Score	Overall Evaluation
1.5	The software development project performed two PI processes.

Exhibit 11: PI Evaluation

2. Software Improvement & Training Processes: The software improvement and training process area addressed five measurement objects:

a. Organizational Process Focus (OPF)

Organizational Process Focus (OPF) : The process that planned and implemented organizational process improvement based on a thorough understanding of the strengths and weaknesses of the organization's processes and process assets.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	17. Did the project have an Organizational Process Focus (OPF) process that planned and implemented organizational process improvement based on a thorough understanding of the strengths and weaknesses of the organization's processes and process assets? No
Survey respondent 31954	17. Did the project have an Organizational Process Focus (OPF) process that planned and implemented organizational process improvement based on a thorough understanding of the strengths and weaknesses of the organization's processes and process assets? No
Survey respondent 31960	17. Did the project have an Organizational Process Focus (OPF) process that planned and implemented organizational process improvement based on a thorough understanding of the strengths and weaknesses of the organization's processes and process assets? Response: No
Score	Overall Evaluation
0.0	The software development project did not perform the OPF process.

Exhibit 12: OPF Evaluation

b. Organizational Process Design (OPD)

Organizational Process Design (OPD) : The process to establish and maintain a usable set of organizational process assets?	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	19. Did the project have a process that performed the Organizational Process Design (OPD) by establishing and maintaining a usable set of organizational process assets? No
Survey respondent 31954	19. Did the project have a process that performed the Organizational Process Design (OPD) by establishing and maintaining a usable set of organizational

	process assets? No
Survey respondent 31960	19. Did the project have a process that performed the Organizational Process Design (OPD) by establishing and maintaining a usable set of organizational process assets? Response: Yes
Survey respondent 31960	20. How many of the following process elements were performed in accomplishing the Organizational Process Design (OPD) process? Response 1: The OPD transformation process formally designated, received, screened and processed inputs. Response 2: The OPD transformation process formally assessed and processed outputs. Response 3: The OPD transformation process included a formal method for feedback to make adjustments and changes based on information coming to and from the process.
Score	Overall Evaluation
0.0	The software development project did not perform the OPD process.

Exhibit 13: OPD Evaluation

c. Organizational Process Performance (OPP)

Organizational Process Performance (OPP): The process to establish and maintain a quantitative understanding of the performance of the organization's set of standard processes in support of quality and process-performance objectives, and provide the process performance data, baselines, and models to quantitatively manage the project.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	21. Did the project perform the Organizational Process Performance (OPP) process to establish and maintain a quantitative understanding of the performance of the organization's set of standard processes in support of quality and process-performance objectives, and provide the process performance data, baselines, and models to quantitatively manage the project? No
Survey respondent 31954	21. Did the project perform the Organizational Process Performance (OPP) process to establish and maintain a quantitative understanding of the performance of the organization's set of standard processes in support of quality and process-performance objectives, and provide the process performance data, baselines, and models to quantitatively manage the project? Yes
Survey respondent 31954	22. How many of the following process elements were performed in accomplishing the Organizational Process Performance (OPP) process? Response: The OPP transformation process included a formal method for feedback to make adjustments and changes based on information coming to and from the process.
Survey respondent 31960	21. Did the project perform the Organizational Process Performance (OPP) process to establish and maintain a quantitative understanding of the performance of the organization's set of standard processes in support of quality and process-performance...Response: No.
Score	Overall Evaluation
0.0	The software development project did not perform the OPP process.

Exhibit 14: OPP Evaluation

d. Organizational Innovation and Deployment (OID)

Organizational Innovation & Deployment (OID): The process to select and deploy incremental and innovative improvements that measurably improve the organization's processes and technologies.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	23. Did the project perform the Organizational Innovation and Deployment (OID) process to select and deploy incremental and innovative improvements that measurably improve the organization's processes and technologies? No
Survey respondent 31954	23. Did the project perform the Organizational Innovation and Deployment (OID) process to select and deploy incremental and innovative improvements that measurably improve the organization's processes and technologies? Yes
Survey respondent 31954	24. How many of the following process elements were performed in accomplishing the Organizational innovation and Deployment (OID) process? Response: The OID transformation process included a formal method for feedback to make adjustments and changes based on information coming to and from the process.
Survey respondent 31960	23. Did the project perform the Organizational Innovation and Deployment (OID) process to select and deploy incremental and innovative improvements that measurably improve the organization's processes and technologies? Response: No
Score	Overall Evaluation
0.0	The software development project did not perform the OID process.

Exhibit 15: OID Evaluation

e. Organizational Training (OT)

Organizational Training (OT): The process to develop the skills and knowledge of people so they could perform their roles effectively and efficiently.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	55. Did the project have a formal Organizational Training (OT) process to develop the skills and knowledge of people so they could perform their roles effectively and efficiently? No
Survey respondent 31954	55. Did the project have a formal Organizational Training (OT) process to develop the skills and knowledge of people so they could perform their roles effectively and efficiently? No
Survey respondent 31960	55. Did the project have a formal Organizational Training (OT) process to develop the skills and knowledge of people so they could perform their roles effectively and efficiently? Response: No
Score	Overall Evaluation
0.0	The software development project did not perform the OT process.

Exhibit 16: OT Evaluation

3. Software Project Infrastructure: The software development project infrastructure area addressed two measurement objects:

a. Use of Software Tools (TOOL)

Use of Software Tools (TOOL): The capability and integration of the tool suite used in developing the software on the project.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	25. Which statement best characterizes the software tool capability and integration of the tool suite used in developing the software on the project? The software development project tools were edit, code, debug with no integration.
Survey respondent 31954	25. Which statement best characterizes the software tool capability and integration of the tool suite used in developing the software on the project? Response: The software development project tools were strong, mature, proactive life-cycle tools that were well integrated with processes, methods and reuse.
Survey respondent 31960	25. Which statement best characterizes the software tool capability and integration of the tool suite used in developing the software on the project? Response: The software development project tools were basic life-cycle tools, with moderate integration.
Score	Overall Evaluation
0.6	Nominal: The software development project tools were basic life-cycle tools, with moderate integration.

Exhibit 17: TOOL Evaluation

b. Multi-site Development (SITE)

Multi-site Development (SITE): The software development project team's co-location and communications profile?	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	26. Which statement best characterizes the software development project team's co-location and communications profile? [Not Answered]
Survey respondent 31954	26. Which statement best characterizes the software development project team's co-location and communications profile? Response: The software development site was multi-city and multi-company and used individual phone and facsimile for communications.
Survey respondent 31960	26. Which statement best characterizes the software development project team's co-location and communications profile? Response: The software development site was in the same city or metro area and used wide band electronic communications.
Score	Overall Evaluation
0.4	Nominal. The software development site was multi-city and used narrow band e-mail for communications.

Exhibit 18: SITE Evaluation

4. Software Life Cycle Support: The software life cycle support area addressed eight measurement objects:

a. Process and Product Quality Assurance (PPQA)

Process & Product QA (PPQA): The process to provide staff and management with objective insight into processes and associated work products.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	27. Did the project have a formal Process and Product Quality Assurance (PPQA) process to provide staff and management with objective insight into processes and associated work products? No
Survey respondent 31954	27. Did the project have a formal Process and Product Quality Assurance (PPQA) process to provide staff and management with objective insight into processes and associated work products? Response: No
Survey respondent 31960	27. Did the project have a formal Process and Product Quality Assurance (PPQA) process to provide staff and management with objective insight into processes and associated work products? Response: No
Score	Overall Evaluation
0.0	The software development project did not perform the PPQA process.

Exhibit 19: PPQA Evaluation

b. Configuration Management (CM)

Configuration Management (CM): The process that established and maintained the integrity of work products using configuration identification, configuration control, configuration status accounting, and configuration audits.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	29. Did the project have a formal Configuration Management (CM) process that established and maintained the integrity of work products using configuration identification, configuration control, configuration status accounting, and configuration audits? No
Survey respondent 31954	29. Did the project have a formal Configuration Management (CM) process that established and maintained the integrity of work products using configuration identification, configuration control, configuration status accounting, and configuration audits? Response: No
Survey respondent 31960	29. Did the project have a formal Configuration Management (CM) process that established and maintained the integrity of work products using configuration identification, configuration control, configuration status accounting, and configuration audits? Response: No
Score	Overall Evaluation
0.0	The software development project did not perform the CM process.

Exhibit 20: CM Evaluation

c. Measurement & Analysis (MA)

Measurement & Analysis (MA): The process to develop and sustain a measurement capability used to support management information needs.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	31. Did the project have a formal Measurement & Analysis (MA) process to develop and sustain a measurement capability used to support management information needs? No
Survey respondent 31954	31. Did the project have a formal Measurement & Analysis (MA) process to develop and sustain a measurement capability used to support management

	information needs? Response: Yes
Survey respondent 31954	32. How many of the following process elements were performed in accomplishing the Measurement & Analysis (MA) process? Response: The MA transformation process included a formal method for feedback to make adjustments and changes based on information coming to and from the process.
Survey respondent 31960	31. Did the project have a formal Measurement & Analysis (MA) process to develop and sustain a measurement capability used to support management information needs? Response: No
Score	Overall Evaluation
0.0	The software development project did not perform the MA process.

Exhibit 21: MA Evaluation**d. Verification (VER)**

Verification (VER): The process to ensure that selected work products met their specified requirements.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	33. Did the project have a formal Verification (VER) process to ensure that selected work products met their specified requirements? No
Survey respondent 31954	33. Did the project have a formal Verification (VER) process to ensure that selected work products met their specified requirements? Response: Yes
Survey respondent 31954	34. How many of the following process elements were performed in accomplishing the Verification (VER) process? Response: The VER transformation process formally assessed and processed outputs. Response: The VER transformation process included a formal method for feedback to make adjustments and changes based on information coming to and from the process.
Survey respondent 31960	33. Did the project have a formal Verification (VER) process to ensure that selected work products met their specified requirements? Response: Yes
Survey respondent 31960	34. How many of the following process elements were performed in accomplishing the Verification (VER) process? Response: The VER transformation process included a formal method for feedback to make adjustments and changes based on information coming to and from the process.
Score	Overall Evaluation
0.5	The software development project performed one VER process.

Exhibit 22: VER Evaluation**e. Validation (VAL)**

Validation (VAL): The process to demonstrate that a product or product component fulfilled its intended use when placed in its intended environment.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	35. Did the project have a formal Validation (VAL) process to demonstrate that a product or product component fulfilled its intended use when placed in its intended environment? No
Survey respondent 31954	35. Did the project have a formal Validation (VAL) process to demonstrate that a product or product component fulfilled its intended use when placed in its intended environment? Response: Yes
Survey respondent 31954	36. How many of the following process elements were performed in accomplishing the Validation (VAL) process? Response: The VAL transformation process included a formal method for feedback to make

	adjustments and changes based on information coming to and from the process.
Survey Respondent 31960	35. Did the project have a formal Validation (VAL) process to demonstrate that a product or product component fulfilled its intended use when placed in its intended environment? Response: Yes
Survey Respondent 31960	36. How many of the following process elements were performed in accomplishing the Validation (VAL) process? Response 1: The VAL transformation process formally designated, received, screened and processed inputs. Response 2: The VAL transformation process formally assessed and processed outputs. Response 3: The VAL transformation process included a formal method for feedback to make adjustments and changes based on information coming to and from the process.
Score	Overall Evaluation
0.5	The software development project performed one VAL process.

Exhibit 23: VAL Evaluation

f. Causal Analysis and Resolution (CAR)

Causal Analysis & Resolution (CAR): The process to identify causes of defects and other problems and take action to prevent them from occurring in the future.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	37. Did the project have a formal Causal Analysis & Resolution (CAR) process to identify causes of defects and other problems and take action to prevent them from occurring in the future? No
Survey respondent 31954	37. Did the project have a formal Causal Analysis & Resolution (CAR) process to identify causes of defects and other problems and take action to prevent them from occurring in the future? Response: Yes
Survey respondent 31954	38. How many of the following process elements were performed in accomplishing the Causal Analysis & Resolution (CAR) process? Response: The CAR transformation process included a formal method for feedback to make adjustments and changes based on information coming to and from the process.
Survey Respondent 31960	37. Did the project have a formal Causal Analysis & Resolution (CAR) process to identify causes of defects and other problems and take action to prevent them from occurring in the future? Response: No
Score	Overall Evaluation
0.0	The software development project did not perform the CAR process.

Exhibit 24: CAR Evaluation

g. Joint Reviews (JR)

Joint Reviews (JR): The process in which the customer and the project team evaluate the status and products of an activity as appropriate, were utilized on the software development project.	
Source of Empirical Evidence	Empirical Evidence
Rifkin (1994)	The city has concluded that it can no longer trust BAE's judgment on the status of the programming.
Keil & Montealegre (2000)	Clearly, the automated baggage system now underway at DIA is not yet at a level that meets the requirements of the city, the airlines, or the traveling public,
Survey respondent 31828	39. How would you characterize the extent to which Joint Reviews, in which the customer and the project team evaluate the status and products of an activity as appropriate, were utilized on the software development project?

	Only joint reviews required by contract were conducted.
Survey respondent 31954	39. How would you characterize the extent to which Joint Reviews, in which the customer and the project team evaluate the status and products of an activity as appropriate, were utilized on the software development project? Response: Joint reviews were used in a few process areas.
Survey respondent 31960	39. How would you characterize the extent to which Joint Reviews, in which the customer and the project team evaluate the status and products of an activity as appropriate, were utilized on the software development project? Response: Joint reviews were used in a few process areas.
Score	Overall Evaluation
1.0	Nominal: Joint reviews were used in a few process areas.

Exhibit 25: JR Evaluation

h. External Auditing (EA)

External Auditing (EA): The process, in which an external management entity determines compliance with requirements, plans, and contract as appropriate, were utilized on the software development project.	
Source of Empirical Evidence	Empirical Evidence
Rifkin (1994)	It has brought in a materials-handling firm with strong systems integration expertise, Frankfort, Germany's Logplan, to monitor and review BAE's work.
Hickerson (2006)	many of the managers involved convinced themselves that "things did not look so bad" and continued down the same project path despite warning signs.
Hickerson (2006)	Struggles between the airport and BAE after the delays started. Soon after the disaster and the public demo, the City of Denver hired a German consulting company, Logplan, who began to audit BAE's work and issued an independent report on what could be fixed in a short period of time.
Keil & Montealegre (2000)	In both the DIA and Taurus cases, hiring an external consultant was instrumental in assessing the severity of the problem and in helping the executives responsible for these projects to extricate themselves and their organizations from failing courses of action.
Survey respondent 31828	40. How would you characterize the extent to which external audits, in which an external management entity determines compliance with requirements, plans, and contract as appropriate, were utilized on the software development project? No external audits were conducted.
Survey respondent 31954	40. How would you characterize the extent to which external audits, in which an external management entity determines compliance with requirements, plans, and contract as appropriate, were utilized on the software development project? Response: External audits were used in a few process areas.
Survey respondent 31960	40. How would you characterize the extent to which external audits, in which an external management entity determines compliance with requirements, plans, and contract as appropriate, were utilized on the software development project? Response: External audits were used in a few process areas.
Score	Overall Evaluation
1.0	Nominal. External audits were used in a few process areas.

Exhibit 26: EA Evaluation

5. Software Project Management: The software management area addressed seven measurement objects:

a. Project Planning (PP)

Project Planning (PP): The process that established and maintained plans that defined project activities.	
Source of Empirical Evidence	Empirical Evidence
Donaldson (2002)	BAE arrived at the scene with fully designed specs which obviously in the long run proved to be a major planning error.
GAO (1994)	Because the airport is so large, airport planners decided that a state-of-the-art automated baggage handling system, capable of moving bags much more quickly than conventional tug-and-cart/conveyor belt systems, would be needed.
De Neufville (1994)	Despite the central importance of the automated baggage system, its design was largely an afterthought. This is a common practice, unfortunately. The Denver system was detailed well after the construction of the airport was under way and only about two years before the airport was to open.
Rifkin (1994)	Evans says that the city did receive bids from three companies, all eager to win the contract. However, Breier, Neidle and Patrone, a New York City-based consulting firm, recommended against the designs, saying the configurations would not meet the airport's needs.
Rifkin (1994)	For \$195 million, BAE would build for the entire airport an expanded version of the system it was constructing for United. "We placed a number of conditions on accepting the job," Di Fonso says. "The design doesn't change beyond this date, and there would be a number of freeze dates for mechanical design, software design, permanent power requirements and the like."
GAO (1994)	It was designed to provide the high-speed transfer of baggage to and from aircraft, thereby facilitating quick turnaround times for aircraft and improved services to passengers. Baggage will travel between the terminal and concourses through interconnecting tunnels. The most distant concourse is located about a mile from the terminal.
Keil & Montealegre (2000)	Not until 1992, two years into the construction of the new airport, did the project's top managers recognize the potential benefits of an airport-wide, IT-based baggage-handling system. As a result, airport planners and consultants developed specifications for such a system.
Goetz & Szyliowicz (1997)	Originally scheduled to open in October 1993, DIA finally opened on February 28, 1995 after four postponements, principally because of difficulties with its automated baggage system. Originally projected to cost \$1.7 billion, its price had escalated to over \$5 billion.
Goetz & Szyliowicz (1997)	The decision to build DIA and the planning process responsible for its realization are typical of the ways in which transportation projects are designed and implemented. Traditional practices are based on a decision-making model and a procedural theory of planning which, because of its assumptions and requirements, frequently does not yield desirable results. Although scholars have demonstrated persuasively that it possesses severe shortcomings (Hirschman, 1967; Rondinelli, 1977; Rycroft and Szyliowicz, 1980), the 'rational comprehensive model' of planning and decision-making remains the dominant planning paradigm. Its influence upon transportation planners dates back many decades and, despite its acknowledged deficiencies, continues to shape the ways in which the craft is practiced (Wachs, 1985).
Donaldson (2002)	The Project Management Team (PMT) became responsible for overseeing planning and
Survey respondent 31828	41. Did the project have a formal Project Planning (PP) process that established and maintained plans that defined project activities? Yes
Survey respondent 31828	42. How many of the following process elements were performed in accomplishing the Project Planning (PP) process? PP was satisfactorily

	performed as specified in the CMM/CMMI.
Survey respondent 31954	41. Did the project have a formal Project Planning (PP) process that established and maintained plans that defined project activities? Response: Yes
Survey respondent 31954	42. How many of the following process elements were performed in accomplishing the Project Planning (PP) process? Response: The PP transformation process included a formal method for feedback to make adjustments and changes based on information coming to and from the process.
Survey respondent 31960	41. Did the project have a formal Project Planning (PP) process that established and maintained plans that defined project activities? Response: Yes
Survey respondent 31960	42. How many of the following process elements were performed in accomplishing the Project Planning (PP) process? Response: The PP transformation process included a formal method for feedback to make adjustments and changes based on information coming to and from the process.
Score	Overall Evaluation
0.5	The software development project performed one PP process.

Exhibit 27: PP Evaluation

b. Project Monitoring and Control (PMC)

Project Monitoring & Control (PMC): The process to provide an understanding of the project's progress so that appropriate corrective actions could be taken when the project's performance deviated significantly from the plan.	
Source of Empirical Evidence	Empirical Evidence
Montealegre & Keil (2000)	Almost certainly, projects that exhibit such volatility are especially difficult to manage and control.
Rifkin (1994)	Evans claims that BAE didn't pay enough attention to the programming issues early enough in the process.
Keil & Montealegre (2000)	The DIA case teaches that the visibility of project costs can play an important role in promoting problem redefinition.
Montealegre & Keil (2000)	The intangible nature of software makes it difficult to obtain accurate estimates of the proportion of work completed, which may promote escalation of commitment by giving a false perception that successful completion of the project is near.
Donaldson (2002)	The tracking system was a disaster. They tried to merge the different systems into one central database structure. Everybody had their own and it took 3 years to get working.
Survey respondent 31828	43. Did the project have a formal Project Monitoring & Control (PMC) process to provide an understanding of the project's progress so that appropriate corrective actions could be taken when the project's performance deviated significantly from the plan? Yes
Survey respondent 31828	44. How many of the following process elements were performed in accomplishing the Project Monitoring & Control (PMC) process? PMC was satisfactorily performed as specified in the CMM/CMMI.
Survey respondent 31954	43. Did the project have a formal Project Monitoring & Control (PMC) process to provide an understanding of the project's progress so that appropriate corrective actions could be taken when the project's performance deviated significantly from the plan? Response: Yes
Survey respondent 31954	44. How many of the following process elements were performed in accomplishing the Project Monitoring & Control (PMC) process? Response 1: The PMC transformation process included a formal method for feedback to make adjustments and changes based on information coming to and from the process. Response 2: The PMC transformation process included a formal method for feedback to make adjustments and changes based on information

	coming to and from the process.
Survey Respondent 31960	43. Did the project have a formal Project Monitoring & Control (PMC) process to provide an understanding of the project's progress so that appropriate corrective actions could be taken when the project's performance deviated significantly from the plan? Response: Yes
Survey Respondent 31960	44. How many of the following process elements were performed in accomplishing the Project Monitoring & Control (PMC) process? Response 1: The PMC transformation process formally designated, received, screened and processed inputs. Response 2: The PMC transformation process formally assessed and processed outputs. Response 3: The PMC transformation process included a formal method for feedback to make adjustments and changes based on information coming to and from the process.
Score	Overall Evaluation
1.0	The software development project performed two PMC processes.

Exhibit 28: PMC Evaluation

c. Supplier Agreement Management (SAM)

Supplier Agreement Management (SAM): The process to manage the acquisition of products or services from suppliers for which there were formal agreements.	
Source of Empirical Evidence	Empirical Evidence
Donaldson (2002)	Five out of sixty contracts awarded for the design of DIA went to Denver companies. - The 60 contracts generated 110 construction contracts and 88 professional service contracts. - Between 200 and 300 (max. 400) companies were involved
Hickerson (2006)	In April of 1992, BAE was awarded the contract of \$175.6 million to proceed with design and construction of an airport-wide CBHS. Gene Di Fonso, President of BAE, recalled later about the sessions with airport management to nail down deadlines and freezes to the plan; "We placed a number of conditions on accepting the job... The design was not to be changed beyond a given date and there would be a number of freeze dates for mechanical design, software design, permanent power designs and the like."
Survey respondent 31828	45. Did the project have a formal Supplier Agreement Management (SAM) process to manage the acquisition of products or services from suppliers for which there were formal agreements? No
Survey respondent 31954	45. Did the project have a formal Supplier Agreement Management (SAM) process to manage the acquisition of products or services from suppliers for which there were formal agreements? Response: Yes
Survey respondent 31954	46. How many of the following process elements were performed in accomplishing the Supplier Agreement Management (SAM) process? Response: The SAM transformation process included a formal method for feedback to make adjustments and changes based on information coming to and from the process
Survey Respondent 31960	45. Did the project have a formal Supplier Agreement Management (SAM) process to manage the acquisition of products or services from suppliers for which there were formal agreements? Response: No
Score	Overall Evaluation
0.0	The software development project did not perform the SAM process.

Exhibit 29: SAM Evaluation

d. Risk Management (RSKM)

Supplier Agreement Management (SAM): The process to manage the acquisition of products or services from suppliers for which there were formal agreements.	
Source of Empirical Evidence	Empirical Evidence
Goetz & Szyliowicz (1997)	A second major factor that falls into the 'risk' category involves technology; the newer and the more untested the technology, the more likely is one to encounter problems. This was obviously true of the baggage system.
Kerzner (2000)	From a project management perspective, there was no question that disaster was on the horizon. Nobody knew what to do about the DCV system. The risks were unknown. Nobody realized the complexity of the system, especially the software requirements. By one account, the launch date should have been delayed by at least two years. The contract for DCV hadn't been awarded yet, and terminal construction was already under way. Everyone wanted to know why the design (and construction) was not delayed until after the airlines had signed on. How could DIA install and maintain the terminal's baseline design without having a design for the baggage handling system? Everyone felt that what they were now building would have to be ripped apart.
Szyliowicz & Goetz (1995)	Perhaps the most serious assumption involved the totally automated baggage handling system, demanded by United, which was to force additional modifications and significantly delay the airport's opening. The planners apparently never considered the problems involved in greatly increasing the scale of a complex technology, especially one so dependent upon the development of a large scale software system of a type where failures are commonplace (Gibbs, 1994).
de Neufville (1994)	Remarkably, the design of the fully automated baggage system at Denver did not include a meaningful backup system. The planners provided neither a fleet of tugs and carts that could cope with the level of baggage expected, nor even access roads between the check-in facilities and the aircraft.
Goetz & Szyliowicz (1997)	Since the baggage system is a critical component of any airport, the decision not to hedge represented a remarkable gamble. If a reasonable time frame (an extension of two to three years has been mentioned)
Survey respondent 31828	47. Did the project have a formal Risk Management (RSKM) process to identify potential problems before they occur, so that risk-handling activities may be planned and invoked as needed across the life of the product or project to mitigate adverse impacts on achieving objectives? No
Survey respondent 31954	47. Did the project have a formal Risk Management (RSKM) process to identify potential problems before they occur, so that risk-handling activities may be planned and invoked as needed across the life of the product or project to mitigate adverse...Response: No
Survey respondent 31960	47. Did the project have a formal Risk Management (RSKM) process to identify potential problems before they occur, so that risk-handling activities may be planned and invoked as needed across the life of the product or project to mitigate adverse...Response: No
Score	Overall Evaluation
0.0	The software development project did not perform the RSKM process.

Exhibit 30: RSKM Evaluation

e. Integrated Project Management (IPM)

Integrated Project Management (IPM): The process to establish and manage the project and the involvement of the relevant stakeholders according to an integrated and defined process.	
Source of Empirical Evidence	Empirical Evidence
Donaldson (2002)	The City of Denver staff and the Consultant team shared leadership of the project and coordinated initial facets of the design.
Applegate, Montealegre, Knoop & Nelson (1996)	Gene Di Fonso, President of BAE, knew that his company could demonstrate that flaws in the overall design of the airport and an unsystematic approach to project changes had affected implementation of the integrated baggage system.
Russell (1994)	Responsibilities were not clearly defined. The muddled chain of command was named by all the designers as the most frustrating aspect of the project. Bradburn faults the city, which he says "infiltrated" the PMT with its own staff in such a way as to inappropriately undercut the authority of Greiner/MKE personnel nominally in positions of higher responsibility.
Russell (1994)	With its mega-project expertise, Greiner and Morrison-Knudsen should have been able to handle the enormous scheduling and coordination issues that came up as the scale of changes enlarged. Coordination, however, was not the responsibility of the PMT, but of the designers. Designers say their firms were not structured to take on coordination at the scale ultimately require.
Survey respondent 31828	49. Did the project have a formal Integrated Project Management (IPM) process to establish and manage the project and the involvement of the relevant stakeholders according to an integrated and defined process? No
Survey respondent 31954	49. Did the project have a formal Integrated Project Management (IPM) process to establish and manage the project and the involvement of the relevant stakeholders according to an integrated and defined process? Response: No
Survey respondent 31960	49. Did the project have a formal Integrated Project Management (IPM) process to establish and manage the project and the involvement of the relevant stakeholders according to an integrated and defined process? Response: Yes
Survey respondent 31960	50. How many of the following process elements were performed in accomplishing the Integrated Project Management (IPM) process? Response 1: The IPM transformation process formally designated, received, screened and processed inputs. Response 2: The IPM transformation process formally assessed and processed outputs. Response 3: The IPM transformation process included a formal method for feedback to make adjustments and changes based on information coming to and from the process.
Score	Overall Evaluation
0.5	The software development project performed one IPM process.

Exhibit 31: IPM Evaluation

f. Decision Analysis and Resolution (DAR)

Decision Analysis & Resolution (DAR): The process which analyzed possible decisions using a formal evaluation process that evaluated identified alternatives against established criteria.	
Source of Empirical Evidence	Empirical Evidence
Hickerson (2006)	it did not seem like there were any feasible alternatives;
GAO (1995)	The City was advised early on by several consultants that building the automated system was a high-risk proposition, especially within the time frames allowed. The City disregarded these opinions and has paid a high price for this decision. In DIA's case, it would have been cheaper to plan for and build an alternative system from the start rather than deciding to install one after major problems surfaced.

Goetz & Szyliowicz (1997)	Preparing for the unexpected might involve a series of 'go/no go' checkpoints, whereby at specific decision stages the situation would be evaluated anew and decisions made on the basis of existing conditions and new information. This strategy is inherent in much of the literature that advocates flexibility and incrementalism (Collingridge, 1992; Hayes, 1992; Weiss and Woodhouse, 1992).
Survey respondent 31828	51. Did the project have a formal Decision Analysis & Resolution (DAR) process which analyzed possible decisions using a formal evaluation process that evaluated identified alternatives against established criteria? No
Survey respondent 31954	51. Did the project have a formal Decision Analysis & Resolution (DAR) process which analyzed possible decisions using a formal evaluation process that evaluated identified alternatives against established criteria? Response: Yes
Survey respondent 31954	52. How many of the following process elements were performed in accomplishing the Decision Analysis & Resolution (DAR) process? Response: The DAR transformation process formally assessed and processed outputs.
Survey Respondent 31960	51. Did the project have a formal Decision Analysis & Resolution (DAR) process which analyzed possible decisions using a formal evaluation process that evaluated identified alternatives against established criteria? Response: No
Score	Overall Evaluation
0.0	The software development project did not perform the DAR process.

Exhibit 32: DAR Evaluation

g. Quantitative Project Management (QPM)

Quantitative Project Management (QPM): The process which quantitatively managed the project's defined processes to achieve the project's established quality and process-performance objectives.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	53. Did the project have a formal Quantitative Project Management (QPM) process which quantitatively managed the project's defined processes to achieve the project's established quality and process-performance objectives? No
Survey respondent 31954	53. Did the project have a formal Quantitative Project Management (QPM) process which quantitatively managed the project's defined processes to achieve the project's established quality and process-performance objectives? Response: No
Survey respondent 31960	53. Did the project have a formal Quantitative Project Management (QPM) process which quantitatively managed the project's defined processes to achieve the project's established quality and process-performance objectives? Response: No
Score	Overall Evaluation
0.0	The software development project did not perform the QPM process.

Exhibit 33: QPM Evaluation

The overall score for the functions element was 9.0 of 50 possible points.

Structure Element

The structure element of the FSE framework had three areas that evaluated 20 measurement objects related to the structural ability of the project to produce software.

The areas address the social system, the cybernetic controls and the technical system worth a total of 25 points. Each area will be evaluated separately.

1. Social System: The social system area addressed six measurement objects:

a. Analyst Capability (ACAP)

Analyst Capability (ACAP): (1) The analysis and design ability, efficiency, and thoroughness, of the analyst team (those who worked on requirements, high-level design, and detailed design) that worked on the software development project and (2) the analyst team's ability to communicate and cooperate during the software development project.	
Source of Empirical Evidence	Empirical Evidence
de Neufville (1994)	The line-balancing problem is compounded by a general ignorance or disregard for its existence. Even knowledgeable designers and operators of automated systems seem not to focus on this issue.
Survey respondent 31828	57. How would you rate the analysis and design ability, efficiency, and thoroughness, of the analyst team (those who worked on requirements, high-level design, and detailed design) that worked on the software development project? 75th-90th percentile.
Survey respondent 31828	58. How would you rate the analyst team's ability to communicate and cooperate during the software development project? 55th-75th percentile
Survey respondent 31954	57. How would you rate the analysis and design ability, efficiency, and thoroughness, of the analyst team (those who worked on requirements, high-level design, and detailed design) that worked on the software development project? Response: 55th-75th percentile.
Survey respondent 31954	58. How would you rate the analyst team's ability to communicate and cooperate during the software development project? Response: 15th-35th percentile
Survey respondent 31960	57. How would you rate the analysis and design ability, efficiency, and thoroughness of the analyst team (those who worked on requirements, high-level design, and detailed design) that worked on the software development project? Response: 75th-90th percentile.
Survey respondent 31960	58. How would you rate the analyst team's ability to communicate and cooperate during the software development project? Response: 35th-55th percentile
Score	Overall Evaluation
0.6	Nominal: 55 th -75 th percentile

Exhibit 34: ACAP Evaluation

b. Programmer Capability (PCAP)

Programmer Capability (PCAP): (1) The analysis and design ability, efficiency, and thoroughness, and the ability to communicate and cooperate of the programmer team (those who implement processes and functions through software code) that worked on the software development project and (2) the programmer team's ability to communicate and cooperate during the software development project.	
Source of Empirical Evidence	Empirical Evidence
Rifkin (1994)	At the heart of the system is a complicated software program called the Empty Car Management system. The program dispatches empty carts to any input point where they are needed. It also coordinates the flow of carts to ensure that

	empty queues are replenished in time for arriving bags. The complex system was written over the past two years by more than 20 BAE software programmers
de Neufville (1994)	The software must, in addition to the usual error checking codes that guard against electrical disturbances in the communications, have multiple levels of redundancy and be able to recover from errors very rapidly. Getting this right can take many expensive programmers a lot of time (Gibbs, 1994).
Survey respondent 31828	59. How would you rate the analysis and design ability, efficiency, and thoroughness, and the ability to communicate and cooperate of the programmer team (those who implement processes and functions through software code) that worked on the software development project? 55th-75th percentile.
Survey respondent 31828	60. How would you rate the programmer team's ability to communicate and cooperate during the software development project? 55th-75th percentile
Survey respondent 31954	59. How would you rate the analysis and design ability, efficiency, and thoroughness, and the ability to communicate and cooperate of the programmer team (those who implement processes and functions through software code) that worked on the software development project? Response: 35th-55th percentile.
Survey respondent 31954	60. How would you rate the programmer team's ability to communicate and cooperate during the software development project? Response: 35th-55th percentile
Survey respondent 31960	59. How would you rate the analysis and design ability, efficiency, and thoroughness, and the ability to communicate and cooperate of the programmer team (those who implement processes and functions through software code) that worked on the software development project? Response: 55th-75th percentile.
Survey respondent 31960	60. How would you rate the programmer team's ability to communicate and cooperate during the software development project? Response: 35th-55th percentile
Score	Overall Evaluation
0.4	Low: 35 th -55 th percentile.

Exhibit 35: PCAP Evaluation

c. Personnel Continuity (PCON)

Personnel Continuity (PCON): The software development project's annual personnel turnover.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	61. What was the software development project's annual personnel turnover? 3-12% per year
Survey respondent 31954	61. What was the software development project's annual personnel turnover? Response: 13-23% per year
Survey respondent 31960	61. What was the software development project's annual personnel turnover? Response: 13-23% per year
Score	Overall Evaluation
0.6	Nominal: 13-23% per year

Exhibit 36: PCON Evaluation

d. Applications Experience (APEX)

Applications Experience (APEX): The level of applications experience of the team that developed the software system? [Note: This is defined in terms of the development team's equivalent years of experience with this type of application.]	
Source of Empirical Evidence	Empirical Evidence
Donaldson (2002)	BAE had a world-wide reputation as a superior baggage system builder.
Rifkin (1994)	BAE is considered the top baggage system company in the world. It has spent the past 26 years building baggage-handling systems for individual airlines at such airports as Atlanta, San Francisco, Dallas-Fort Worth and JFK in New York.
Kerzner (2000)	BAE underestimated the complexity of the routing problems. During trials, cars crashed into one another, luggage was dropped at the wrong location, cars that were needed to carry luggage were routed to empty waiting pens, and some cars traveled in the wrong direction. Sensors became coated with dirt, throwing the system out of alignment, and luggage was dumped prematurely because of faulty latches, jamming cars against the side of a tunnel. By the end of May, BAE was conducting a worldwide search for consultants who could determine what was going wrong and how long it would take to repair the system.
Hickerson (2006)	BAE was also criticized for overreaching in engineering literature as the delays started to become public knowledge. One reviewer pointed out that, while the computer system did know what to do with the telecars that were full, BAE had no experience with empty cart management software, or in other words, how to maximize where the telecars should go after fulfilling a delivery ¹⁴
Rifkin (1994)	Di Fonso acknowledges that BAE ran into a raft of programming nightmares. One was writing the code for establishing and maintaining communications with the airlines' reservations systems, especially United's Apollo computers. In order for the system to operate, it must be able to "converse" in the software language of each airline. Such translation work is painstaking and laden with bugs.
Rifkin (1994)	In addition, there were repeated problems with the printers at the ticket counters, and bugs in the Empty Car Management system plagued the trials. In early March, when BAE started using a lot of baggage during its tests, it became clear that the software too often was sending a cart out too early or too late.
de Neufville (1994)	the contractor responsible for the installation (BAE Automated Systems) had enjoyed the reputation of being among the best and, on the strength of its good work, has been responsible for most of the major baggage systems recently installed in the United States.
Kerzner (2000)	The problem appeared to be with the software required to get computers to talk to computers. The fact that a mere software failure could hold up Denver's new airport for more than a year put in question the project's risk management program.
Goetz & Szyliowicz (1997)	the software required the writing of millions of lines of computer code which were necessary to direct bags safely and correctly to their destination. Software errors would cause telecars to be misloaded and misdirected, often resulting in spectacular accidents. One such incident was captured by a local television station and broadcast several times to an incredulous public.
Hickerson (2006)	United Airlines, which committed early to the DIA project, was the first to start work on a baggage handling system, aiming for an advanced solution. To this end, they commissioned BAE Automated Systems Inc. to build the CBHS at the new airport. BAE was considered a leading manufacturer of these systems, with a solid track record of past performance.

Rifkin (1994)	using IBM's OS/2 operating system
Survey respondent 31828	62. How would you rate the level of applications experience of the team that developed the software system? [Note: This is defined in terms of the development team's equivalent years of experience with this type of application. >6 years.
Survey respondent 31954	62. How would you rate the level of applications experience of the team that developed the software system? [Note: This is defined in terms of the development team's equivalent years of experience with this type of application.] Response: 1-6 years
Survey respondent 31960	62. How would you rate the level of applications experience of the team that developed the software system? [Note: This is defined in terms of the development team's equivalent years of experience with this type of application.] Response: 1-6 years.
Score	Overall Evaluation
0.8	High: 1-6 years

Exhibit 37: APEX Evaluation

e. Platform Experience (PLEX)

Platform Experience (PLEX): The level of platform experience of the team that developed the software system? [Note: This is defined in terms of the development team's equivalent years of experience with the type of platform (i.e. graphical user interface, database, networking, middleware, etc.).]	
Source of Empirical Evidence	Empirical Evidence
Hickerson (2006)	Baggage handling systems would also run on a client-server system, which meant that central control of the system would be placed outside the control of the airport's information systems department.
de Neufville (1994)	The problem is further complicated at Denver because it uses a distributed system of about 150 computers.
Survey respondent 31828	63. How would you rate the level of platform experience of the team that developed the software system? [Note: This is defined in terms of the development team's equivalent years of experience with the type of platform (i.e. graphical user interface, database, networking, middleware, etc.).] 1-6 years.
Survey respondent 31954	63. How would you rate the level of platform experience of the team that developed the software system? [Note: This is defined in terms of the development team's equivalent years of experience with the type of platform (i.e. graphical user interface, database, networking, middleware, etc.).] Response: >6 years.
Survey respondent 31960	63. How would you rate the level of platform experience of the team that developed the software system? [Note: This is defined in terms of the development team's equivalent years of experience with the type of platform (i.e. graphical user interface, database, networking, middleware, etc.).] Response: 6-12 months.
Score	Overall Evaluation
0.8	High: 1-6 years

Exhibit 38: PLEX Evaluation

f. Language and Tool Experience (LTEX)

Platform Experience (PLEX): The level of platform experience of the team that developed the software system? [Note: This is defined in terms of the development team's equivalent years of experience with the type of platform (i.e. graphical user interface, database, networking, middleware, etc.).]	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	64. How would you rate the level of programming language and software tool experience of the team that developed the software system? [Note: This is defined in terms of the development team's equivalent years of experience with this type of language and toolset.] >6 years.
Survey respondent 31954	64. How would you rate the level of programming language and software tool experience of the team that developed the software system? [Note: This is defined in terms of the development team's equivalent years of experience with this type of language and toolset.] Response: 1-6 years
Survey respondent 31960	64. How would you rate the level of programming language and software tool experience of the team that developed the software system? [Note: This is defined in terms of the development team's equivalent years of experience with this type of language...Response: 1-6 years.
Score	Overall Evaluation
0.8	High: 1-6 years

Exhibit 39: LTEX Evaluation

2. Cybernetic Control: The cybernetic controls area addressed five measurement objects:

a. Control (CTRL)

Control (CTRL): The operational measures the project used most commonly to control the performance of the software development project.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	65. What operational measures did the project use most commonly to control the performance of the software development project? Cost or Schedule data.
Survey respondent 31954	65. What operational measures did the project use most commonly to control the performance of the software development project? Response: Cost and Schedule data.
Survey respondent 31960	65. What operational measures did the project use most commonly to control the performance of the software development project? Response: Cost and Schedule data.
Score	Overall Evaluation
1.0	Cost & Schedule: Cost and schedule data are used to control the performance of the software development project.

Exhibit 40: CTRL Evaluation

b. Policy (POL)

Policy (POL): The operational controls and strategic measures used to ensure that the software development project remained viable.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	66. How were operational controls and strategic measures balanced to ensure that the software development project remained viable? No coordination between the intelligence or policy functions existed.
Survey respondent 31954	66. How were operational controls and strategic measures balanced to ensure that the software development project remained viable? Response: There was a formal linkage between the operational controls in the project measurement and control process (PMC) and the policy functions in the integrated project management (IPM) process.
Survey respondent 31960	66. How were operational controls and strategic measures balanced to ensure that the software development project remained viable? Response: No coordination between the intelligence or policy functions existed.
Score	Overall Evaluation
0.0	No coordination: No coordination between the intelligence or policy functions existed..

Exhibit 41: POL Evaluation

c. Intelligence (INT)

Intelligence (INT): The external intelligence measures used as part of the planning process for the software development project.	
Source of Empirical Evidence	Empirical Evidence
Montealegre & Keil (2000)	By August 1994, Logplan issued an 11 page report that characterized BAE's system as "highly advanced" and "theoretically" capable of living up to its promised "capacities, services, and performances," but acknowledged that software and mechanical problems "make it most improbable to achieve a stable and reliable operation." Logplan recommended constructing a conventional tug-and-cart backup baggage system that could be built in less than five months and opening DIA with it and whatever parts of the BAE system could be ready (Booth and O'Driscoll 1994).
Survey respondent 31828	67. What external intelligence measures did the project consider as part of the planning process for the software development project? No intelligence measures were used.
Survey respondent 31954	67. What external intelligence measures did the project consider as part of the planning process for the software development project? Response: No intelligence measures were used
Survey respondent 31960	67. What external intelligence measures did the project consider as part of the planning process for the software development project? Response: No intelligence measures were used.
Score	Overall Evaluation
0.0	No intelligence measures were used.

Exhibit 42: INT Evaluation

d. Communications Channels (CC)

Communications Channels (CC): How information was communicated within the software development project.	
Source of Empirical Evidence	Empirical Evidence
Applegate, Montealegre, Knoop & Nelson (1996)	BAE had to change its working structure to conform to DIA's project management structure. Di Fonso explained, There was a senior manager for each of the concourses and a manager for the main terminal. The bag system, however, traversed all of them. If I had to argue a case for right of way I would have to go to all the managers because I was traversing all four empires. In addition, because changes were happening fast at each of these sites, there was no time to have an information system to see what is concourse A deciding and what is concourse B deciding. We had to be personally involved to understand what was going on. There was no one to tie it all together and overlap all these effects because the basic organization was to manage it as discrete areas. It was pandemonium. We would keep saying that over and over again. Who is in charge?
Donaldson (2002)	Communication was a problem from the beginning channels between: (a) The City, (b) The Project Management Team and (c) Consultants, were never well defined
Applegate, Montealegre, Knoop & Nelson (1996)	DIA's operational project structure comprised five different areas subdivided into smaller units. The working areas were: site development (earthmoving, grading, and drainage); roadways and on-grade parking (service roads, on-airport roads, and off-airport roads connecting to highways); airfield paving; building design (people-mover /baggage-handler, tunnel, concourses, passenger bridge, terminal, and parking); and utility/special systems and other facilities (electrical transmission, oil, and gas line removal and relocation). An area manager controlled construction within each area. Area managers were responsible for the administration of all assigned contracts and, in coordination with other area managers, for management of the portion of the overall site in which their work took place.
Applegate, Montealegre, Knoop & Nelson (1996)	Much of the effort for implementing the baggage system was directed within one of the four working areas. "The relationship with the management team was very poor," recalled Di Fonso. The management team had no prior baggage handling competence or experience. This was treated as a major public works project. The management team treated the baggage system as similar to pouring concrete or putting in air-conditioning ducts. When we would make our complaints about delays and access and so forth, other contractors would argue their position. The standard answer was, "Go work it out among yourselves." . . . With contractors basically on their own, this led almost to anarchy. Everyone was doing his or her own thing
Survey respondent 31828	68. How was information communicated within the software development project? Informal communications channels.
Survey respondent 31954	68. How was information communicated within the software development project? Response: Communications channels were formalized and included within the processes.
Survey respondent 31960	68. How was information communicated within the software development project? Response: Informal communications channels.
Score	Overall Evaluation
0.0	Informal communications channels.

Exhibit 43: CC Evaluation

e. Attenuation (ATT)

Environmental Attenuation (ATT): How the overall variety presented to the project was controlled by using attenuation and/or amplification methods.	
Source of Empirical Evidence	Empirical Evidence
Hickerson (2006)	A number of factors caused the delays, including electric power problems, software problems, and the fact that the client-server architecture, in a place so big, made problems in the system hard to track down and eliminate [13 Mahring et al, 2004, p. 219].
Donaldson (2002)	The City of Denver was unable to supply clean electricity to the baggage system: (1) The motors and circuitry was extremely sensitive to power surges and fluctuations. (2) Feedback continually tripped circuit breakers and filters to remedy this took months to arrive. (March 1994).
Rifkin (1994)	there were constant interruptions in testing due to overloaded motors, which in turn were due to faulty power supplies.
Applegate, Montealegre, Knoop & Nelson (1996)	Another problem was the city's inability to supply "clean" electricity to the baggage system. The motors and circuitry used in the system were extremely sensitive to power surges and fluctuations. When electrical feedback tripped circuit breakers on hundreds of motors, an engineer was called in to design filters to correct the problem. Although ordered at that time, the filters still had not arrived several months later. A city worker had canceled a contract without realizing that the filters were part of it. The filters finally arrived in March 1994.
Survey respondent 31828	69. How were the affects of unwanted or unnecessary information and/or materials from the external environment, upon the software development project, reduced? No attenuation methods were in place.
Survey respondent 31828	70. How were the necessary information and/or materials from the external environment or the software development project amplified? No amplification functions were in place.
Survey respondent 31954	69. How were the affects of unwanted or unnecessary information and/or materials from the external environment, upon the software development project, reduced? Response: Informal methods for attenuating unwanted information and/or materials to and from the environment were in place.
Survey respondent 31954	70. How were the necessary information and/or materials from the external environment or the software development project amplified? Response: Formal methods for amplifying important information and/or materials from the environment were in place.
Survey Respondent 31960	69. How were the affects of unwanted or unnecessary information and/or materials from the external environment, upon the software development project, reduced? Response: Informal methods for attenuating unwanted information and/or materials from the environment were in place.
Survey respondent 31960	70. How were the necessary information and/or materials from the external environment or the software development project amplified? Response: Informal methods for amplifying important information and/or materials from the environment were in place.
Score	Overall Evaluation
1.0	Informal methods for attenuation and amplification were in place.

Exhibit 44: ATT Evaluation

3. Technical System: The technical system area addressed nine measurement objects:

a. Software Reliability (RELY)

Software Reliability (RELY): The design of the software ensured that in the event of complete system failure the users would consider this to be:	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	71. The design of the software ensured that in the event of complete system failure the users would consider this to be a: Disastrous, high financial loss.
Survey respondent 31954	71. The design of the software ensured that in the event of complete system failure the users would consider this to be a: Response: Disastrous, high financial loss.
Survey respondent 31960	71. The design of the software ensured that in the event of complete system failure the users would consider this to be a: Response: Disastrous, high financial loss.
Score	Overall Evaluation
0.4	High: Disastrous, high financial loss.

Exhibit 45: RELY Evaluation

b. Database Size (DATA)

Database Size (DATA): The ratio of bytes in the system test database (D) to the number of bytes (source lines of code) in the application program (P).	
Source of Empirical Evidence	Empirical Evidence
De Neufville (1994)	Managing the information accurately is also difficult. The database needs to track tens of thousands of bags, going to hundreds of destinations, all in real time.
Survey respondent 31828	72. The ratio of bytes in the system test database (D) to the number of bytes (source lines of code) in the application program (P) was: $D/P < 10$
Survey respondent 31954	72. The ratio of bytes in the system test database (D) to the number of bytes (source lines of code) in the application program (P) was: Response: $D/P < 10$
Score	Overall Evaluation
1.0	Low: $D/P < 10$

Exhibit 46: DATA Evaluation

c. Development for Reuse (RUSE)

Development for Reuse (RUSE): The decision to utilize reusable software components as part of the software system design required the use of components.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	73. The decision to utilize reusable software components as part of the software system design required the use of components: Across multiple product lines.
Survey respondent 31954	73. The decision to utilize reusable software components as part of the software system design required the use of components: Response: Across multiple product lines.
Score	Overall Evaluation
0.2	Extra High: Across multiple product lines.

Exhibit 47: RUSE Evaluation

d. Life Cycle Documentation (DOCU)

Life Cycle Documentation (DOCU): How the design of the software system affected the life-cycle documentation needs of the software system.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	74. How did the design of the software system affect the life-cycle documentation needs of the software system? [Not Answered]
Survey respondent 31954	74. How did the design of the software system affect the life-cycle documentation needs of the software system? Response: Design has created excessive life-cycle needs.
Score	Overall Evaluation
0.4	Very High: Design has created excessive life-cycle needs.

Exhibit 48: DOCU Evaluation

e. Execution Time Constraint (TIME)

Execution Time Constraint (TIME): The percentage of the customer specified system response time was used in the design of the software system.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	75. What percentage of the customer specified system response time was used in the design of the software system? [Not Answered]
Survey respondent 31954	75. What percentage of the customer specified system response time was used in the design of the software system? Response: 70-90%
Score	Overall Evaluation
0.6	Very High: 70-90%

Exhibit 49: TIME Evaluation

f. Main Storage Constraint (STOR)

Main Storage Constraint (STOR): The percentage of the customer specified storage was used in the design of the software system.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	76. What percentage of the customer specified storage was used in the design of the software system? [Not Answered]
Survey respondent 31954	76. What percentage of the customer specified storage was used in the design of the software system? Response: 70-90%
Score	Overall Evaluation
0.6	Very High: 70-90%

Exhibit 50: STOR Evaluation

g. Platform Volatility (PVOL)

Platform Volatility (PVOL): How often changes to the software that made up the system are expected to be required.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	77. How often would changes to the software that made up the system be expected to be required? [Not Answered]
Survey respondent 31954	77. How often would changes to the software that made up the system be expected to be required? Response: Major changes every 6 months, minor changes every 2 weeks.
Score	Overall Evaluation
0.8	Low: Major changes every 6 months, minor changes every 2 weeks.

Exhibit 51: PVOL Evaluation

h. Software System Complexity (COMP)

System Complexity (COMP): The complexity of the software that the development project was working on.	
Source of Empirical Evidence	Empirical Evidence
Auguston (1994)	All complex handling systems must deal with the problem of resiliency and recovery. As you get closer and closer to peak capacity, the response time of the system increases.
Auguston (1994)	In short, what you're dealing with is a complicated cascade of queues, says de Neufville. "When the loads are heavy the performance of such systems are highly variable and service is often terrible, either overall or for particular connections. It's like driving through a busy city at rush hour."
Survey respondent 31828	78. Which description best characterizes the complexity of the software that the development project was working on? [Not Answered]
de Neufville (1994)	A more usual well-proven, well-maintained system of laser readers can boast of 96 to 97% accuracy, which still means that 3 to 4 bags out of 100 go to the misread pile or the wrong destination.
Hickerson (2006)	After some negotiation, BAE proposed building what one article called, "the most complex baggage-handling system ever built." According to one account, it was to include 3,100 independent "telecars" on 22 miles of tracks and six miles of conveyor belt to route and deliver baggage among 20 different airlines. BAE built a prototype system in a warehouse near its manufacturing plant near Carrollton, Texas, and the prototype systems was enough to convince top management to move ahead with the plan.
Rifkin (1994)	All of the airport's resident airlines, except United, use individual bar codes printed on stickers right at the ticket counter and attached to the handles of the bags. The bar codes identify the passenger and the flight information. United uses a tiny photocell that performs the same function.
Montealegre & Keil (2000)	Although BAE had significant experience implementing this technology in smaller scale projects, it had never implemented a system at the level of complexity that was required for DIA.
de Neufville (1994)	An efficient control system for any automated baggage system is likely to take a long time to develop successfully.
Donaldson (2002)	BAE came up with a proposal (most ambitious ever - biggest, most complex automated system ever)
Donaldson (2002)	BAE had installed Telecar (laser barcode readers and conveyor belt system) but never on the size envisaged in the tender offer DIA was going to need something much bigger.

Applegate, Montealegre, Knoop & Nelson (1996)	BAE presented the City of Denver with a proposal to develop the "most complex automated baggage system ever built," according to Di Fonso. It was to be effective in delivering bags to and from passengers, and efficient in terms of operating reliability, maintainability, and future flexibility. The system was to be capable of directing bags (including suitcases of all sizes, skis, and golf clubs) from the main terminal through a tunnel into a remote concourse and directly to a gate. Such efficient delivery would save precious ground time, reduce close-out time for hub operations, and cut time-consuming manual baggage sorting and handling.
Montealegre & Keil (2000)	BAE responded with a proposal to develop the "most complex baggage handling system ever built." The proposed system was to include 3,100 independent "telecars" to route and deliver luggage among the counters, gates, and claim areas of 20 different airlines. A total of 55 personal computers would be networked to one another and to 5,000 optical detectors, 400 radio receivers, and 56 bar-code scanners in a central control system. The system would move a passenger's bag from any injection point to any destination in the airport in less than 15 minutes and would process more than 1,000 bags per minute--two to three times faster than a conventional conveyor belt. Faster baggage handling would translate into increased ground time efficiency, thus reducing enplanement turnaround time for hub operations and improving services to passengers (Bouton 1993).
Applegate, Montealegre, Knoop & Nelson (1996)	BAE was among the companies that had decided not to bid for the job. BAE had installed the Telecar system at a number of other airports and the basic technologies of the Telecar, laser barcode readers, and conveyor belt systems were not new. What was new was the size and complexity of the system. "A grand airport like DIA needs a complex baggage system," explained Di Fonso, "Therefore the type of technology to be used for such a system is the kind of decision that must be made very early in a project. If there is a surprise like no bidders there is still time to react. At DIA, this never happened. Working with United Airlines, we had concluded that destination-coded vehicles moving at high speed was the technology needed. But quite honestly, although we had that technology developed, its implementation in a complex project like this would have required significantly greater time than the city had left available."
Auguston (1994)	Complex systems like the baggage handling system at the Denver International Airport (DIA) are often difficult to manage because they involve technology that is either unproven or has never been attempted on such a scale before.
Henderson (1994)	Designed to move 42,000 pieces of luggage an hour at speeds of up to 19 mph in 4,000 Destination Coded Vehicles controlled and monitored by computer and propelled by 2,100 linear induction motors, "the most complex baggage system in North America, if not the world..."
GAO (1994)	Even after modifications are complete, the automated baggage handling system will have two main components: (1) high-speed, bag-carrying telecars mounted on tracks and (2) connecting conveyor belts to load and off-load baggage. The tracks are suspended from the basement ceilings of the terminal and concourses. Electric motors and synchronous drives move the telecars along the tracks at varying speeds. Photocells and radio frequency reading devices direct each telecart to the right location. In total, the original system included over 17 miles of track; 5.5 miles of conveyors; 4,000 telecars; 5,000 electric motors; 2,700 photocells; 59 laser bar code reader arrays; 311 radio frequency readers; and over 150 computers, workstations, and communication servers. The automated system was originally designed to carry up to 70 bags per minute to and from the baggage check-in and baggage claim areas at speeds of up to 24 miles per hour. This would allow the airlines to receive checked baggage at their aircraft within 20 minutes.
Gibbs (1994)	Even more impressive than its girth is the airport's subterranean baggage-handling system. Tearing like intelligent coal-mine cars along 21 miles of steel

	track, 4,000 independent "telecars" route and deliver luggage between the counters, gates and claim areas of 20 different airlines. A central nervous system of some 100 computers networked to one another and to 5,000 electric eyes, 400 radio receivers and 56 bar-code scanners orchestrates the safe and timely arrival of every valise and ski bag.
Goetz & Szyliowicz (1997)	First, the mechanical aspects, such as the construction of the tracks and the operation of the carts, demand precision. If angles and tolerances are not calculated correctly, the carts would either not be read properly or would crash or fly off the tracks.
de Neufville (1994)	If relative complexity is measured by a factor of the increases in the salient dimensions of a system, the fully automated system originally designed for Denver would be 100 times as complex as comparable systems elsewhere. This crude estimate factors speed (10x) and the number of destinations (~10x). Massive problems and therefore extensive delays should have been expected from the start. The enormous increase in complexity, that distinguishes the fully automated baggage system attempted at Denver from all others, represents much more than a simple evolution of technology. It is not just a change from a third to a fourth generation of technology, say; it is more like an attempted leap from the third to the fifth or sixth generation of baggage systems.
de Neufville (1994)	Managing a complex network of interacting, fully loaded queues efficiently for any single scenario is complicated. Managing these flows under all the realistic scenarios is exponentially more difficult. Learning how to do this appears to be a major, long-term research project. Both airports, such as Frankfurt am Main, and companies attempting to automate their materials handling, have routinely spent years trying to make their systems work correctly under all circumstances (Auguston, 1994; Zitterstein, 1994). It is not clear that anyone, anywhere, is currently capable of managing a fully automated baggage system -- one without any backup system or use of tugs and carts for transfers -- to ensure full capacity, on-time performance, or is likely to be able to do so anytime in the near future (Knill, 1994).
GAO (1995)	Provide for alternative or back-up systems when dealing with new and untested technology. The automated baggage system, which will cost about \$234 million, was to be one of the largest and most sophisticated systems of its kind in the world.
GAO (1994)	The automated baggage handling system, with a contract price of \$193 million, will be one of the largest and most sophisticated systems of its type in the world.
Szyliowicz & Goetz (1995)	The baggage system, whose cost had by now escalated to about \$200 million, was to be the largest and most sophisticated in the world. It was designed to move 700 bags a minute to their specific load points in less than ten minutes through a system of 4,000 individual carts traveling at speeds up to 24 miles an hour over 17 miles of track suspended from basement ceilings. The system, which included 5,000 electric motors, 2,700 photocells, 59 laser bar code reader stations, 311 radio frequency readers, and more than 150 computers, workstations, and communication servers, was expected to have all of its carts operating simultaneously during peak hours. ³
de Neufville (1994)	The delivery mechanism consists of about 9 km. (5.5 miles) of conveyors and over 27 km. (17 miles) of track on which circulate 4000 individual, radio-controlled carts, the so-called "destination coded vehicles" or "DCVs"
de Neufville (1994)	The destination of each bag and its individual cart is defined by bar-coded labels, and transmitted by radio to tags (the "radio frequency identification" or "RFID") on the constantly moving vehicles. The operation of these vehicles is to be entirely controlled by a network of about 150 computers (Myerson, 1994; US Government Accounting Office, 1994).

Rifkin (1994)	the DIA system is the most ambitious effort yet. Whereas other baggage systems are located in the concourse of a specific airline, the DIA system is the first integrated system to serve an entire airport. It is also the first to include a transfer system for the rerouting of bags in case of sudden gate changes.
de Neufville (1994)	the geometry was tight. The automated system had to fit within the confines of the airport passenger buildings and the underground tunnel connecting the concourses and the terminal; in many instances it was shoe-horned in at considerable inconvenience.
Kerzner (2000)	The system would contain 100 computers, 56 laser scanners, conveyor belts, and thousands of motors. As designed, the system would contain 400 fiberglass carts, each carrying a single suitcase through 22 miles of steel tracks. Operating at 20 miles per hour, the system could deliver 60,000 bags per hour from dozens of gates. United was worried that passengers would have to wait for luggage since several of their gates were more than a mile from the main terminal. The system design was for the luggage to go from the plane to the carousel in 8-10 minutes. The luggage would reach the carousel before the passengers. The baggage handling system would be centered on track-mounted cars propelled by linear induction motors. The cars slow down, but don't stop, as a conveyor ejects bags onto their platform. During the induction process, a scanner reads the bar-coded label and transmits the data through a programmable logic controller to a radio frequency identification tag on a passing car. At this point, the car knows the destination of the bag it is carrying, as does the computer software that routes the car to its destination. To illustrate the complexity of the situation, consider 4,000 taxicabs in a major city, all without drivers, being controlled by a computer through the streets of a city.
Applegate, Montealegre, Knoop & Nelson (1996)	To prove the capability of its mechanical aspects, and demonstrate the proposed system to the airlines and politicians, BAE built a prototype automated baggage handling system in a 50,000 square foot warehouse near its manufacturing plant in Carrollton, Texas.
Donaldson (2002)	To prove the capability, BAE proposed to build a prototype automated baggage handling system in a 50,000 sq. ft. warehouse near its manufacturing plant in Texas.
Russell (1994)	Having agreed to design and build a system that could serve up to three hubbing airlines, BAE, of Dallas, was defeated by computer-programming complexities.
Survey respondent 31954	78. Which description best characterizes the complexity of the software that the development project was working on? Response: System: collection of subsystems with multiple functions.
Score	Overall Evaluation
0.6	System: Collection of subsystems with multiple functions.

Exhibit 52: COMP Evaluation

i. Technology Application (TECH)

Technology Application (TECH): The software and/or hardware technology in place at the start of the development project.	
Source of Empirical Evidence	Empirical Evidence
Montealegre & Keil (2000)	No one [in the DIA management team] realized the complexity of the technology as it relates to this baggage system (O'Driscoll 1994a). A project manager for United Airlines recalled: "BAE told them from the beginning that they were going to need at least one more year to get the system up and running, but no one wanted to hear that." The City of Denver was getting the same story from the technical advisers to the Franz Josef Strauss Airport in Munich, but apparently chose not to listen (Rifkin 1994).
Kerzner (2000)	DIA's \$200 million baggage handling system was designed to be state of the art
Auguston (1994)	Dr. de Neufville, who has researched and taught courses in airport systems planning for the past 25 years, believes that designers of the baggage handling system at the Denver International Airport (DIA) should have expected massive problems and extensive delays from the start. "This system represents an enormous technological leap. It is not simply a step up from a third to a fourth generation system, but more like an attempted leap to the fifth or sixth generation," says de Neufville.
De Neufville (1994)	"The development of a fully integrated, automated baggage system, such as the one originally designed for Denver, represents an enormous technological leap over current practice. No airline, for example, has used a fully automated system to deliver ""hot"" or time sensitive baggage for passengers transferring between aircraft in 45 minutes or less. The individual elements of the baggage system at the New Denver Airport have each, separately and on a much smaller scale, been used successfully -- but they have not functioned together in such a large system. This enormous increase in complexity is the root of the problem. It is a truism in systems design, that as you increase the complexity, the difficulties in making the system work increase ""exponentially"". If the system is 10 times as complex, the difficulties could be 100 times as great. The fully automated system at the New Denver Airport is far more complex than predecessor systems. It features about 12 times as many carts as in the existing comparable systems in San Francisco or Atlanta, which are also very much simpler in layout and the number of connections. The speed of its carts is about 10 times as great as on conventional conveyor belts. ""The Denver system represents a leap in scale, with 14 times the capacity of San Francisco's. It is the first such system to serve an entire airport. It is also the first where the carts will only slow down, not stop, to pick up and drop off bags, the first to be run by a network of desktop computers rather than a mainframe, the first to use radio links and the first with a system for oversized bags, which in Denver tend to be skis."" (Myerson, 1994)
Hickerson (2006)	the initial plans always meant to push the envelope as far as technology and operating an airport were concerned. For the systems that ran the airport, distributed client-server architectures would be put in place to run all sorts of specialized functions
Applegate, Montealegre, Knoop & Nelson (1996)	There were, however, a number of risks inherent in the endeavor: the scale of the large project size; the enormous complexity of the expanded system; the newness of the technology; the large number of resident entities to be served by the same system; the high degree of technical and project definition uncertainty; and the short time span for completion. Due to its significant experience implementing baggage-handling technology on a smaller scale, BAE Automated Systems Inc., an engineering consulting and manufacturing

	company based in Carrollton, Texas, was awarded the contract.
Goetz & Szyliowicz (1997)	This technology was known to be a difficult one to implement. The Frankfurt airport planners had adopted a smaller version of this system and it took them 15 years to build and test it before it worked properly.
Szyliowicz & Goetz (1995)	Under optimal conditions, achieving this task would have been no easy matter. The baggage system was a highly complex technology that was designed to deliver bags automatically from check-in to the gate. Luggage with bar-coded tags would be placed into individual tele-carts and electronically scanned to identify their appropriate destinations and be routed to the correct gate. The system's scale and complexity is illustrated by the need for hundreds of individual tele-carts, 17 miles of track, over 150 computers and servers, dozens of bar-code and radio frequency readers, and thousands of switching software programs, electric motors and photocells (General Accounting Office, 1994; Rifkin, 1994). Furthermore, the entire system would be operated within a tunnel network which had to be greatly modified to meet the new requirements.
Goetz & Szyliowicz (1997)	Unfortunately this system, which involved state-of-the-art technology, proved to be a disaster. It encountered numerous mechanical and software problems that, when tests were run on a small loop of the system, resulted in misloaded bags, jammed carts, spilled luggage and, general chaos. Following extensive discussions involving United, the City, BAE (the system designer), and external consultants, it was decided in September 1994 to drastically simplify the system to serve, at first, only United's concourse and, simultaneously, to build an alternative system of traditional tugs and carts to serve the other concourses at a cost of more than \$ 50 million to enable the airport to open on February 28, 1995. This decision led to major protests by the other airlines, especially those located at Concourse C, the farthest from the terminal, who feared that United would gain a significant competitive advantage since other airlines' baggage would be delivered much more slowly.
Survey respondent 31828	79. Which description best characterizes the software and/or hardware technology in place at the start of the development project? [Not Answered]
Survey respondent 31954	79. Which description best characterizes the software and/or hardware technology in place at the start of the development project? Response: Super-High tech: necessary technologies did not exist at project initiation.
Score	Overall Evaluation
0.2	Super High-Tech: Necessary technologies did not exist at project initiation.

Exhibit 53: TECH Evaluation

The overall score for the structure element was 10.8 of 25 possible points.

Environment Element

The environment element of the FSE framework had three areas that evaluated 14 measurement objects related to the environmental factors affecting the ability of the project to produce software. The areas address the external controls, the resources, and the stakeholders worth a total of 25 points. Each area will be evaluated separately.

1. External Controls: The external controls area addressed two measurement objects:

a. Laws and Regulations (LAW)

Laws and Regulations (LAW): The extent to which government laws and regulations were addressed by the project and/or parent organization.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	80. How would you characterize the extent to which government laws and regulations were addressed by the project and/or parent organization? The project and/or the parent organization were not involved in influencing laws and regulations.
Survey respondent 31954	80. How would you characterize the extent to which government laws and regulations were addressed by the project and/or parent organization? Response: The project and/or the parent organization were not involved in influencing laws and regulation
Survey respondent 31960	80. How would you characterize the extent to which government laws and regulations were addressed by the project and/or parent organization? Response: The project and/or the parent organization were not involved in influencing laws and regulations.
Score	Overall Evaluation
0.2	The project and/or the parent organization are not involved in influencing laws and regulations.

Exhibit 54: LAW Evaluation

b. Industry Standards (STD)

Industry Standards (STD): The extent to which industry standards were addressed by the project and/or parent organization.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	81. How would you characterize the extent to which industry standards were addressed by the project and/or parent organization? {Choose one} () The project and/or the parent organization were not involved in influencing industry standards.
Survey respondent 31954	81. How would you characterize the extent to which industry standards were addressed by the project and/or parent organization? Response: The project and/or the parent organization were not involved in influencing industry standards.
Survey respondent 31960	81. How would you characterize the extent to which industry standards were addressed by the project and/or parent organization? Response: The project and/or the parent organization had limited involvement in influencing industry standards.
Score	Overall Evaluation
0.2	The project and/or the parent organization are not involved in influencing industry standards.

Exhibit 55: STD Evaluation

2. Resources: The resources area addressed six measurement objects:

a. Manpower: Labor Availability (MAN)

Manpower Labor Availability (MAN): The availability of labor skills required by the project.	
Source of Empirical Evidence	Empirical Evidence
Applegate, Montealegre, Knoop & Nelson (1996)	A third, albeit disputed, complication related to Denver's requirement, and city law, that a certain percentage of jobs be contracted to minority-owned companies. The City of Denver had denied BAE's original contract because it did not comply with hiring requirements, where upon BAE engaged some outside contractors in lieu of BAE employees.
Donaldson (2002)	Amongst the legal requirements for funding were the requirements that 30% minority-owned firms were used, and 6% of women-owned firms were used.
Donaldson (2002)	Denver's city laws required that a certain percentage of jobs be contracted to minority owned companies. - BAE's original contract had been denied because they did not meet this requirement and BAE engaged outside contractors to address this requirement. BAE estimate that this increased costs by \$6 million (a claim rejected by the Mayor's Office of Contract Compliance).
Survey respondent 31828	82. How would you characterize the availability of labor skills required by the project? All required labor skills were present in the local workforce permitting the project to do all development locally.
Survey respondent 31954	82. How would you characterize the availability of labor skills required by the project? Response: Some of the required labor skills were present in the local workforce. Key labor skills were required to be imported. The project was a hybrid of local and remote development.
Survey respondent 31960	82. How would you characterize the availability of labor skills required by the project? Response: Most of the required labor skills were present in the local workforce. Only a few key labor skills were required to be imported. The project was developed locally and used some limited remote development.
Score	Overall Evaluation
1.0	Nominal: Some of the required labor skills were present in the local workforce. Key labor skills were required to be imported. The project was a hybrid of local and remote development.

Exhibit 56: MAN Evaluation

b. Critical Material Availability (MAT)

Critical Material Availability (MAT): The availability of critical materials (typically hardware and software components) required by the project.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	83. How would you characterize the availability of critical materials (typically hardware and software components) required by the project? Critical materials were readily available.
Survey respondent 31954	83. How would you characterize the availability of critical materials (typically hardware and software components) required by the project? Response: Critical materials were in development but had not been released at project initiation.
Survey respondent 31960	83. How would you characterize the availability of critical materials (typically hardware and software components) required by the project? Response: Critical materials were dependent upon an emerging technology and did not

	exist at project initiation.
Score	Overall Evaluation
0.5	Developmental: Critical materials were in development but had not been released at project initiation.

Exhibit 57: MAT Evaluation

c. Money: Capital Investment (CAP)

Money: Capital Investment (CAP): The availability of capital required to fund indirect activities (i.e. process improvement, skill development, training, etc.) on the project.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	84. How would you characterize the availability of capital required to fund indirect activities (i.e. process improvement, skill development, training, etc.) on the project? {Choose one} () Capital funds were readily available.
Survey respondent 31954	84. How would you characterize the availability of capital required to fund indirect activities (i.e. process improvement, skill development, training, etc.) on the project? Response: Capital funds for the project were included in the project budget.
Survey respondent 31960	84. How would you characterize the availability of capital required to fund indirect activities (i.e. process improvement, skill development, training, etc.) on the project? Response: Capital funds for the project were included in the project budget.
Score	Overall Evaluation
0.5	Budgeted: Capital funds were included in the budget.

Exhibit 58: CAP Evaluation

d. Schedule Pace (PACE)

Schedule Pace (PACE): The pace required to achieve the schedule for the project.	
Source of Empirical Evidence	Empirical Evidence
Applegate, Montealegre, Knoop & Nelson (1996)	A United project manager concurred: "BAE told them from the beginning that they were going to need at least one more year to get the system up and running, but no one wanted to hear that." The City of Denver was getting the same story from the technical advisers to the Franz Josef Strauss Airport in Munich. The Munich Airport had an automated baggage system, but one far less complex than DIA's. Nevertheless, Munich's technical advisors had spent two years testing the system and the system had been running 24 hours a day for six months before the airport opened.
Kerzner (2000)	BAE Automated Systems personnel began to complain that they were pressured into doing the impossible. The only other system of this type in the world was in Frankfurt, Germany. That system required six years to install and two years to debug. BAE was asked to do it all in two years.
Donaldson (2002)	BAE told UA that it would take at least a year to get the system up and running, but no one wanted to hear that. City of Denver got the same story from technical advisers to the Franz Joseph Strauss airport in Munich (that less complicated system had taken 2 years testing and was running 24 hours a day for 6 months before the airport opened.
Kerzner (2000)	BAE was selected to design and build the baggage handling system. The airport had been under construction for three years before BAE was brought on board. BAE agreed to do eight years of work in two years to meet the October, 1993 opening date.

Applegate, Montealegre, Knoop & Nelson (1996)	Construction problems kept the new airport from opening on the originally scheduled opening date in October 1993. Subsequently, problems with the implementation of the baggage system forced delays in the opening of the airport another three times in seven months.
Rifkin (1994)	Of 16 companies contacted, both in the U.S. and abroad, Di Fonso claims that not a single one was willing to make a bid. All had the same response: there was not enough time to build such a system.
Donaldson (2002)	Owing to the tight deadlines, BAE would have priority in any area where it needed to install the system.
GAO (1995)	The City had already explored the feasibility of installing an airport wide automated baggage system. In August 1990, a study commissioned by the City indicated that the highly complex and technically difficult state-of-the-art automated baggage system necessary for an airport of that size could probably not be built and tested in time to meet the scheduled opening date of October 1993. Specifically, the consultant's report discussed the risks involved with five baggage system options. ³
Goetz & Szyliowicz (1997)	The DIA system was expected to operate flawlessly at increased levels of scale and complexity and to do so without allowing more time in the construction schedule for difficulties that could have been readily anticipated.
de Neufville (1994)	the schedule was tight. The system was to be implemented within 21 months, since Denver executed the contract only in January 1992. This schedule precluded extensive simulation or physical testing of the full design.
Survey respondent 31828	85. How would you characterize the pace required to achieve the schedule for the project? Completion time was crucial for success-window of opportunity.
Survey respondent 31954	85. How would you characterize the pace required to achieve the schedule for the project? Response: Completion time was crucial for success-window of opportunity.
Survey respondent 31960	85. How would you characterize the pace required to achieve the schedule for the project? Response: Completion time was crucial for success-window of opportunity.
Score	Overall Evaluation
0.5	Critical: Completion time is crucial for success-window of opportunity.

Exhibit 59: PACE Evaluation

e. Formal Methods (METH)

Formal Methods (METH): The adoption and implementation of formal methods on the project.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	86. How would you characterize the adoption and implementation of formal methods on the project? No formal standards were used.
Survey respondent 31954	86. How would you characterize the adoption and implementation of formal methods on the project? Response: Formal standards were used in most process areas.
Survey respondent 31960	86. How would you characterize the adoption and implementation of formal methods on the project? Response: Formal standards were used in a few process areas.
Score	Overall Evaluation
1.0	Nominal: Formal standards were used in a few process areas.

Exhibit 60: METH Evaluation

e. Information: External Communications (COMM)

Information: External Communications (COMM): The methods used to control the flow of information between the project and the external environment.	
Source of Empirical Evidence	Empirical Evidence
Rifkin (1994)	Communications were shaky at best. Engineers out in the concourses simply couldn't talk to their colleagues in the terminal because of dead spots in radio transmission around the airport.
Applegate, Montealegre, Knoop & Nelson (1996)	Moreover, communication channels between the city, project management team, and consultants were neither well-defined nor controlled.
Survey respondent 31828	87. How would you characterize the methods used to control the flow of information between the project and the external environment? No formal information controls existed.
Survey respondent 31954	87. How would you characterize the methods used to control the flow of information between the project and the external environment? Response: Information was closely monitored and dissemination controls were strictly enforced.
Survey respondent 31960	87. How would you characterize the methods used to control the flow of information between the project and the external environment? Response: Information controls were limited to formal correspondence.
Score	Overall Evaluation
1.0	Limited: Information controls are limited to formal correspondence.

Exhibit 61: COMM Evaluation3. Stakeholders: The stakeholder area addressed six measurement objects:

a. Owners/Shareholder Boards (OWN)

Owners/Shareholders (OWN): The formal level of involvement of the owner(s) of the company or the corporate board of directors with the development project.	
Source of Empirical Evidence	Empirical Evidence
Montealegre & Keil (2000)	At DIA, external constituencies also had a powerful influence on the de-escalation process. United Airlines played an important role in attempting to prevent the City of Denver from abandoning the automated baggage handling system.
Keil & Montealegre (2000)	Eventually, however, he withdrew his commitment to the system. Dealing with the costs of further delays meant redefining the problem and finding an expedient way to open the airport as soon as possible.
Szyliowicz & Goetz (1995)	Finally, we must note important distinctions between the actual planning process and the Rational model. Despite the appearance of 'rationality' which the stages project, the entire process was subject to a range of influences that are not accounted for by the type of activities called for by the Rational model. An obvious example is Governor Romer's attempt to influence voters in Adams County to support the new Denver airport, an effort which proved critical to a favorable election outcome and the late baggage system decision with its disastrous consequences.
Montealegre & Keil (2000)	In the DIA case, the consultant's report allowed Mayor Webb to place the blame for the baggage handling problems squarely on the contractor, BAE, even though it was the city that had initiated and pursued the project so vigorously. Blaming the failure on BAE's own faulty management and lack of technical expertise was a recurring theme in public statements by city officials

	and members of the project management team. By using this and other impression management techniques, Mayor Webb was able to save face in dealing with the media and other stakeholders.
Montealegre & Keil (2000)	In the DIA case, the costs of delaying the airport opening prompted a redefinition of the problem and a search for alternative courses of action. In the process, an outside consultant (Logplan) was engaged to identify and legitimize an alternative course of action.
Montealegre & Keil (2000)	In the DIA case, this involved appealing to stakeholders to reach a mutually agreeable implementation strategy and de-institutionalizing the project.
Hickerson (2006)	The costs incurred towards stakeholders involved to keep the airport running but not open. After the 1994 demonstration that showed to reporters that DIA's system was nowhere near ready, the City of Denver had to renegotiate with all the airlines to help carry the cost of not opening the airport. Tenant airlines agreed to carry the costs, but those turned into problems for the consumers later on; airlines levied a \$40 surcharge on round-trip tickets to Denver when the airport finally was operational, and generated ill will towards travelers to the area.
Goetz & Szyliowicz (1997)	The first sign of trouble was the reluctance of the airlines to sign leases at the new facility. Neither of the major hub carriers (Continental and United) were enthusiastic about the new airport because of the potentially high operating costs involved.
Montealegre & Keil (2000)	United Airlines objected to the manual system, saying it would not accommodate the airline's heavy schedule. A United Airlines official told a Denver Post reporter that, "Webb's choice would gridlock the DIA baggage movement disastrously." United feared that a traditional system would hurt its huge Denver hub--with 284 flights a day--by slowing luggage transfers and lengthening the time needed to send bags from ticket counters to airline gates. As United's senior vice president for customer service put it, "the alternative-system plan will take us back 30 years" (Mark 1994).
de Neufville (1994)	United Airlines, the dominant airline at Denver, 1 insisted on a rapid baggage handling system before signing its lease with Denver (Flynn, 1994b).
Russell (1994)	The city and Greiner/MKE acted as a Project Management Team (PMT), coordinating schedule, cost control, information management, and administration of some 100 design contracts and, ultimately, some 160 general contractors and more than 2,000 subcontractors (chart opposite). This entity was the "owner" to whom the architects reported.
Survey respondent 31828	88. How would you characterize the formal level of involvement of the owner(s) of the company or the corporate board of directors with the development project? Owners or shareholder boards had limited knowledge about the project (i.e. revenue contribution and profit/loss).
Survey respondent 31954	88. How would you characterize the formal level of involvement of the owner(s) of the company or the corporate board of directors with the development project? Response: Owners or shareholder boards were informed about overall project performance (i.e. cost, schedule and customer satisfaction).
Survey respondent 31960	88. How would you characterize the formal level of involvement of the owner(s) of the company or the corporate board of directors with the development project? Response: Owners or shareholder boards were informed about overall project performance (i.e. cost, schedule and customer satisfaction).
Score	Overall Evaluation
1.0	Informational: Owners or shareholder boards were informed about overall project performance (i.e. cost, schedule and customer satisfaction).

Exhibit 62: OWN Evaluation

b. External Management (MGT)

External Management (MGT): The formal level of involvement of external management with the development project.	
Source of Empirical Evidence	Empirical Evidence
Donaldson (2002)	For the first 2 years of the project the BAE chairman was the project manager.
Keil & Montealegre (2000)	In the case of DIA, managerial action during this phase only became possible when the executives involved were able to both clarify the magnitude of the problem and redefine it.
Survey respondent 31828	89. How would you characterize the formal level of involvement of external management with the development project? External management had no involvement with the project.
Survey respondent 31954	89. How would you characterize the formal level of involvement of external management with the development project? Response: External management was informed about overall project performance (i.e. cost, schedule and customer satisfaction).
Survey Respondent 31960	89. How would you characterize the formal level of involvement of external management with the development project? Response: External management was informed about overall project performance (i.e. cost, schedule and customer satisfaction).
Score	Overall Evaluation
1.0	Informational: External management was informed about overall project performance (i.e. cost, schedule and customer satisfaction).

Exhibit 63: MGT Evaluation

c. Customers (CUST)

Customers (CUST): The formal level of involvement of the customer (those that contract and pay for the project and as such are differentiated from users) with the progress of the development project.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	90. How would you characterize the formal level of involvement of the customer (those that contract and pay for the project and as such are differentiated from users) with the progress of the development project? Periodic cost and schedule reports were sent to the customer.
Survey respondent 31954	90. How would you characterize the formal level of involvement of the customer (those that contract and pay for the project and as such are differentiated from users) with the progress of the development project? Response: The customer required periodic face-to-face reviews of project progress and costs.
Survey respondent 31960	90. How would you characterize the formal level of involvement of the customer (those that contract and pay for the project and as such are differentiated from users) with the progress of the development project? Response: Periodic reviews and evaluations with the customer were formalized and included within the project's processes.
Score	Overall Evaluation
1.5	Managed: The customer required periodic face-to-face reviews of project progress and costs.

Exhibit 64: CUST Evaluation

d. Suppliers (SUP)

Suppliers (SUP): The involvement of those who supplied products and/or services to your development project.	
Source of Empirical Evidence	Empirical Evidence
Survey respondent 31828	91. How would you characterize the involvement of those who supplied products and/or services to your development project? The supplier's had no knowledge as to their involvement with the project (i.e. only fee and delivery date).
Survey respondent 31954	91. How would you characterize the involvement of those who supplied products and/or services to your development project? Response: The supplier's involvement was constrained to his product or service area on the project (i.e. limited project cost and schedule visibility).
Survey respondent 31960	91. How would you characterize the involvement of those who supplied products and/or services to your development project? Response: The supplier's involvement was constrained to his product or service area on the project (i.e. limited project cost and schedule visibility).
Score	Overall Evaluation
0.5	Limited: The supplier's involvement was constrained to his product or service area on the project (i.e. limited project cost and schedule visibility).

Exhibit 65: SUP Evaluation

e. Users (USER)

Users (USER): The involvement of the end user's of the software being developed by your development project.	
Source of Empirical Evidence	Empirical Evidence
Keil & Montealegre (2000)	Faster baggage handling would translate into increased ground-time efficiency, reducing turnaround time for hub operations and improving services to passengers.
Applegate, Montealegre, Knoop & Nelson (1996)	From the public's perspective, the "friendliness" of any airport is measured by time. No matter how architecturally stimulating a new airport structure, the perception of business or leisure travelers is often registered in terms of efficiency in checking luggage at the departure area or waiting to claim a bag in the arrival area. The larger the airport, the more critical the efficient handling of baggage. Remote concourses connected by underground tunnels present special problems for airport planners and operators because of the great distances passengers and baggage must travel. The purpose of an airport being to move passengers as efficiently as possible, moving bags as quickly is part and parcel of that responsibility. Rapid transport of frequent flyers accomplishes very little if bags are left behind.
GAO (1995)	In 1989, the City began to solicit bids for construction without obtaining formal input on the airport's design from the ultimate users of the facility--the airlines. In negotiating with these major tenants to sign gate leases, the City agreed to some very large and significant design changes. These decisions triggered far-reaching changes to the design and construction of DIA's buildings and systems, many of them in mechanical, electrical, and telecommunications systems that are complex and difficult to coordinate.
Survey respondent 31828	92. How would you characterize the involvement of the end user's of the software being developed by your development project? Users were not involved.
Survey respondent 31954	92. How would you characterize the involvement of the end user's of the

	software being developed by your development project? Response: Users were involved in the development of requirements, validation of design, and user acceptance testing.
Survey respondent 31960	92. How would you characterize the involvement of the end user's of the software being developed by your development project? Response: Users were involved in the development of requirements, validation of design, and user acceptance testing.
Score	Overall Evaluation
1.5	Involved: Users were involved in the development of requirements, validation of design, and user acceptance testing.

Exhibit 66: USER Evaluation

f. Politics (POLT)

Political Involvement (POLT): The extent to which Politics played a role on the software development project.	
Source of Empirical Evidence	Empirical Evidence
Goetz & Szyliowicz (1997)	After the fourth delay in May 1993, the city decided to focus on making the automated system operational only for United's concourse and to build, in addition, a traditional conveyor belt/tug-and-cart system to serve the rest of the airport. Altogether, the troubled automated baggage system's cost escalated to \$360 million, the traditional system added another \$50 million
Goetz & Szyliowicz (1997)	Although the key actors responsible for the design and implementation of the project (the city and the FAA) may have tried to be rational, their attempt to do so was inevitably frustrated by the presence of so many other actors, each of which had its own interests, strategies and power bases.
Goetz & Szyliowicz (1997)	At DIA, despite two elections, many observers believe the pro-airport coalition's resources (including strong media support) did not permit genuine debate.
Montealegre & Keil (2000)	At the same time, Mayor Webb notified BAE of a \$12,000-a-day penalty for not finishing the baggage system by DIA's original October 29, 1993, completion date. Webb also demanded that BAE pay for the \$50 million conventional tug-and-cart baggage system. Di Fonso, reviewing Mayor Webb's letter, summed up the situation as follows: We have gotten to the point with the city that we are literally not talking to each other. Consultants recommended a backup baggage system, and the minute that the decision was made, the city had to defend it. We are left out in limbo.
Applegate, Montealegre, Knoop & Nelson (1996)	In August 1994, Mayor Webb approved the construction of a backup baggage system. At the same time, he notified BAE of a \$12,000-a-day penalty for not finishing the baggage system by DIA's original October 29, 1993 completion date. Webb also demanded that BAE pay for the \$50 million conventional tug-and-cart baggage system.
Applegate, Montealegre, Knoop & Nelson (1996)	In May 1994, under growing pressure from shareholders, the business community, Denver residents, Federal Aviation Administration (FAA) commissioners, and the tenant airlines and concessionaires, Denver mayor Wellington Webb announced that he was hiring the German firm Logplan to help assess the state of the automated baggage system. In July, Logplan issued an 11-page report to the City of Denver that characterized BAE's system as "highly advanced" and "theoretically" capable of living up to its promised "capacities, services and performances," but acknowledged mechanical and electrical problems that "make it most improbable to achieve a stable and reliable operation." Logplan suggested that it would take approximately five months to get the complete BAE system working reliably. It also suggested

	that a backup system of tugs, carts, and conveyor belts could be constructed in less than five months.
Goetz & Szyliowicz (1997)	In terms of costs and delays, however, the city's bargaining strategy proved to be a disaster. Each airline extracted major financial and design concessions that were to have a devastating effect upon the project's budget and schedule.
Montealegre & Keil (2000)	Just one week after Mayor Webb announced the plan to develop the alternative baggage system, the Denver City Council approved the hiring of the Michigan-based Rapistan Demag firm to design, engineer, and install a conventional baggage handling system.
Kerzner (2000)	Most important, United wanted a destination-coded vehicle (DCV) baggage handling system where bags could be transferred between gates in less than 10 minutes, thus supporting short turnaround times. The DCV was to be on Concourse B (United) only. Within a few weeks thereafter, DIA proposed that the baggage handling system be extended to the entire airport.
Montealegre & Keil (2000)	On August 4 1994, Mayor Webb announced a plan to develop "a temporary, low-tech alternative system for the Denver International Airport's high-tech baggage system."
Montealegre & Keil (2000)	On September 1, 1994, the City of Denver, United Airlines, and BAE, following intensive talks, struck a deal to break the baggage system contract and implement two separate systems. As a result of these negotiations, the original contract was divided into two pads: United was left managing the implementation of a simplified version of BAE's automated system to serve its Concourse B, and the City of Denver was left managing the implementation of a traditional baggage system to serve other airlines operating on Concourses A and C. Under the new arrangement, airlines other than United would not have access to the automated system unless BAE installed new telecar track and United granted rights for access.
Goetz & Szyliowicz (1997)	Once United obtained the automated baggage system concession for its concourse, the city decided (in the interest of efficiency and equity) that the system be expanded to the entire airport and thus requested bids for the job. The response was meager, so the city sought out BAE Automated Systems and was able to persuade them to build the expanded \$193 million system within the original fast-track time frame (de Neufville, 1994)
Szyliowicz & Goetz (1995)	One scholar, E. J. Feldman (1985), has recognized the importance of political variables. He states that the fate of megaprojects is determined not only by difficulties in forecasting but by such political factors as the nature of bureaucracies, the role of citizens, and how the financing and administration of these projects proceed.
Montealegre & Keil (2000)	the real problem was getting the airport open and that continued commitment to the IT-based baggage handling system would lead only to further delays. Before this point, the problem had been defined as "how to complete the automated baggage system as originally planned," whereas the new goal became "do whatever it takes to make the airport operational so that it can be opened as soon as possible."
Hickerson (2006)	The social and political context of the project also created high stakes, since several city-wide political campaigns were based on the airport's success or failure.
GAO (1995)	United Airlines also requested substantial modifications when it negotiated an agreement with the City. Most significantly, United requested an automated baggage handling system for Concourse B to ensure that nearly all of its transferring passengers' bags reached flights very quickly. At that time, the City planned to allow each airline to develop its own baggage system as long as this system did not interfere with any airport wide automated baggage system that the City might wish to install in the future.
Szyliowicz & Goetz	United Airlines finally reached an agreement with the City in June 1991,

(1995)	committing itself to 45 gates in return for even greater design changes and concessions which totalled \$204 million. Moreover, it was agreed that the airlines would pay the City no more than \$20 per enplaned passenger to use the new facility. United's other demands, however, were to profoundly affect the project's implementation. In addition to various concourse modifications, United insisted upon a fully automated, high speed baggage system that would move the passengers' luggage rapidly between the terminal and its concourse through a system of tunnels. A few weeks later, the City decided that, rather than having separate baggage systems for each airline, the United system should be extended to the entire airport (Russell, 1994).
Keil & Montealegre (2000)	At DIA, the bond repayment schedule for the airport forced decision-makers to confront the costs associated with continuing to pursue the airport-wide automated baggage-handling system.
Keil & Montealegre (2000)	Webb was forced to negotiate with United and BAE to reach a settlement that was acceptable to all major stakeholders. Ultimately, these negotiations led to a decision to fragment the baggage system contract, allowing United to continue working with BAE on a semi-automated system to serve its concourse (a course of action strikingly similar to that which United had initiated originally with BAE).
Russell (1994)	Facing ultimatums from bond-rating agencies (DIA's revenues were supposed to service the bonds) and airlines, the city had little choice but to stick with its opening date.
Survey respondent 31828	93. How would you characterize the extent to which Politics played a role on the software development project? No political behaviors exist on the project.
Survey respondent 31954	93. How would you characterize the extent to which Politics played a role on the software development project? Response: Political behaviors occur across the project's management hierarchy.
Survey Respondent 31960	93. How would you characterize the extent to which Politics played a role on the software development project? Response: Political behaviors involve external management and parties external to the project and/or parent organization.
Score	Overall Evaluation
0.0	Highly Political: Political behaviors involve external management and party's external to the project and/or parent organization.

Exhibit 67: POLT Evaluation

The overall score for the environment element was 10.4 of 25 possible points.

The overall DIA BHS scored 9/50 in the function element, 10.8/25 in the structure element, and 10.4/25 in the environment element. The overall score against the FSE Framework was 30.2 out of 100.

6. CONCLUSION

This case study has presented the important facts surrounding the design and implementation of the automated baggage handling system at the Denver International Airport. The 1st part of the study provided background material essential in

understanding the decision made to adopt a complex automated baggage handling system at DIA. The 2nd part of the study reviewed the scope of the baggage handling system and its supporting software system. The 3rd part of the case study reviewed the outcome of the baggage handling system design and implementation at DIA. The 4th and final part of the case study evaluated the design and implementation of the baggage handling system using the 60 measurement objects in the Function-Structure-Environment (FSE) Framework.

A complete interpretation of the evaluation against the FSE Framework has been provided in Chapter 6, Discussion of Results. If the reader is interested in additional details related to the DIA BHS a complete list of references is provided in the next section. Particularly thorough narratives are provided in Kerzner (1998; 2000) and Applegate, Montealegre, Knoop & Nelson (1996). Perspective related to airport planning are addressed in Szyliowicz & Goetz (1995) and Goetz & Szyliowicz (1997). Perhaps the most thorough review of the decision making surrounding the software is provided by Montealegre & Keil (2000).

7. DIA BHS CASE STUDY REFERENCES

- Applegate, L.M., Montealegre R., Knoop, C I. & Nelson H.J. (1996a). "BAE Automated Systems (A): Implementing the Denver International Airport Baggage-Handling System," *Harvard Business School Case 396-311*. Harvard: Harvard Business School Press.
- Applegate, L.M., Montealegre R., Knoop, C I. & Nelson H.J. (1996b). "BAE Automated Systems (B): Implementing the Denver International Airport Baggage-Handling System," *Harvard Business School Case 396-312*. Harvard: Harvard Business School Press.
- Auguston, K.A. (1994). "The Denver Airport: A Lesson in Coping with Complexity," *Modern Materials Handling*, Vol. 49, No. 12, pp. 40-46.

- de Neufville, R. (1994) "The Baggage System at Denver: Prospects and Lessons," *Journal of Air Transport Management*, Vol. 1, No. 4, Dec., pp. 229-236.
- Dempsey, P.S., Goetz, A.R., & Szyliowicz, J. S. (1997). *Denver International Airport: Lessons Learned*. McGraw-Hill, New York.
- Donaldson, A.J.M. (2002). "A Case Narrative of the Project Problems with the Denver Airport Baggage Handling System (DABHS)," *Software Forensics Centre Technical Report TR 2002-01*. London: Middlesex University, School of Computing Science.
- Gibbs, W.W. (1994). "Software's Chronic Crisis," *Scientific American*, Vol. 271, No. 3, pp. 86-95.
- Glass R.L. (1998). *Software Runaways: Lessons Learned from Massive Software Project Failures*. New York: Prentice Hall.
- Goetz, A.R. & Szyliowicz, J.S. (1997). "Revisiting Transportation Planning and Decision Making Theory: The Case of Denver International Airport," *Transportation Research*, Part A, Vol. 31, No. 4, pp. 263-280.
- Henderson, D.K. (1994). "It's in the Bag(s)," *Air Transport World*, Vol. 31, No. 9, pp. 54-58.
- Hickerson, T. B. (2006). *Failure at the Speed of Light: Project Escalation and De-escalation in the Software Industry*, (Master of Arts in Law and Diplomacy Thesis, Tufts University, 2006).
- Keil, M. & Montealegre, R. (2000). "Cutting Your Losses: Extricating Your Organization When a Big Project Goes Awry," *MIT Sloan Management Review*, Vol. 41, No. 3, pp. 55-68.
- Kerzner, H. (1998). *Project Management: A Systems Approach to Planning, Scheduling, and Controlling* (6th ed.). New York: John Wiley & Sons.
- Kerzner, H. (2000). *Applied Project Management: Best Practices on Implementation*. New York: John Wiley & Sons.
- Montealegre, R. & Keil, M. (2000). "De-escalating Information Technology Projects: Lessons from the Denver International Airport," *MIS Quarterly*, Vol. 24, No. 3, pp. 417-447.
- Rifkin, G. (1994). "What Really Happened at Denver's Airport," *Forbes*, August 29.
- Russell, J.S. (1994). "Is This Any Way to Build an Airport?" *Architectural Record*, Vol. 182, No. 11, pp. 30-36.

Szyliowicz, J.S. & Goetz, A.R. (1995). "Getting Realistic about Mega-project Planning: The Case of the New Denver International Airport," *Policy Sciences*, Vol. 28, No. 4, pp. 347-367.

United States General Accounting Office, Resources, Community, and Economic Development Division (1994). *New Denver Airport: Impact of the Delayed Baggage System* (GAO/RCED-95-35BR). Washington: Government Printing Office.

United States General Accounting Office, Resources, Community, and Economic Development Division (1995). *Denver International Airport: Statement of Michael Gryzkowicz; Director, Planning and Reporting, Resources, Community, and Economic Development Division* (GAO/T-RECD/AIMD-95-184). Washington: Government Printing Office.

APPENDIX G: FBI VCF SYSTEM CASE STUDY

1. INTRODUCTION

This case study will present the facts surrounding the design and effort to implement the Virtual Case File (VCF) system at the Federal Bureau of Investigation (FBI). The 1st part of the study will provide background material essential in understanding the decision to adopt a complex information system for handling case files at the FBI. The 2nd part of the study will review the replacement of the existing Automated Case Support (ACS) system with the VCF software system. The 3rd part of the case study will review the outcome of the effort to design and implement the VCF at the FBI. The 4th part of the case study will evaluate the FBI VCF project using the 60 measurement objects in the Function-Structure-Environment (FSE) Framework. The 5th and final part of the case study contains recent information about the case that directly impacted this study.

2. BACKGROUND

The background for the Case Study involves two key elements: the information technology (IT) available to the FBI and the management of IT projects across the Bureau. Both of these elements are essential in understanding why the FBI chose to cancel their complex virtual case file system.

Information Technology at the Federal Bureau of Investigation

In 2004 the FBI, the nation's premier law enforcement agency, had 12,000 special agents and 16,000 mission-support personnel who worked domestically in 56 field and 400 satellite offices, and overseas in 51 legal attaché offices in embassies and consulates. (GAO, 2004a) Agents investigated crimes and meticulously documented their work,

recording every detail and step on paper documents, which were assembled into *case files*. The system of paper documents could be traced back to J. Edgar Hoover [1895-1972] the Bureau's director from 1924 to 1972. The forms and methods were standardized and formed the basis for all FBI investigative data. Once the forms were completed and approved they were passed to a clerk who entered the data into the Automated Case System (ACS) and filed the paper forms as part of the official record (paper forms were required to be kept as part of the legal act of *discovery*).

The ACS was the FBI's primary investigative computer application and the centerpiece of its investigative record-keeping system. (FBI, 2002) ACS was designed to upload and store case files electronically. ACS was a DOS-based mainframe-application:

Based on the 1970s-era database Adabas and written in a programming language called Natural, both from Software AG, Darmstadt, Germany, the Automated Case Support system, which debuted in 1995, was antiquated even as it was deployed—and it is still being used today. Originally, agents and clerks accessed the program via vintage IBM 3270 green-screen terminals connected to a mainframe over dedicated lines. Eventually, the 3270 terminals were emulated on standard desktop PCs. By navigating complicated menus using function keys and keystroke commands, agents could do basic Boolean and keyword searches for things like an informant's name or the dates of wiretap surveillance, information related to cases they were working. (Goldstein, 2005, p. 27)

"The archaic Automated Case Support system—which some agents have avoided using—is cumbersome, inefficient, and limited in its capabilities, and does not manage, link, research, analyze, and share information as effectively or timely as needed" (Goldstein, 2005, p. 26).

The ACS's fatal flaw, though, is that it simply automated already onerous administrative chores. Over the course of its 95-year history, the FBI's bureaucracy has devised some 900 standard forms, to be filled out for everything from recording attendance (Form 420) to filing a memorandum (Form 467) to conducting an interview (Form 302). Until very recently,

the FBI's automation approach was "to just build macros for everything," You become a huge bureaucrat, doing one hour of investigation and seven hours of administration. (Kumagai, 2003, p. 31)

The situation at the FBI was several generations behind industry standards. The need to upgrade the IT systems at the FBI was apparent and the Bureau prepared and submitted IT modernization plans, but Congress failed to fund either the Information Sharing Initiative (ISI) in 1998 or the eFBI Program in 2000. The situation in September 2000 was as follows (DOJ, 2005, pp. 2-3):

- More than 13,000 of the FBI's desktop computers were 4 to 8 years old and could not run modern software.
- The communications capability (networks) between and within FBI offices was up to 12 years old.
- Most of the FBI's network components were no longer manufactured or supported.
- Most resident agency offices were connected to the network at speeds equivalent to a 56k modem.
- Agents were unable to reliably e-mail each other case-related information and often resorted to facsimiles.
- Agents were unable to e-mail U.S. Attorney Offices, other federal agencies, or local law enforcement agencies.

The FBI finally convinced the Congress to approve and fund a comprehensive IT infrastructure and systems modernization plan. In September 2000 the FBI Information Technology Upgrade Project (FITUP) was approved \$379.8 million was appropriated for its implementation. The FITUP project was renamed Trilogy and had three components.

1. Information Presentation Component (IPC) to provide hardware and operating system software.
2. Transportation Network Component (TNC) to provide the communications network.
3. User Application Component (UAC) to replace five investigative applications, which included the ACS, with a single application.

The agents and support staff at the FBI were anxious to receive the modern information technology. They all had modern systems at home and were frustrated by the lack of technology at the Bureau. The lack of modern technology affected everyone at the Bureau, even the Director. Upon arriving at the FBI on September 4, 2001, Director Mueller asked that Microsoft Office be installed on his desktop. He was told "We can put it on there, but it won't be compatible with anything else in the FBI," "He hit the roof." (Kumagai, 2003, p. 28) After the events of September 11th the need for Trilogy became imperative.

Management of Information Technology at the FBI

Information technology, like everything at the FBI, was governed by the Bureau's culture of secrecy and the *need-to-know* mindset, which permeated into decision making and organizational structures. The management of information technology resources was decentralized, and best described as stove-piped. Each of the FBI's 23 divisions had their own IT budgets and systems. Because of this freedom, and control of the supporting funds, the FBI had 40-50 different investigative databases and applications, many of which duplicated functions and information found in another system (Goldstein, 2005). The FBI's Chief Information Officer (CIO), mandated by the Information Technology

Management Reform Act (commonly referred to as the Clinger-Cohen Act), was filled by a number of qualified managers, none of whom had any real power, the funding was key. It was not until 2004 that the FBI CIO finally received authority over all IT budgets and systems.

In addition to the structural problem, there was a problem with executive level IT leadership. Between November 2001 and November 2004 the FBI had 5 CIOs, which is an average term of 219 days or slightly over 7 months. Turnover of the executive responsible for the development and implementation of the strategic vision for all IT programs created turmoil and ever-changing direction. Exhibit 68 lists the 5 FBI CIOs that served since the position was created in November 2001.

Incumbent	Dates of Service
Bob E. Dies	Nov 2001 to May 2002
Mark Tanner	May 2002 to July 2002
Darwin John	Jul 2002 to May 2003
W. Wilson Lowery	May 2003 to Dec 2003
Zalmai Azmi	Dec 2003 to present

Exhibit 68: FBI Chief Information Officers

The single most damaging factor that may be attributed to the rapid turnover of the CIO was the failure to produce a comprehensive blueprint for IT systems at the FBI. Prior to the start of a major IT initiative like Trilogy, an IT roadmap (enterprise architecture) is designed and approved by the functional leadership of the organization. The FBI Director and Associate Directors would be expected to be involved in the creation of the enterprise architecture, with the CIO acting as a facilitator. The involvement of the top levels of the organization ensures that all information technology systems and investments directly support the mission and needs of the enterprise; in this case, the FBI. This deficiency was reported to the FBI by the Government Accountability Office (GAO 2003, 2004a, 2005), the FBI Inspector General (FBI, 2002),

and the National Research Council (McGroddy & Lin, 2004). The National Research Council made the following category 1 finding in their review of the Trilogy program:

The committee believes that if the FBI's IT modernization program is to succeed, the FBI's top leadership, including the director, must make the creation and communication of a complete enterprise architecture a top priority. This means that they must be personally involved and invested in the key decisions that the process will require be made, such as the tradeoffs between the security of and access to information in the various data sources that are used in criminal investigation and counterterrorism efforts. Indeed, it is critical that the director be well versed in, and comfortable with, the operational aspects of the enterprise architecture and their overall linkage to the high-level system design. (McGroddy & Lin, 2004, p. 49)

Closely associated with the management problems in the CIO office was contract management. The FBI's culture demanded tight security and kept most work *in-house*. As a result the FBI did not have a robust centrally controlled contract management capability and no significant experience with either IT contracts or contractors. Because of this deficiency a number of major IT contract errors were made on Trilogy:

- Because the Bureau did not have an experienced IT contracting staff they chose to use the General Services Administration's (GSA) Millenia contracting services. GSA's Federal Technologies Services' Federal Systems Integration and Management (FEDSIM) Center provides contracting services for Federal government agencies. FEDSIM had a number of qualified contractors on the Millenia contract and was able to convince the FBI that the contract for Trilogy could be submitted for bids and awarded quickly. FEDSIM would act as the contracting office, receiving a fixed percentage of the contract as a fee. (DOJ, 2005)

- Because the Bureau did not have experience contracting for major IT services the Justice Department contracting managers convinced the decision-makers in FBI Headquarters to award two contracts for Trilogy. In May of 2001 the bureau awarded the contract for the infrastructure components (IPC/TNC) of Trilogy to DynCorp of Reston, VA. In June 2001 the Bureau awarded the contract for the software component (UAC) of trilogy to Science Applications International (SAIC) of San Diego. (DOJ, 2005)
- Instead of paying a fixed price for the hardware, networks, and software, the FBI used cost-plus-award fee (CPAF) contracts. CPAF contracts would pay the cost of all labor and materials plus additional money if the contractor managed costs commendably. This was not a best-practice approach for system acquisition.
- The Trilogy contracts lacked the specificity necessary to determine whether the project was making adequate progress within schedule and budget constraints. (McGroddy & Lin, 2004)

3. RELACEMENT OF ACS

The replacement of ACS is characterized by three elements, the development of the requirements for VCF, the introduction of IT project management at the FBI, and the construction of the VCF software.

VCF Software Requirements Development

With the Trilogy IT contract in place the FBI began replacement and upgrade of its basic computer hardware, software, and network infrastructure. The \$534 million Trilogy project was slated to give each of the 11,400 agents and 16,400 mission-support

employees a Dell Pentium desktop PC running Microsoft Office, with secure, high speed connections to FBI headquarters and hundreds of field and satellite offices within 3 years. (Kumagai, 2003, p. 28)

The User Application Component (UAC) was intended to provide the FBI with (FBI, 2002):

- Improved capabilities to communicate inside and outside the FBI.
- Access to information from internal and external databases that is properly authorized using primarily commercial products.
- The capability to evaluate cases and patterns of crimes through the use of commercial and FBI-enhanced analytical and case management tools.
- The ability to find information in FBI databases without having to know where it is, and to search all FBI databases with a single query through the use of intelligent search engines.

The Trilogy schedule called for delivery of the UAC by June 2004. However, given the urgent need for improved IT capabilities, the FBI was looking for ways to accelerate the development of Trilogy. The FBI reported that they had devised a plan to complete the infrastructure elements (IPC/TNC) one year ahead of schedule, completing deployment by June 2003. This put tremendous pressure on SAIC to accelerate the delivery of the UAC. SAIC developed a plan to put a web-based front end on the existing mainframe ACS. This would move the outdated *green-screen* technology to a windows-based point and click technology which could be completed by July 2002. The accelerated plan called for the UAC, now renamed the Virtual Case File (VCF), to be delivered in two phases. The 1st phase would include the basic case file application and the migration of

data from ACS and other applications, with a completion in December of 2003. The 2nd phase would include upgrades to three other existing applications and add audio/video streaming and content management capability, with a completion in June 2004.

After the events on September 11th, and in response to public outcry and congressional pressure, Director Mueller brought in a group of computer-literate agents to review the FBI's IT applications strategy. After a careful review of the modern web-based front end SAIC was building for the archaic ACS the FBI determined

The bureau needed an entirely new database, graphical user interface, and applications, which would let agents search across various investigations to find relationships to their own cases. The new case management system would host millions of records containing information on everything from witnesses, suspects, and informants to evidence such as documents, photos, and audio recordings. (Goldstein, 2005, p. 28)

In December 2001 the FBI asked SAIC to halt development on the web front-end for ACS and to design a new application, database, and graphical user interface to completely replace the ACS system. The new system must be able to search on not just text but also photos, video, and audio records, all with a view to detecting and connecting the traces of terrorist and criminal activity.

With no detailed description of the FBI's functional processes and IT infrastructure (i.e. enterprise architecture) as a guideline, a team of FBI agents began the process of characterizing investigative processes such as witness interviews and surveillance operations and mapping them to the FBI's existing software and databases. The team of up to 40 FBI subject matter experts worked with engineers from SAIC, constructing diagrams and flowcharts of how ACS actually operated (the *as-is* state) and then transforming this into how they wanted the new VCF to operate in the future (the *to be* state). FBI Director "Mueller himself attended one of these meetings to tell the agents

to design a system that would work best for them and not to feel constrained by 50-year-old business rules.” (Goldstein, 2005, p. 29)

In January 2002 SAIC conducted Joint Application Development (JAD) sessions where the FBI functional experts and SAIC engineers would determine the specific functions VCF would perform. The outcomes of these meetings were the inputs to the functional requirements document that would guide SAIC's application designers and programmers. When the JAD sessions finished in July of 2002 the earlier vision for the replacement of the ACS and four other systems had changed and a new electronic workflow with all systems integrated into a single process had been adopted. (DOJ, 2005) The requirements document would not only create an entirely new case management system but new functional processes. In order to do this the Trilogy contract required modification to account for added labor and the new aggressive 22 month completion date for the VCF. SAIC's contract labor ceilings were increased to permit the additional labor hours associated with the change. However, the FBI's inexperienced IT project and contract management staffs failed to include system acceptance criteria (which will be important later in the case), milestones, or a formal schedule in the contract change. Furthermore, the new plan included an implementation strategy called a *flash cutover*, in which the old ACS would be turned off and the new VCF turned on, overnight. In the IT world this is a very risky maneuver. SAIC and the FBI embarked on this path without an established backup plan.

By the end of 2002 the formal requirements for the VCF element of Trilogy were complete and a highly detailed 800 page requirements document was produced. With the

requirements frozen, approval for development was granted, and SAIC went into full production mode.

VCF Project Management

In response to the array of management audits directed at the FBI, an experienced project manager was required for the Trilogy Project. The project manager was to work directly for the FBI Director as the Project Management Executive (PME) for a newly created Office of Program Management. This office was to act as the single point of contact for all FBI IT management and would be responsible for the development and implementation of Trilogy.

When the PME arrived an immediate review of the overall Trilogy program was conducted. This was the first integrated assessment of the two independent contract schedules (IPC hardware and TNC networks were under contract to DynCorp and UAC/VCF software was under contract to SAIC). DynCorp had no schedule. "In contrast, SAIC, with its programmers pecking away at its secure data center in Vienna, Va., always had a detailed schedule posted prominently in the 'war room' there." (Goldstein, 2005, p. 30) The review revealed that the hardware and network communications elements of Trilogy would not be delivered in July of 2002 as the contract delivery called for. The DynCorp team "... didn't have a detailed schedule that mapped out how it would deploy, integrate, and test the new computers and networks." (Goldstein, 2005, p. 30) In May 2002 the PME informed the FBI Director of the delay. During the conversation with the Director the PME tried to understand why DynCorp had agreed to the aggressive 22 month schedule, asking

Did somebody come to you and say, okay, Mr. Director, sir, you can have it sooner, but it's going to cost you this much more money or you're going to have to do without something. (Goldstein, 2005, p. 32)

His response was

No, nobody ever told me that. (Goldstein, 2005, p. 32)

The PME responded

Well, lesson No. 1: faster, cheaper, better. Pick two, but you can't have all three. (Goldstein, 2005, p. 32)

It would appear that the project had escalated due to the *mum effect* (Keil & Robey (1999) referenced in the research, where people knew of the problem but did not report it to management. The PME had inherited a project with escalating costs and expanding schedules and the Director accepted the only alternative: a better system.

At about the same time the FBI's IT management changed (see Exhibit 68). Bob Dies, the 1st CIO left and Mark Tanner took the job for 3 months. In July Darwin John, an FBI outsider and former CIO of the Mormon Church became FBI CIO.

By the end of 2002 the requirements for the VCF element of Trilogy had stabilized and the functional design was frozen and approved for development. The FBI reported this progress to Congress and received an additional appropriation of \$124 million. At the same time the FBI's Inspector General issued a report critical of the management of the Trilogy project citing missed milestones and uncertainty about cost, schedule and technical requirements. (FBI, 2002)

VCF Software Construction

The SAIC development team, in response to the increased project pace, was split into eight groups. The idea was to have the groups work in parallel and then integrate the work at the end. In the IT world this is another risky maneuver. The risk comes from

underestimating the time required to complete integration testing for independently developed software modules. The failure to increase integration time and associated testing was an additional error. The pace required to meet the accelerated 22 month schedule was terrific.

SAIC used a spiral method of development for VCF. The method required the SAIC programmers to iteratively write software, show it to the FBI, make changes, and repeat the process. A formal software configuration management process and configuration management board (CCB) was in place to track the software and approved changes. The problems that arose were part of the FBI's culture. The FBI agents were starting to tell the SAIC engineers and programmers *how to* build the system instead of focusing on *what* the system needed to do. The culture in the FBI was "We're going to tell you how to do it." (Goldstein, 2005, p. 32) When a dispute arose between the FBI and SAIC a formal change was submitted to the CCB for review. Between December 2002 and December 2003 roughly 400 change requests were received. (Goldstein, 2005)

The VCF code construction team was given some unwelcome schedule cover when, in March 2003, Computer Sciences Corporation (CSC) which had acquired DynCorp that month, informed the FBI that the IPC hardware and TNC networks elements of Trilogy would be delayed until October. CSC slipped the dates two more times: in August, the October 2003 date became December 2003 and in October 2003, December 2003 became April 2004. The problems were attributed to the FBI's request to change the e-mail system, the FBI's inability to precisely account for the existing IT infrastructure components and networks and, according to the FBI Inspector General's

2005 audit, obtaining the components needed to connect the field offices to the wide area network (FBI, 2005).

While the FBI management was resolving the schedule delays related to the infrastructure problems SAIC was besieged by additional changes to the VCF. Unable to test the VCF on the delayed infrastructure (hardware and network) the VCF continued in development. Because SAIC had made the decision to split the team into 8 development groups, changes to the maturing code became much more difficult. A change in 1 of the 8 areas required integration that could potentially affect the code in any of the other 7 system areas.

SAIC voiced their concern about not being able to properly test the VCF system on the intended infrastructure. SAIC was worried about the implementation decision that called for the *flash cutover*, where ACS would be shut off and VCF turned on. The risk for failure was significantly increased by the inability to access the intended hardware and software. However, SAIC informed the FBI that, in order to meet the compliance element of the contract, they would deliver VCF in December. On December 13, 2003, SAIC delivered the initial version of VCF to the FBI

4. OUTCOME OF THE VCF SYSTEM

In December 2003 another IT management change was made at the FBI. W. Wilson Lowery, the FBI's 4th CIO resigned and Zalmay Azmi became the new FBI CIO. One of Azmi's first acts was to reject SAIC's delivery of the VCF. The FBI stated that there were 17 *functional deficiencies* it wanted SAIC to fix before the system was deployed. SAIC argued that some of these deficiencies were based on changes requested by the FBI to the system's requirements. Being unable to agree on the cause an arbitrator

was used and ruled that of the 59 issues derived from the original 17 deficiencies, 19 were requirements changes (FBI problems) and 40 were errors (SAIC problems).

While the arbitration was taking place Azmi's team of functional experts created a series of investigation scenarios that would be tested on the VCF system. This is standard IT practice, but the scenarios are developed as part of the requirements development process (mentioned earlier), before the system is constructed. When the investigation scenarios were run on the VCF 400 more system deficiencies were identified. SAIC told the FBI that they could correct the deficiencies in one year at a cost of an additional \$56 million; Azmi immediately rejected the SAIC proposal.

Coincidentally, the Computer Science and Telecommunications Board of the National Research Council had just finished their report on Trilogy (McGroddy & Lin, 2004). Azmi's chose to listen to the Board and their two major recommendations: (1) the plan to use a *flash cutover* as an implementation strategy for the transition from the ACS to the VCF should not be continued because failure would be catastrophic for the bureau; (2) the FBI should create an enterprise architecture to guide development of its IT portfolio. It is interesting to note that committee had made both of these recommendations in September 2002, and according to McGroddy, both suggestions had been ignored until Azmi took charge. (Goldstein, 2005)

In June, Azmi hired the Aerospace Corporation to conduct an external review of the VCF that SAIC had delivered in December 2003. Azmi wanted the independent review to assess whether the VCF system requirements were met and to make a recommendation about what to do with the VCF. The report was completed in December 2004 and released in January 2005.

The report recommended discarding the VCF and starting over with a COTS-based solution. The contractor concluded that a lack of effective engineering discipline had led to inadequate specification, design, and development of VCF. Further, the contractor could find no assurance that the architecture, concept of operations and requirements were correct or complete, and no assurance that they could be made so without substantial rework. In sum, the contractor reported that VCF was a system whose true capability was unknown, and whose capability may remain unknown without substantial time and resources applied to remediation. (FBI, 2006, p. 51)

The FBI officially terminated the VCF system project in April 2005.

The VCF was one of the most highly publicized software failures in history. Instead of automating the FBI's paper-based work environment, allowing agents and intelligence analysts to share vital investigative information, and replacing the obsolete Automated Case Support (ACS) system, the FBI spent \$170 million (\$51 million more than the original estimate) and owns 700,000 lines of unusable code.

4. FBI VCF SYSTEM PROJECT EVALUATION

In this section the FBI VCF system has been evaluated against the systemically-based FSE Framework. The framework has three elements; Function, Structure, and Environment. The evaluation matched the empirical facts concerning the VCF system against each framework element. The evaluation was conducted with the assistance of a qualitative software program called NVivo. The NVivo program served as the database for the empirical evidence used in the case study analysis. Each of the 14 journal articles in the reference section and the results of the interview questionnaires were entered into the NVivo tool. The hierarchical structure of the FSE Framework was also entered into NVivo where it became a *node tree*. The hierarchical node tree would serve as the collection point for the empirical facts *coded* from the journal articles.

The researcher manually *coded* each of the 14 journal articles and questionnaires. Coding is the term used for a specific type of analysis; the analysis that involved differentiating and combining data associated with the phenomena under investigation. In this case the phenomenon under investigation was the performance of the FBI VCF system software development project; and the data was the empirical evidence in the journal articles and questionnaires. The researcher reviewed all of the empirical evidence and selected relevant *chunks* of varying size – words, phrases, sentences, or whole paragraphs, connected to a specific measurement object in the framework. The *chunks of information* were called free nodes. 239 free nodes were coded from the journal articles.

The hierarchical node tree (the FSE framework) served as the collection point for the free nodes (the *coded* empirical facts). The node tree had 60 collection points, which corresponded with each of the 60 FSE Framework measurement objects. The 239 free nodes were moved to relevant positions on the hierarchical node tree. The completed node tree served as the starting point for evaluation of the FBI VCF system software development project.

Function Element

The functions element of the FSE framework had five areas that evaluated 26 measurement objects. The areas address software development, improvement & training processes, infrastructure, life cycle support processes, and management worth a total of 50 points. Each area will be evaluated separately.

1. Development: The development area addressed four measurement objects:

a. Requirements Management (REQM)

Requirements Management (REQM): The process to manage the requirements of the project's products and product components and to identify inconsistencies between those requirements and the project's plans and work products.	
Source of Empirical Evidence	Empirical Evidence
GAO (2004a)	given the FBI's experience on the Virtual Case File, particularly with regard to requirements management and the bureau's reported efforts and plans going forward. Specifically, it is critical for the FBI to examine and control its requirements in the context of what capabilities are to be addressed
GAO (2005b)	The actual consequences of not having effective requirements development and management policies and procedures can be seen in the performance of the bureau's Trilogy project
Score	Overall Evaluation
0.5	The software development project performed one REQM process.

Exhibit 69: REQM Evaluation

b. Requirements Development (RD)

Requirements Development (RD): The process which produced and analyzed customer, product, and product-component requirements.	
Source of Empirical Evidence	Empirical Evidence
Goldstein (2005)	It was a classic case of not getting the requirements sufficiently defined in terms of completeness and correctness from the beginning. And so it required a continuous redefinition of requirements that had a cascading effect on what had already been designed and produced.
Goldstein (2005)	the ever-shifting nature of the requirements, and the agents'
Goldstein (2005)	Ideas captured in these sessions formed the basis of the requirements document that guided SAIC's application designers and programmers.
GAO (2005b)	inadequately defined requirements
Goldstein (2005)	poorly defined and slowly evolving design requirements
GAO (2004a)	poorly defined requirements,
Goldstein (2005)	Recalling the Web pages the agents would bring into the JAD sessions to demonstrate how they wanted the VCF to look, Higgins blamed both SAIC and the agents for creating the overstuffed requirements document.
FBI (2002)	Since January 2002, the FBI and the contractor were participating in a Joint Application Development planning process to define and prioritize the users' operational requirements
Harris (2005)	That project also suffered from a broad scope, ill-defined requirements and expectations,
Hickerson (2006)	the contract was signed in 2001, the FBI did not identify all the requirements right away. Even when the character of the project changed in 2002, not all the requirements were spelled out yet.
GAO (2004a)	the Inspector General reported that the original delivery date for Trilogy's first two components (Transportation Network Component and Information Presentation Component) slipped 8 months, in part due to inadequately defined requirements
US Senate (2005b)	the most damaging aspect of this development environment was the ever-shifting nature of the requirements
McGroddy & Lin (2004)	the processes supporting the intelligence mission were not included in the VCF design

Hickerson (2006)	The requirements document that was finally drafted was about 800 pages long
Kumagai (2003)	The Virtual Case File system looks to be better, she notes. "You can't just automate, you have to reengineer," she says. This time around, experienced street agents are being brought into the development process. "Every form is being examined. Can we get rid of it? Can we do this automatically? It should make us incredibly more productive." Data input into the system is being streamlined, and the extra bandwidth being added through the Trilogy upgrade will allow photos and video to be uploaded and downloaded. Querying the system will yield, among much else, a linked diagram of where each relevant document resides.
US Senate (2005a)	We did not have a complete set of defined VCF requirements when the original contract was signed in June 2001
Kumagai (2003)	Replacing the FBI's ancient DOS-based Automated Case Support (ACS) database with a more user-friendly Windows-based system that can search on not just text but also photos, video, and audio records. Known as the Virtual Case File system, it's set to come online by the end of 2003.
Goldstein (2005)	But the User Applications Component, which would ultimately become the VCF, staked out the most ambitious goals. First, it was to make the five most heavily used investigative applications—the Automated Case Support system, IntelPlus, the Criminal Law Enforcement Application, the Integrated Intelligence Information Application, and the Telephone Application—accessible via a point-and-click Web interface. Next, it would rebuild the FBI's intranet. Finally, it was supposed to identify a way to replace the FBI's 40-odd investigative software applications, including ACS.
McGroddy & Lin (2004)	The Virtual Case File, the user application component of Trilogy, is a custom-designed software application that is intended to facilitate case file management by integrating data from older, separate investigative systems, including the Automated Case Support (ACS) system, and eventually replacing them. The VCF is intended to create efficiencies in entering case-related information by reducing the number of steps in filing documents and to facilitate the storage and retrieval of data for wider access, tracking, and analysis of case-related data.
DOJ (2005)	During the initial years of the project, the FBI had no firm design baseline or roadmap for Trilogy. According to one FBI Trilogy project manager, Trilogy's scope grew by about 80 percent since initiation of the project.
DOJ (2005)	The design of and schedule for the UAC portion of Trilogy were substantially modified after the September 11 attacks. The most significant design change was eliminating the web-enablement of ACS and instead developing an enterprise-wide solution to replace ACS,
DOJ (2005)	the lack of fully developed requirements for the project negatively affected schedule, cost, technical, and performance baselines.
DOJ (2005)	One major reason for the delays and cost growth in the overall project was a lack of specific design requirements for each of the project components.
DOJ (2005)	The VCF plan that resulted from these JAD sessions in 2002 rejected the previous plan to replace the five separate investigative applications in favor of developing an entirely new electronic workflow with systems that are integrated into one process. The VCF concept not only would change where the data for case files is stored, but also would create an entirely new environment in which agents, analysts, and support personnel operate.
DOJ (2005)	The FBI refined the VCF concept through Joint Application Development (JAD) sessions held between January and June 2002. The JAD sessions brought together FBI representatives to determine what applications were needed to support the case management and information requirements of FBI agents, analysts, and support personnel; UAC contractor representatives to determine what applications could be created; and infrastructure contractor

	representatives to ensure that the applications could be supported by the groundwork that was being developed.
DOJ (2005)	poorly defined and evolving design requirements
DOJ (2005)	the lack of a firm understanding of the design requirements by both the FBI and the contractors. Trilogy's design requirements were ill-defined and evolving as the project progressed
Score	Overall Evaluation
0.5	The software development project performed one RD process.

Exhibit 70: RD Evaluation

c. Technical Solution (TS)

Technical Solution (TS): The process to design, develop, and implement solutions to requirements.	
Source of Empirical Evidence	Empirical Evidence
Hickerson (2006)	"The culture within the FBI was, 'We're going to tell you how to do it'
McGroddy & Lin (2004)	The committee is also concerned that the VCF's current design and technical specifications lack the flexibility needed to incrementally improve the application
McGroddy & Lin (2004)	the current implementation of the VCF appears to have embedded the workflows describing how information is to be entered, reviewed, and used. Embedding the workflow in the application (that is, hard-wiring it) will make any such changes in the future much more difficult (more expensive and slower) to implement
McGroddy & Lin (2004)	the data models of the IDW presented to the committee (and the VCF for that matter) were far too abstract to be very useful
McGroddy & Lin (2004)	The IDW appeared to have been designed to overwrite old copies of databases with newer copies, so data can be there one day and not the next (that is, the IDW is not equipped to handle time-series of versioned data).
McGroddy & Lin (2004)	no prototype has been developed for any of the major components of Trilogy (the Trilogy network or the VCF) or for the IDW
Goldstein (2005)	trial-and-error, 'We will know it when we see it' approach to development."
DOJ (2005)	lack of an Enterprise Architecture
DOJ (2005)	The group's decision paper cited 37 basic design flaws, including network, server, and storage infrastructure issues, operating system and software issues, application issues, and problems with the test plan. The lack of redundancy and resiliency in the system was seen as a major flaw because the design failed the basic "can of soda" test.
Score	Overall Evaluation
1.0	The software development project performed one TS process.

Exhibit 71: TS Evaluation

d. Product Integration (PI)

Product Integration (PI): The process that assembled the product from the product components, ensured that the product, as integrated, functioned properly, and delivered the product.	
Source of Empirical Evidence	Empirical Evidence
GAO (2005b)	lack of integration planning
McGroddy & Lin (2004)	the FBI must allow adequate time for testing before any IT application (including the VCF) is deployed, even if dates of initial operational capability are delayed
McGroddy & Lin (2004)	the FBI runs a very high risk that its planned "flash cutover" from the old ACS

	system to the VCF will cause mission-disruptive failures and further delays
US Senate (2005b)	the new VCF, in spite of its then undefined requirements, would not be implemented via a low risk, evolutionary strategy, but rather would be built as a grand design in record time and be implemented all at once in a "flash cutover" from the legacy systems to the new VCF. SAIC informed the Bureau this was a high-risk strategy
GAO (2004a)	Trilogy funding grew from an original estimate of \$379.8 million to \$596 million, due in part to the lack of integration planning for one of the three components of Trilogy
McGroddy & Lin (2004)	implement the VCF in what it describes as a "flash cutover." That is, the VCF would be rolled out for employee use all over the bureau simultaneously (or nearly so).
Score	Overall Evaluation
1.5	The software development project performed one PI process.

Exhibit 72: PI Evaluation

2. Software Improvement & Training Processes: The software improvement and training process area addressed five measurement objects:

a. Organizational Process Focus (OPF)

Organizational Process Focus (OPF) : The process that planned and implemented organizational process improvement based on a thorough understanding of the strengths and weaknesses of the organization's processes and process assets.	
Source of Empirical Evidence	Empirical Evidence
None	There is no discussion of OPF in any of the literature.
Score	Overall Evaluation
0.0	The software development project did not perform the OPF process.

Exhibit 73: OPF Evaluation

b. Organizational Process Design (OPD)

Organizational Process Design (OPD) : The process to establish and maintain a usable set of organizational process assets?	
Source of Empirical Evidence	Empirical Evidence
None	There is no discussion of OPF in any of the literature.
Score	Overall Evaluation
0.0	The software development project did not perform the OPD process.

Exhibit 74: OPD Evaluation

c. Organizational Process Performance (OPP)

Organizational Process Performance (OPP): The process to establish and maintain a quantitative understanding of the performance of the organization's set of standard processes in support of quality and process-performance objectives, and provide the process performance data, baselines, and models to quantitatively manage the project.	
Source of Empirical Evidence	Empirical Evidence
None	There is no discussion of OPP in any of the literature.
Score	Overall Evaluation
0.0	The software development project did not perform the OPP process.

Exhibit 75: OPP Evaluation

d. Organizational Innovation and Deployment (OID)

Organizational Innovation & Deployment (OID): The process to select and deploy incremental and innovative improvements that measurably improve the organization's processes and technologies.	
Source of Empirical Evidence	Empirical Evidence
None	There is no discussion of OID in any of the literature.
Score	Overall Evaluation
0.0	The software development project did not perform the OID process.

Exhibit 76: OID Evaluation

e. Organizational Training (OT)

Organizational Training (OT): The process to develop the skills and knowledge of people so they could perform their roles effectively and efficiently.	
Source of Empirical Evidence	Empirical Evidence
None	There is no discussion of OT in any of the literature.
Score	Overall Evaluation
0.0	The software development project did not perform the OT process.

Exhibit 77: OT Evaluation

3. Software Project Infrastructure: The software development project infrastructure area addressed two measurement objects:

a. Use of Software Tools (TOOL)

Use of Software Tools (TOOL): The capability and integration of the tool suite used in developing the software on the project.	
Source of Empirical Evidence	Empirical Evidence
None	There is no discussion of TOOL in any of the literature.
Score	Overall Evaluation
0.6	Assumed to be Nominal: The software development project tools were basic life-cycle tools, with moderate integration.

Exhibit 78: TOOL Evaluation

b. Multi-site Development (SITE)

Multi-site Development (SITE): The software development project team's co-location and communications profile?	
Source of Empirical Evidence	Empirical Evidence
McGroddy & Lin (2004)	the team also needs an environment that facilitates working effectively, such as proximity of offices, meeting spaces, and areas in which information can be passed informally over lunch or in a chance hallway encounter
Score	Overall Evaluation
0.4	Nominal. The software development site was multi-city and used narrow band e-mail for communications.

Exhibit 79: SITE Evaluation

4. Software Life Cycle Support: The software life cycle support area addressed eight measurement objects:

a. Process and Product Quality Assurance (PPQA)

Process & Product QA (PPQA): The process to provide staff and management with objective insight into processes and associated work products.	
Source of Empirical Evidence	Empirical Evidence
Goldstein (2005)	delivered 700 000 lines of code so bug-ridden and functionally off target that this past April, the bureau had to scrap the US \$170 million project, including \$105 million worth of unusable code.
Score	Overall Evaluation
0.0	The software development project did not perform the PPQA process.

Exhibit 80: PPQA Evaluation

b. Configuration Management (CM)

Configuration Management (CM): The process that established and maintained the integrity of work products using configuration identification, configuration control, configuration status accounting, and configuration audits.	
Source of Empirical Evidence	Empirical Evidence
FBI (2002)	Because the UAC portion is focused on making significant changes to, or possibly complete replacements of, five of the FBI's investigative systems, documentation for the exact configuration of these systems is critical to designing the requirements for UAC. According to a senior FBI official, the FBI must know what it has before it can define the right solution to fix the problem. "Lack of" documentation for the configuration of these five investigative systems has caused the FBI to engage in a process of reverse engineering.
FBI (2002)	Not having the documentation of the configuration of these five investigative systems has caused the FBI to engage in a process of reverse engineering.
Score	Overall Evaluation
0.5	The software development project performed one CM process.

Exhibit 81: CM Evaluation

c. Measurement & Analysis (MA)

Measurement & Analysis (MA): The process to develop and sustain a measurement capability used to support management information needs.	
Source of Empirical Evidence	Empirical Evidence
McGroddy & Lin (2004)	An effective program management function will provide the FBI with a focal point for monitoring and collecting project data and allow for the reporting of the progress of active IT projects based on well-defined metrics
US Senate (2005a)	We are updating an IT Metrics program that identifies and measures IT performance according to industry standards, government regulations, and Earned Value Management System (EVMS) principles
DOJ (2005)	the Budget Unit of the FBI's Information Resources Division was not reconciling or updating portions of the Trilogy tracking report, which resulted in discrepancies in the dollar amounts reported to management.
DOJ (2005)	the FBI's Financial Management System did not capture detailed Trilogy-related expenditures, while numerous entities tracked and monitored specific segments of the operation.
Score	Overall Evaluation
0.0	The software development project did not perform the MA process.

Exhibit 82: MA Evaluation

d. Verification (VER)

Verification (VER): The process to ensure that selected work products met their specified requirements.	
Source of Empirical Evidence	Empirical Evidence
FBI (2002)	a contractor performing independent verification and validation work
Score	Overall Evaluation
0.5	The software development project performed one VER process.

Exhibit 83: VER Evaluation

e. Validation (VAL)

Validation (VAL): The process to demonstrate that a product or product component fulfilled its intended use when placed in its intended environment.	
Source of Empirical Evidence	Empirical Evidence
FBI (2002)	a contractor performing independent verification and validation work ¹
Score	Overall Evaluation
0.5	The software development project performed one VAL process.

Exhibit 84: VAL Evaluation¹Note: This is a duplicate node also reported for the verification (VER) process.

f. Causal Analysis and Resolution (CAR)

Causal Analysis & Resolution (CAR): The process to identify causes of defects and other problems and take action to prevent them from occurring in the future.	
Source of Empirical Evidence	Empirical Evidence
None	There is no discussion of OT in any of the literature.
Interview	The interview subject stated that this process was not directly addressed.
Score	Overall Evaluation
0.0	The software development project did not perform the CAR process.

Exhibit 85: CAR Evaluation

g. Joint Reviews (JR)

Joint Reviews (JR): The process in which the customer and the project team evaluate the status and products of an activity as appropriate, were utilized on the software development project.	
Source of Empirical Evidence	Empirical Evidence
US Senate (2005a)	However, when SAIC delivered the product to us in December 2003, we immediately identified a number of deficiencies in VCF that made it unusable. Upon further examination, we discovered nearly 400 problems with the software and, in April 2004, provided SAIC with a document outlining the corrections needed
US Senate (2005b)	In December 2003, we delivered an evaluation copy of the VCF system. The FBI reviewed the product and identified 17 deficiencies, some of which were actually more changes in requirements
Goldstein (2005)	Sometimes Depew's team had only two days to review a batch of code. Agents would pull all-nighters to get the evaluation finished, "and in the next iteration their comments wouldn't be taken into account,"
McGroddy & Lin (2004)	specific milestones, frequent contract reviews,
Hickerson (2006)	there are 17 deficiencies, we decomposed the 17 deficiencies, they turned into 59 deficiencies. Then we had a two-week sit-down with SAIC [Science Applications International Corporation] and those 59 turned into 400 deficiencies... You know, I have a base-line software that I was told that 90 percent was ready, yet they were asking for another \$56 million to develop the other 10 percent
Goldstein (2005)	Under Azmi's direction, the FBI rejected SAIC's delivery of the VCF. The bureau found 17 "functional deficiencies" it wanted SAIC to fix before the system was deployed.
Score	Overall Evaluation
1.0	Nominal: Joint reviews were used in a few process areas.

Exhibit 86: JR Evaluation

h. External Auditing (EA)

External Auditing (EA): The process, in which an external management entity determines compliance with requirements, plans, and contract as appropriate, were utilized on the software development project.	
Source of Empirical Evidence	Empirical Evidence
Goldstein (2005)	In June, the FBI contracted an independent reviewer, Aerospace Corp., in El Segundo, Calif., to review the December 2003 delivery of the VCF to determine, among other things, whether the system requirements were correct

	and complete and to recommend what the FBI should do with the VCF.
Goldstein (2005)	Of the 59 issues and sub issues derived from the original 17 deficiencies, the arbitrator found that 19 were requirements changes—the FBI's fault; the other 40 were SAIC's errors.
FBI (2002)	The FBI had three internal assessments performed concerning the management of the Trilogy project. These assessments were done by the FBI's Inspection Division, CJIS Division
McGroddy & Lin (2004)	The FBI reported that it had undergone more than 100 investigations and audits in the IT area
McGroddy & Lin (2004)	The FBI should seek independent and regular review of its enterprise architecture as it develops by an external panel of experts with experience in both operations and technology / architecture
FBI (2002)	The IRD hired an outside contractor to obtain an independent perspective on Trilogy. The objective of the assessment was to determine the labor requirements, level of effort, and verification and validation tasks necessary to ensure that the Trilogy acquisition meets the requirements of FBI users into the future within the established schedule and budget
DOJ (2005)	FBI management did not act in a timely manner on a number of critical internal and external reports that demonstrated significant project risks.
DOJ (2005)	the Trilogy project, was the focus of several reports issued both by components within the FBI and external reviewing entities, including the OIG.
Score	Overall Evaluation
1.0	Nominal. External audits were used in a few process areas.

Exhibit 87: EA Evaluation

5. Software Project Management: The software management area addressed seven measurement objects:

a. Project Planning (PP)

Project Planning (PP): The process that established and maintained plans that defined project activities.	
Source of Empirical Evidence	Empirical Evidence
Goldstein (2005)	poor planning
US Senate (2005b)	SAIC used eight development teams working in parallel and a program staff that reached 250 full-time equivalents
GAO (2004a)	the bureau lacked an enterprise architecture—a key component in developing and modernizing systems. We found that the absence of the architecture contributed to unnecessary rework to integrate several modernization initiatives, including Trilogy
Goldstein (2005)	The company had settled on a spiral development methodology, an iterative approach to writing software. Basically, SAIC programmers would write and compile a block of code that performed a particular function, then run it to show Depew's agents what it would do
McGroddy & Lin (2004)	The development methodology for Trilogy seems to be based on a one-way non-iterative process where rigid specifications are generated in advance
Goldstein (2005)	the lack of a plan to guide hardware purchases, network deployments, and software development for the bureau
GAO (2004a)	to introduce an integrated approach to IT project planning,
Goldstein (2005)	With no detailed description of the FBI's processes and IT infrastructure as a guideline, Depew said that his team of agents began "to feel our way in the

	dark,"
FBI (2002)	The UAC portion is also going to be deployed in two phases in the accelerated plan, release one and release two.
DOJ (2005)	for completion in December 2003. The second and third deliveries, which were intended to upgrade and add additional investigative applications to the VCF, were targeted for completion in June 2004.
DOJ (2005)	SAIC had developed a plan to make the ACS web-enabled by July 2002 – 24 months earlier than scheduled – without increasing project costs.
DOJ (2005)	was to be deployed in two phases under an accelerated plan: delivery one and delivery two. A third delivery was added in March 2003.
DOJ (2005)	unrealistic scheduling of tasks
DOJ (2005)	This lack of oversight included a lack of project management plans that would include cost and schedule controls
DOJ (2005)	the scheduled completion dates for individual project components were unrealistic
Score	Overall Evaluation
0.5	The software development project performed one PP process.

Exhibit 88: PP Evaluation

b. Project Monitoring and Control (PMC)

Project Monitoring & Control (PMC): The process to provide an understanding of the project's progress so that appropriate corrective actions could be taken when the project's performance deviated significantly from the plan.	
Source of Empirical Evidence	Empirical Evidence
Goldstein (2005)	But while the Trilogy contracts were changed to reflect the aggressive new deadlines, neither the original software contract nor the modified one specified any formal criteria for the FBI to use to accept or reject the finished VCF software,
McGroddy & Lin (2004)	Clear and detailed schedules with intermediate milestones, earned-value metrics, and severe penalties for missed delivery dates and missing functionality are desperately needed
McGroddy & Lin (2004)	contract schedules lack the specificity necessary to determine whether a project is making adequate progress within schedule and budget constraints
McGroddy & Lin (2004)	earned-value metrics
FBI (2002)	FBI officials told us that the rapid procurement and deployment of Trilogy has prevented the project managers from performing earned value management
FBI (2002)	FBI officials told us that the rapid procurement and deployment of Trilogy has prevented the project managers from performing earned value management, ⁵ as promised to Congress. While FBI officials were confident they know how much money has been spent on Trilogy to date, and how much funding has been committed, they have less assurance as to whether Trilogy is on budget, over budget, or under budget
McGroddy & Lin (2004)	In the wake of the VCF prime contractor's failure to meet a critical delivery date
McGroddy & Lin (2004)	it is imperative that senior management of the FBI monitor contractor progress closely and step in when necessary to forestall difficulties seen down the road, although day-to-day involvement is not necessary
Goldstein (2005)	managers at DynCorp, which was working on the hardware (computers and network) portions of Trilogy, for copies of the two project schedules. She was told the delivery dates instead.
Goldstein (2005)	SAIC, with its programmers pecking away at its secure data center in Vienna, Va., always had a detailed schedule posted prominently in the "war room"

	there, which Higgins's team would review with SAIC periodically,
FBI (2002)	they have less assurance as to whether Trilogy is on budget, over budget, or under budget
US Senate (2005a)	When a program or project metric varies by more than 10 percent of the acceptable thresholds for cost, schedule, and performance, it will trigger closer scrutiny and remedial action
DOJ (2005)	The FBI's Project Management Office did not implement a centralized budget, accounting, and procurement structure to ensure global financial management oversight.
DOJ (2005)	If monitoring the scheduling of the project had been a priority, the FBI could have taken more timely action to effectively address Trilogy's problems.
Score	Overall Evaluation
0.5	The software development project performed one PMC process.

Exhibit 89: PMC Evaluation

c. Supplier Agreement Management (SAM)

Supplier Agreement Management (SAM): The process to manage the acquisition of products or services from suppliers for which there were formal agreements.	
Source of Empirical Evidence	Empirical Evidence
US Senate (2005a)	Although the requirements were solidified in November 2002, the contract remained a cost-plus-award-fee contract
McGroddy & Lin (2004)	contract schedules were almost totally lacking in specifications, deliverables, and commitment to checkpoints
US Senate (2005b)	Currently, the contract has a negotiated value of \$130.3 million and a funded value of \$123 million. To date, SAIC has been paid \$115.2 million
FBI (2002)	Had a more rigorous proposal selection process been in place to require sufficient documentation of the technical requirements and risks of the project, the expending of time and resources on thin-client technology and web-enablement of ACS may have been minimized
US Senate (2005a)	The contract was based on hours worked -- cost plus an award fee
McGroddy & Lin (2004)	The contracts must include performance measures for key deliverables, milestones, and service levels with penalties and escalation procedures. Contracts should ensure that the vendor suffers severe penalties if it is not meeting the performance measures, and major vendor failure should result in a penalty that allows the FBI to transition to a new vendor with little or no financial impact to the FBI
McGroddy & Lin (2004)	the FBI is unable to take managerial actions such as reprogramming amounts in excess of \$500,000 without explicit congressional approval
McGroddy & Lin (2004)	the FBI should exploit proven methodologies of contracting and contract management,
McGroddy & Lin (2004)	the FBI should exploit proven methodologies of contracting and contract management, including the use of detailed functional specifications, specific milestones, frequent contract reviews, and earned-value metrics
Hickerson (2006)	The FBI, citing lack of contracting manpower, used a Government-wide Acquisition Contract as the method of managing the grant money behind Trilogy. The GAC is a model that removes management control from the government agency. In this case, the FBI had to relinquish control and allow SAIC to run the program
McGroddy & Lin (2004)	the FBI's contract management process is inadequate
FBI (2002)	the user application component of Trilogy, recognized by FBI officials as the most important aspect of the project in terms of improving agent performance, is at high risk of not being completed within the funding levels appropriated by

	Congress
US Senate (2005b)	SAIC was tasked to in February 2002 to develop the replacement for the legacy systems using the original contract. The SAIC UAC contract was restructured to incorporate an aggressive development plan first conceived in February 2002. This became the electronic Virtual Case File (VCF) contract
US Senate (2005b)	At the time of award in June 2001, the contract scope for SAIC called for development of a web front-end to the existing legacy applications used to manage case information
FBI (2002)	In January 2002, Congress supplemented Trilogy's FY 2002 budget with \$78 million ⁴ to expedite the deployment of all three components. This supplemental appropriation increased the total funding of Trilogy from approximately \$380 million to \$458 million.
US Senate (2005b)	In June 2001, SAIC was competitively awarded a cost-plus-award fee developmental contract for the Trilogy User Application Component (UAC)
Goldstein (2005)	In May and June 2001, the bureau awarded Trilogy contracts to two major U.S. government contractors: DynCorp, of Reston, Va., for the hardware and network projects, and to SAIC for software.
DOJ (2005)	a primary reason for the schedule and cost problems associated with Trilogy was weak statements of work in the contracts
DOJ (2005)	According to FBI and Department officials, the Department required the FBI to use two contractors for Trilogy because the Department considered the project too large for a single contractor to manage.
DOJ (2005)	Because the FBI wanted to award the Trilogy contracts quickly and did not have clearly defined requirements, it used the cost-plus-award-fee contract vehicle
DOJ (2005)	contracting weaknesses
DOJ (2005)	did not provide for penalties if the milestones were not met
DOJ (2005)	did not require specific completion milestones
DOJ (2005)	did not include critical decision review points
DOJ (2005)	had the Trilogy contracts included fully established requirements and firm completion milestones, the adverse effects of such changes could have been mitigated.
DOJ (2005)	Under cost-plus-award-fee contracts, contractors are only required to make their best effort to complete the project
DOJ (2005)	weak government contract management was more of the problem with Trilogy than the terms of the contracts.
DOJ (2005)	The FBI's decision to use a cost-plus-award-fee contract to develop Trilogy placed it at a significant disadvantage because the contract did not establish firm milestones or prescribe penalties for a contractor that missed deadlines or delivered an unacceptable product.
DOJ (2005)	The Trilogy contract was offered as a cost-plus-award -fee on labor whereby the contractors' costs are reimbursed and fees can be awarded to the contractor.
DOJ (2005)	the FBI decided to use the General Services Administration's (GSA) Millennium contracting process. The GSA's Federal Technologies Services' Federal Systems Integration and Management (FEDSIM) Center provides IT contracting services for its federal agency clients. FEDSIM's role is to oversee competing contracts, and to award and manage existing contracts. In other words, FEDSIM acts as the contracting office.
Score	Overall Evaluation
0.0	The software development project did not perform the SAM process.

Exhibit 90: SAM Evaluation

d. Risk Management (RSKM)

Supplier Agreement Management (SAM): The process to manage the acquisition of products or services from suppliers for which there were formal agreements.	
Source of Empirical Evidence	Empirical Evidence
McGroddy & Lin (2004)	First, foremost, and most critical in light of the impending rollout of the Virtual Case File (VCF) application, the FBI should not proceed with deployment of the VCF until it has a validated contingency plan for reverting completely or partially to the Automated Case Support (ACS) system
McGroddy & Lin (2004)	the impending VCF system rollout, is that the FBI not proceed with deployment of the VCF until it has a validated contingency plan for reverting completely or partially to the ACS, if necessary, and clear and measurable criteria to determine when the ACS can safely be turned off
FBI (2002)	The three internal risk-assessments on Trilogy found significant risks associated with the management of the project
Score	Overall Evaluation
0.0	The software development project did not perform the RSKM process.

Exhibit 91: RSKM Evaluation

e. Integrated Project Management (IPM)

Integrated Project Management (IPM): The process to establish and manage the project and the involvement of the relevant stakeholders according to an integrated and defined process.	
Source of Empirical Evidence	Empirical Evidence
GAO (2004a)	as of August 2004, Trilogy has experienced a delay of at least 21 months and a cost increase of \$201 million. According to the CIO, the project's added time and cost were due in large part to requirements development and management process weaknesses
Goldstein (2005)	Computer Sciences Corp didn't have its hardware and network in place, so SAIC couldn't adequately test the VCF, crucial for a successful flash cutover.
Goldstein (2005)	In mid-April 2002, Higgins gave DynCorp a week to deliver a detailed schedule. After she got it, she pulled the project teams from the FBI and DynCorp into a meeting and went through the document. Shortly after that, Higgins broke the news to the director: the computers and networks would not be delivered in July of that year as had been scheduled.
DOJ (2005)	The problems involved with the scheduling of the project would have been more apparent to the FBI had proper project integration efforts taken place. A professional project integrator could have coordinated the scheduling of the infrastructure with the VCF implementation.
McGroddy & Lin (2004)	Perhaps the most important-and commendable-development in the VCF effort is the appointment of a very experienced and computer-savvy FBI special agent as program manager who has played a strong role in driving the design from user requirements.
GAO (2005b)	project management deficiencies
GAO (2004a)	project management deficiencies
GAO (2004a)	Reviews of the bureau's centerpiece systems modernization project, Trilogy, have identified management weaknesses as the cause for cost, schedule, and performance shortfalls that have been experienced by the project
GAO (2005b)	Reviews of this project identified management weaknesses as the cause for its cost, schedule, and performance shortfalls
McGroddy & Lin (2004)	Senior-level contract managers, experienced and empowered, should be assigned for the duration of the project, and should provide periodic actionable

	status information to FBI senior management
GAO (2004a)	the bureau's weaknesses in IT management controls, such as investment management and enterprise architecture, contributed to Trilogy schedule delays of at least 21 months and cost increases of about \$120 million
FBI (2002)	the lack of critical IT investment management processes contributed to missed milestones and led to uncertainties about cost, schedule, and technical goals
GAO (2003)	the modernization program management office told us that the office recently assumed responsibility for managing three system modernization initiatives ¹⁹ and found that they will require rework in order for them to be integrated
GAO (2004b)	These Trilogy shortfalls in meeting cost and schedule commitments can be in part attributed to the absence of the kind of IT management controls discussed earlier. Specifically, in its study of the FBI's investment management processes which included a case study of Trilogy, the Inspector General cited the lack of an enterprise architecture and mature IT investment management processes as the cause for missed Trilogy milestones and uncertainties associated with the remaining portions of the project
Harris (2005)	an inattentive and frequently changing management staff.
Goldstein (2005)	By April, 22 251 computer workstations, 3408 printers, 1463 scanners, 475 servers, and new local and wide area networks would all be up and running, 22 months later than the accelerated schedule called for.
FBI (2002)	The initial Virtual Case File release will migrate data from the current ACS and IntelPlus to the Virtual Case File. The Virtual Case File will replace ACS and serve as the backbone of the FBI's information systems, replacing the FBI's paper files with electronic case files that include multi-media capabilities. The first release of Virtual Case File has a targeted completion date of December 2003. This release is intended to allow different types of users, such as agents, analysts, and supervisors, to access information from their desktop computers that is specific to their individual needs
DOJ (2005)	FBI management did not exercise adequate control over the Trilogy project and its evolution in the early years of the project.
DOJ (2005)	the FBI used a contractor, Mitretek Systems, to assist the FBI with a wide array of tasks, including program and contract management, system engineering and architecture, fiscal and budgetary oversight, communications, testing, configuration management, cost estimating, acquisition and source selection, requirements definition, training, database management, security certification and accreditation, and web development.
DOJ (2005)	the contractor for the User Applications Component (UAC), SAIC, used a scheduling tool for the development of the VCF with which the FBI was unfamiliar. As a result, the FBI was unable to determine if the assumptions within the schedule were reasonable and whether the implications on the schedule were adequately reflected.
DOJ (2005)	project management was not consistently followed by IT project managers
DOJ (2005)	the FBI did not hire a professional project integrator to manage contractor interfaces and take responsibility for the overall integrity of the final product until the end of 2003.
DOJ (2005)	In its July 6, 2001, Quarterly Congressional Status Report the FBI stated that the IPC/TNC infrastructure could be completed in June 2003, nearly one year ahead of schedule, with a two-phase implementation plan. The FBI also wanted to accelerate deployment of the urgently needed user applications component, which was scheduled to take three years.
DOJ (2005)	The CIO brought in a Project Management Executive to manage the Trilogy project in place of the Deputy Assistant Director in the Information Resources Division.
DOJ (2005)	lack of adequate project integration
Score	Overall Evaluation

0.5	The software development project performed one IPM process.
-----	---

Exhibit 92: IPM Evaluation

f. Decision Analysis and Resolution (DAR)

Decision Analysis & Resolution (DAR): The process which analyzed possible decisions using a formal evaluation process that evaluated identified alternatives against established criteria.	
Source of Empirical Evidence	Empirical Evidence
None	There is no discussion of DAR in any of the literature. This is an advanced behavior and would not be expected to be performed on a project with a low level of maturity.
Interview	The interview subject stated that this process was not directly addressed.
Score	Overall Evaluation
0.0	The software development project did not perform the DAR process.

Exhibit 93: DAR Evaluation

g. Quantitative Project Management (QPM)

Quantitative Project Management (QPM): The process which quantitatively managed the project's defined processes to achieve the projects established quality and process-performance objectives.	
Source of Empirical Evidence	Empirical Evidence
None	There is no discussion of QPM in any of the literature. This is an advanced behavior and would not be expected to be performed on a project with a low level of maturity.
Interview	The interview subject stated that this process was not directly addressed.
Score	Overall Evaluation
0.0	The software development project did not perform the QPM process.

Exhibit 94: QPM Evaluation

The overall score for the functions element was 7.5 of 50 possible points.

Structure Element

The structure element of the FSE framework had three areas that evaluated 20 measurement objects related to the structural ability of the project to produce software. The areas address the social system, the cybernetic controls and the technical system worth a total of 25 points. Each area will be evaluated separately.

1. Social System: The social system area addressed six measurement objects:

a. Analyst Capability (ACAP)

Analyst Capability (ACAP): (1) The analysis and design ability, efficiency, and thoroughness, of the analyst team (those who worked on requirements, high-level design, and detailed design) that worked on the software development project and (2) the analyst team's ability to communicate and cooperate during the software development project.	
Source of Empirical Evidence	Empirical Evidence
DOJ (2005)	The contractors performing the review stated that the VCF design was adversely affected by a lack of engineering expertise on the project.
Score	Overall Evaluation
0.2	Low: 15 th -35 th percentile.

Exhibit 95: ACAP Evaluation

b. Programmer Capability (PCAP)

Programmer Capability (PCAP): (1) The analysis and design ability, efficiency, and thoroughness, and the ability to communicate and cooperate of the programmer team (those who implement processes and functions through software code) that worked on the software development project and (2) the programmer team's ability to communicate and cooperate during the software development project.	
Source of Empirical Evidence	Empirical Evidence
None	There is no discussion of PCAP in any of the literature. A nominal value for PCAP is assumed.
Score	Overall Evaluation
0.6	Nominal: 55 th -75 th percentile.

Exhibit 96: PCAP Evaluation

c. Personnel Continuity (PCON)

Personnel Continuity (PCON): The software development project's annual personnel turnover.	
Source of Empirical Evidence	Empirical Evidence
GAO (2005a)	frequent turnover of FBI IT managers
GAO (2004a)	frequent turnover of key personnel
DOJ (2005)	Turnover in key positions has inhibited the FBI's ability to manage and oversee the Trilogy project.
McGroddy & Lin (2004)	frequent turnover among key FBI staff, make it unsurprising that Trilogy is significantly behind schedule and over budget
US Senate (2005b)	Since November 2001, there have been 19 Government management personnel changes that had a direct and significant impact on the management of this project (11 FBI Changes and 8 FEDSIM Changes). This lack of continuity among key Government managers contributed to the problems of ensuring the effective and timely implementation of this system
Goldstein (2005)	The real killers, he said, were "significant management turbulence" at the FBI
Score	Overall Evaluation
0.2	Very Low: >48% per year

Exhibit 97: PCON Evaluation

d. Applications Experience (APEX)

Applications Experience (APEX): The level of applications experience of the team that developed the software system? [Note: This is defined in terms of the development team's equivalent years of experience with this type of application.]	
Source of Empirical Evidence	Empirical Evidence
None	There is no discussion of APEX in any of the literature. A nominal value for APEX is assumed.
Score	Overall Evaluation
0.6	Nominal: 6-12 months

Exhibit 98: APEX Evaluation

e. Platform Experience (PLEX)

Platform Experience (PLEX): The level of platform experience of the team that developed the software system? [Note: This is defined in terms of the development team's equivalent years of experience with the type of platform (i.e. graphical user interface, database, networking, middleware, etc.).]	
Source of Empirical Evidence	Empirical Evidence
None	There is no discussion of PLEX in any of the literature. A nominal value for PLEX is assumed.
Score	Overall Evaluation
0.6	Nominal: 6-12 months

Exhibit 99: PLEX Evaluation

f. Language and Tool Experience (LTEX)

Platform Experience (PLEX): The level of platform experience of the team that developed the software system? [Note: This is defined in terms of the development team's equivalent years of experience with the type of platform (i.e. graphical user interface, database, networking, middleware, etc.).]	
Source of Empirical Evidence	Empirical Evidence
None	There is no discussion of LTEX in any of the literature. A nominal value for LTEX is assumed.
Score	Overall Evaluation
0.6	Nominal: 6-12 months

Exhibit 100: LTEX Evaluation

2. Cybernetic Control: The cybernetic controls area addressed five measurement objects:

a. Control (CTRL)

Control (CTRL): The operational measures the project used most commonly to control the performance of the software development project.	
Source of Empirical Evidence	Empirical Evidence
McGroddy & Lin (2004)	Clear and detailed schedules with intermediate milestones, earned-value metrics, and severe penalties for missed delivery dates and missing

	functionality are desperately needed ¹
DOJ (2005)	If monitoring the scheduling of the project had been a priority, the FBI could have taken more timely action to effectively address Trilogy's problems. ¹
Goldstein (2005)	managers at DynCorp, which was working on the hardware (computers and network) portions of Trilogy, for copies of the two project schedules. She was told the delivery dates instead. ¹
Goldstein (2005)	SAIC, with its programmers pecking away at its secure data center in Vienna, Va., always had a detailed schedule posted prominently in the "war room" there, which Higgins's team would review with SAIC periodically. ¹
Score	Overall Evaluation
1.0	Cost & Schedule: Cost and schedule data are used to control the performance of the software development project.

Exhibit 101: CTRL Evaluation

¹Note: These are duplicate nodes also reported for the performance monitoring and control (PMC) process.

b. Policy (POL)

Policy (POL): The operational controls and strategic measures used to ensure that the software development project remained viable.	
Source of Empirical Evidence	Empirical Evidence
McGroddy & Lin (2004)	an architecture is necessary to provide a strategic view of its mission and operational needs, and would begin with a detailed characterization of the bureau's goals, tasks, strategies, and key operational processes. This view links operational objectives and processes to IT strategy and will allow the FBI to specify how investment is tied to the achievement of operational objectives
GAO (2004a)	For example, in March 2004, the Department of Justice Inspector General testified ¹⁷ that the lack of an architecture was a contributing factor to the continuing cost and schedule shortfalls being experienced by the bureau on its Trilogy investigative case management system
GAO (2005a)	In addition, it found that modernization initiatives, such as Trilogy, were not closely linked to a coherent view of the bureau's mission and operational needs
Goldstein (2005)	SAIC's bid on the original contract, and each subsequently revised cost estimate, was based on there being "minimal, minor changes" to the program once a baseline set of requirements had been agreed on.
McGroddy & Lin (2004)	Second, the success of the FBI's information technology efforts will require the development of a close linkage between IT and a coherent view of the bureau's mission and operational needs
McGroddy & Lin (2004)	the FBI's top leadership, including the director, must make the creation and communication of a complete enterprise architecture a top priority
GAO (2005a)	the National Research Council reported ¹⁸ in May 2004 that while the bureau had made significant progress in its IT systems modernization program, the FBI was not on the path to success, in part, because it had not yet developed an EA
FBI (2002)	the philosophy employed in implementing Trilogy was "to get 80% of what is needed into the field now rather than 97% later
GAO (2004b)	According to FBI estimates, the bureau manages hundreds of systems and associated networks and databases at an average annual cost of about \$800 million. In addition, the bureau plans to invest about \$255 million and \$286 million in fiscal years 2004 and 2005, respectively, in IT services and systems, such as the Trilogy project. Trilogy is the bureau's centerpiece project to (1) replace its system infrastructure (e.g., wide area network) and (2) consolidate

	and modernize key investigative case management applications. The goals of Trilogy include speeding the transmission of data, linking multiple databases for quick searching, and improving operational efficiency by replacing paper with electronic files
Goldstein (2005)	each division had its own IT budget and systems. And because divisions had the freedom and money to develop their own software, the FBI now has 40 to 50 different investigative databases and applications, many duplicating the functions and information found in others.
DOJ (2005)	policies and procedures were not developed for management oversight of IT projects
DOJ (2005)	IT investment management weaknesses,
DOJ (2005)	Without a complete Enterprise Architecture, the FBI's systems are not defined. As a result, in the Trilogy project the FBI needed to conduct reverse engineering to identify existing IT capabilities before developing the infrastructure and user applications requirements
GAO (2004b)	Nevertheless, the bureau's longstanding approach to managing IT is not fully consistent with leading practices, as has been previously reported by us and others. The effect of this, for example, can be seen in the cost and schedule shortfalls being experienced on Trilogy
Score	Overall Evaluation
0.0	No coordination: No coordination between the intelligence or policy functions existed..

Exhibit 102: POL Evaluation

c. Intelligence (INT)

Intelligence (INT): The external intelligence measures used as part of the planning process for the software development project.	
Source of Empirical Evidence	Empirical Evidence
None	There is no discussion of INT in any of the literature. Because neither the FBI (customer) nor SAIC (developer) could fix the requirements for the system a CPAF contract was used. The CPAF contract assumes inadequate requirements definition and pays for all developer costs.
Score	Overall Evaluation
0.0	No intelligence measures were used.

Exhibit 103: INT Evaluation

d. Communications Channels (CC)

Communications Channels (CC): How information was communicated within the software development project.	
Source of Empirical Evidence	Empirical Evidence
Goldstein (2005)	bad communication
Score	Overall Evaluation
0.0	Informal communications channels.

Exhibit 104: CC Evaluation

e. Attenuation (ATT)

Environmental Attenuation (ATT): How the overall variety presented to the project was controlled by using attenuation and/or amplification methods.	
Source of Empirical Evidence	Empirical Evidence
None	There is no formal discussion of ATT in any of the literature. However, the literature discusses the inability of the project to control the growth of user requested requirements. The requirements emanated from the environment, in this case the system's potential user's, and were not controlled.
Score	Overall Evaluation
0.0	No attenuation or amplification methods were in place.

Exhibit 105: ATT Evaluation3. Technical System: The technical system area addressed nine measurement objects:

a. Software Reliability (RELY)

Software Reliability (RELY): The design of the software ensured that in the event of complete system failure the users would consider this to be:	
Source of Empirical Evidence	Empirical Evidence
Alfonsi (2005)	"immediately develop plans that address recovery of data and functionality in the event that essential technology services come under denial-of-service attacks" — for example, viruses and pervasively replicated software bugs.
US Senate (2005b)	high system availability
McGroddy & Lin (2004)	Seemingly inadequate contingency plans for operating under attack. A basic principle of managing a critical operational network is that plans for maintaining operation in the face of a compromised element must be made in advance
McGroddy & Lin (2004)	The costs of maintaining a fallback plan and a supporting infrastructure are small compared to the operational costs associated with large-scale VCF problems
McGroddy & Lin (2004)	The Trilogy IT modernization put into place a single operating system environment, and the security vulnerabilities of an operating system mono-culture are well known
McGroddy & Lin (2004)	there is some risk that the problems will be so severe as to prevent the effective use of the entire system for some period of time. Accordingly, there must be backup and contingency plans in place that anticipate a wide range of failure conditions
Score	Overall Evaluation
0.4	High. High financial loss. (In this case it is the loss of capability)

Exhibit 106: RELY Evaluation

b. Database Size (DATA)

Database Size (DATA): The ratio of bytes in the system test database (D) to the number of bytes (source lines of code) in the application program (P).	
Source of Empirical Evidence	Empirical Evidence
None	There is no formal discussion of DATA in any of the literature. A nominal value for DATA is assumed.

Score	Overall Evaluation
0.8	Nominal: $10 < D/P < 100$.

Exhibit 107: DATA Evaluation

c. Development for Reuse (RUSE)

Development for Reuse (RUSE): The decision to utilize reusable software components as part of the software system design required the use of components.	
Source of Empirical Evidence	Empirical Evidence
Goldstein (2005)	Patton also claimed that SAIC was determined to write much of the VCF from scratch.
Score	Overall Evaluation
1.0	Nominal: Re-use was not an element of the design.

Exhibit 108: RUSE Evaluation

d. Life Cycle Documentation (DOCU)

Life Cycle Documentation (DOCU): How the design of the software system affected the life-cycle documentation needs of the software system.	
Source of Empirical Evidence	Empirical Evidence
FBI (2002)	Because the UAC portion of Trilogy is focused on making significant changes to, or possibly complete replacements of, five of the FBI's investigative systems, having documentation of the exact configuration of these systems is critical to designing the requirements for UAC
US Senate (2005b)	In addition to these capabilities, SAIC performed substantial analysis and engineering efforts to document the complex and largely undocumented legacy environment that has evolved over the years
Score	Overall Evaluation
0.8	Nominal: The design has created some life-cycle needs.

Exhibit 109: DOCU Evaluation

e. Execution Time Constraint (TIME)

Execution Time Constraint (TIME): The percentage of the customer specified system response time was used in the design of the software system.	
Source of Empirical Evidence	Empirical Evidence
US Senate (2005b)	providing a 3-second response to users
Score	Overall Evaluation
0.4	Extra-High: 90-95%

Exhibit 110: TIME Evaluation

f. Main Storage Constraint (STOR)

Main Storage Constraint (STOR): The percentage of the customer specified storage was used in the design of the software system.	
Source of Empirical Evidence	Empirical Evidence
None	There is no discussion of STOR in any of the literature. A nominal value for

	STOR is assumed.
Score	Overall Evaluation
1.0	Nominal: < 50%

Exhibit 111: STOR Evaluation

g. Platform Volatility (PVOL)

Platform Volatility (PVOL): How often changes to the software that made up the system are expected to be required.	
Source of Empirical Evidence	Empirical Evidence
None	There is no discussion of PVOL in any of the literature. However, based on the high level of user defined changes requests during development in is logical to assume that the user's would expect major changes every 2 months, and minor changes every week.
Score	Overall Evaluation
0.6	High: Major changes every 2 months, minor changes every week.

Exhibit 112: PVOL Evaluation

h. Software System Complexity (COMP)

System Complexity (COMP): The complexity of the software that the development project was working on.	
Source of Empirical Evidence	Empirical Evidence
Hickerson (2006)	a process that was supposed to take hours would now have to be completed in three seconds. The VCF contract became a more ambitious system that would handle millions of case files in a variety of formats with a three-second response time to specific queries
McGroddy & Lin (2004)	bureau-wide technology deployment necessarily entails a set of systems and data that can be accessed easily across the geographic reach of the FBI's missions. (The FBI encompasses 56 field offices in major cities in the United States, approximately 400 resident agencies (i.e., satellite offices in smaller cities and towns), and foreign posts in 52 nations.
Alfonsi (2005)	Furthermore, any new program will have to strike a tenuous balance between allowing for crucial information sharing and keeping classified information top secret. "We just don't know how they will achieve this," our source says. "But the report's recommendation of having two separate systems (one with shared access and one requiring special security clearance) seems to be the most logical way to go."
Goldstein (2005)	Patton's descriptions of the 800-plus pages of requirements show the project careening off the rails right from the beginning. For starters, this bloated document violated the first rule of software planning: keep it simple.
Goldstein (2005)	SAIC agreed to deliver the initial version of the VCF in December 2003 instead of June 2004. SAIC and the FBI were now committed to creating an entirely new case management system in 22 months, which would replace ACS in one fell swoop, using a risky maneuver known in the IT business as a flash cutover.
Goldstein (2005)	the FBI wanted a "page crumb" capability added to all the screens. Also known as "bread crumbs," a name inspired by the Hansel and Gretel fairy tale, this navigation device gives users a list of URLs identifying the path taken through the VCF to arrive at the current screen. This new capability not only added more complexity

US Senate (2005b)	VCF was a large and complex enterprise-level undertaking. There are no other criminal investigative management systems of this scale in the world
US Senate (2005a)	We underestimated the complexity
Score	Overall Evaluation
0.6	System: Collection of subsystems with multiple functions.

Exhibit 113: COMP Evaluation

i. Technology Application (TECH)

Technology Application (TECH): The software and/or hardware technology in place at the start of the development project.	
Source of Empirical Evidence	Empirical Evidence
Goldstein (2005)	13 000 computers could not run modern software. Most of the 400 resident agency offices were connected to the FBI intranet with links about the speed of a 56-kilobit-per-second modem
Kumagai (2003)	Lastly, there are the many unresolved technical questions: is it really possible to build a system that can precisely identify a crime's precursors, when the would-be perpetrators are doing their utmost to be untraceable and unpredictable?
Goldstein (2005)	Many of the bureau's network components were no longer manufactured or supported
US Senate (2005a)	the pace of technology has overtaken the development of unique software applications for the FBI, and we may turn to Commercial Off-The-Shelf, or COTS-based, products
Score	Overall Evaluation
0.8	Medium-Tech: Some new technology.

Exhibit 114: TECH Evaluation

The overall score for the structure element was 10.2 of 25 possible points.

Environment Element

The environment element of the FSE framework had three areas that evaluated 14 measurement objects related to the environmental factors affecting the ability of the project to produce software. The areas address the external controls, the resources, and the stakeholders worth a total of 25 points. Each area will be evaluated separately.

1. External Controls: The external controls area addressed two measurement objects:

a. Laws and Regulations (LAW)

Laws and Regulations (LAW): The extent to which government laws and regulations were addressed by the project and/or parent organization.	
Source of Empirical Evidence	Empirical Evidence
None	There is no mention of LAW in the literature. However, SAIC is one of the

	largest IT companies in the world and has a staff that follows federal legislation and works with the legislative and executive branches of the federal government.
Score	Overall Evaluation
0.6	The project and/or the parent organization have limited involvement in influencing laws and regulations.

Exhibit 115: LAW Evaluation

b. Industry Standards (STD)

Industry Standards (STD): The extent to which industry standards were addressed by the project and/or parent organization.	
Source of Empirical Evidence	Empirical Evidence
None	There is no mention of STD in the literature. However, SAIC is one of the largest IT companies in the world and has a number of employees actively involved on software committees with the IEEE and ISO/IEC.
Score	Overall Evaluation
0.8	The project and/or the parent organization are closely involved in influencing industry standards.

Exhibit 116: STD Evaluation

2. Resources: The resources area addressed six measurement objects:

a. Manpower: Labor Availability (MAN)

Manpower Labor Availability (MAN): The availability of labor skills required by the project.	
Source of Empirical Evidence	Empirical Evidence
Goldstein (2005)	By August 2002, it had around 200 programmers on the job.
Score	Overall Evaluation
1.5	Above Average: Most of the required labor skills were present in the local workforce. Only a few key labor skills were required to be imported. The project was developed locally and used some limited remote development.

Exhibit 117: MAN Evaluation

b. Critical Material Availability (MAT)

Critical Material Availability (MAT): The availability of critical materials (typically hardware and software components) required by the project.	
Source of Empirical Evidence	Empirical Evidence
None	There is no mention of MAT in the literature. Because the project was using no emerging or developing technology, it is logical to assume that critical materials are readily available.
Score	Overall Evaluation
2.0	Available: Critical materials are readily available.

Exhibit 118: MAT Evaluation

c. Money: Capital Investment (CAP)

Money: Capital Investment (CAP): The availability of capital required to fund indirect activities (i.e. process improvement, skill development, training, etc.) on the project.	
Source of Empirical Evidence	Empirical Evidence
None	There is no mention of CAP in the literature. However, it is logical to assume that funds for capital projects would come from the project management reserve.
Score	Overall Evaluation
1.0	Reserve: Capital funds for the project must come from the project management reserve.

Exhibit 119: CAP Evaluation

d. Schedule Pace (PACE)

Schedule Pace (PACE): The pace required to achieve the schedule for the project.	
Source of Empirical Evidence	Empirical Evidence
Goldstein (2005)	People forget the urgency that we were under and our customer was under, said SAIC's Reynolds.
Goldstein (2005)	changes and fixes continued to strangle the VCF in the crib. Many of the changes had to be made by all eight of SAIC's development teams. Arnold Punaro, SAIC executive vice president and general manager, admitted in a posting on the company's Web site that in the rush to get the program finished by December, SAIC didn't ensure that all of its programmers were making the changes the same way.
McGroddy & Lin (2004)	in the interests of rapid deployment, the current VCF schedule appears to give little consideration to testing and presumes success at every stage-a highly risky approach
Goldstein (2005)	overly ambitious schedules
US Senate (2005b)	SAIC was asked to devise an approach to deliver VCF in record time-on an even more aggressive schedule. The new challenge was to define, develop, and deploy a bureau-wide enterprise-level case management system in just 22 months
McGroddy & Lin (2004)	the bureau-wide rollout of this application is months delayed from its originally scheduled deployment in December 2003
McGroddy & Lin (2004)	The project schedule for completion of Trilogy as it was represented to the committee in October 2003 appears to leave inadequate time for testing. The committee was shown briefing charts indicating that the FBI allocated less than 10 percent of its schedule for testing, and under schedule pressure the contractor was trimming that amount even further
DOJ (2005)	The September 11 attacks provided even greater impetus to completing Trilogy, and the FBI continued to explore options to accelerate deployment of all three Trilogy components.
Score	Overall Evaluation
0.0	Crisis: Immediate delivery of the software is necessary.

Exhibit 120: PACE Evaluation

e. Formal Methods (METH)

Formal Methods (METH): The adoption and implementation of formal methods on the project.	
Source of Empirical Evidence	Empirical Evidence
US Senate (2005b)	With multiple teams working on vertical slices of the system at breakneck speed, SAIC did not adequately enforce coding standards across the teams and this resulted in less than uniform code
GAO (2005b)	the bureau's long-standing approach to managing IT has not always been fully consistent with leading practices. The effects of this approach can be seen in, for example, the cost and schedule shortfalls experienced on a key infrastructure and applications modernization program
Score	Overall Evaluation
0.5	Limited: Formal standards are those required by contract.

Exhibit 121: METH Evaluation

e. Information: External Communications (COMM)

Information: External Communications (COMM): The methods used to control the flow of information between the project and the external environment.	
Source of Empirical Evidence	Empirical Evidence
None	There is no mention of COMM in the literature. It is logical to assume that information controls were limited to formal correspondence.
Score	Overall Evaluation
1.0	Limited: Information controls are limited to formal correspondence.

Exhibit 122: COMM Evaluation3. Stakeholders: The stakeholder area addressed six measurement objects:

a. Owners/Shareholder Boards (OWN)

Owners/Shareholders (OWN): The formal level of involvement of the owner(s) of the company or the corporate board of directors with the development project.	
Source of Empirical Evidence	Empirical Evidence
Company literature	SAIC is an employee owned company. An Executive Vice President (Arnold Punaro) and a Vice President (Brice Zimmerman and Rick Reynolds) were assigned responsibility for the project
Score	Overall Evaluation
1.0	Informational: Owners or shareholder boards were informed about overall project performance (i.e. cost, schedule and customer satisfaction).

Exhibit 123: OWN Evaluation

b. External Management (MGT)

External Management (MGT): The formal level of involvement of external management with the development project.	
Source of Empirical Evidence	Empirical Evidence
Kumagai (2003)	After becoming the U.S. attorney for San Francisco, Mueller led an overhaul of the office's computer system for tracking cases; the new program, called Alcatraz, is now used by all U.S. attorneys. Upon arriving at the FBI, Mueller asked that Microsoft Office be installed on his desktop. "They told him, 'We can put it on there, but it won't be compatible with anything else in the FBI,' " says Kessler, "He hit the roof."
Goldstein (2005)	Azmi, unlike the previous three CIOs, inserted himself into the day-to-day operations of the IOC project. All through the second half of 2004, he met with his project manager every morning at 8:15. Every night before 10 p.m., the project manager would issue a status report indicating what milestones had been hit, identifying risks, and suggesting actions to be taken to avoid mistakes and delays.
Goldstein (2005)	Chiaradio and Depew met with Dies. They convinced him, and later the director himself, that the bureau needed an entirely new database, graphical user interface, and applications, which would let agents search across various investigations to find relationships to their own cases.
Goldstein (2005)	Depew joined a team of seven that assessed the Web interface SAIC was designing for the ACS system.
US Senate (2005a)	FBI management did not exercise adequate control over the Trilogy project and its evolution in the early years of the project
Goldstein (2005)	former IBM executive Bob E. Dies, who became assistant director in charge of the FBI Information Resources Division on 17 July 2000. He was the first of five officials who, over the next four years, would struggle to lead the FBI's sprawling and antiquated information systems and get the VCF project under way.
Goldstein (2005)	In the Summer of 2002, turmoil roiled the FBI's IT management. In May, Bob Dies, the CIO who had launched Trilogy, left the bureau, turning over his duties to Mark Tanner, who held the position of acting CIO for just three months, until July 2002. He stepped aside for Darwin John, former CIO for the Mormon Church. Chiaradio, who declined to be interviewed for this article, left for a lucrative job in the private sector with BearingPoint Inc., a global consultancy in McLean, Va., and was replaced by W. Wilson Lowery Jr. Within a year, Lowery would replace John.
GAO (2004a)	Moreover, the National Research Council reported ¹⁹ in May 2004 that the bureau was experiencing significant challenges in developing and implementing Trilogy. For example, the council found that the bureau did not have a permanent CIO with the technical knowledge to provide the strong direction needed for the Trilogy program
Goldstein (2005)	On SAIC's side, Rick Reynolds assumed executive oversight on the project from Brice Zimmerman. Reynolds replaced VCF project manager Pat Boyle with Charlie Kanewske. (SAIC declined repeated requests to interview them.) Depew, like other FBI officials, had only good things to say about Kanewske.
McGroddy & Lin (2004)	the enterprise architecture be created by a combined effort involving both senior operational management and key technologists. The CIO plays a key role as the facilitator of the process
Hickerson (2006)	the FBI, lacking a sound project management structure at the time, had pushed most of the reporting on the progress of the VCF down to the lower levels of management
McGroddy & Lin (2004)	the FBI's senior leadership is insufficiently engaged in the development of the

	enterprise architecture
McGroddy & Lin (2004)	The senior management of the FBI has a substantive and direct role to play in the FBI's IT modernization efforts. This role either has not been understood or it has been given a lower priority based on the perception of more immediate operational priorities
US Senate (2005a)	We also experienced a high turnover in Trilogy program managers and Chief Information Officers
FBI (2002)	Without effective oversight of IT projects, FBI officials do not have adequate assurance that IT projects are being developed on schedule and within established budgets
DOJ (2005)	lack of management continuity and oversight
Score	Overall Evaluation
1.0	Informational: External management was informed about overall project performance (i.e. cost, schedule and customer satisfaction).

Exhibit 124: MGT Evaluation

c. Customers (CUST)

Customers (CUST): The formal level of involvement of the customer (those that contract and pay for the project and as such are differentiated from users) with the progress of the development project.	
Source of Empirical Evidence	Empirical Evidence
Goldstein (2005)	former IBM executive Bob E. Dies, who became assistant director in charge of the FBI Information Resources Division on 17 July 2000. He was the first of five officials who, over the next four years, would struggle to lead the FBI's sprawling and antiquated information systems and get the VCF project under way.
Goldstein (2005)	In the Summer of 2002, turmoil roiled the FBI's IT management. In May, Bob Dies, the CIO who had launched Trilogy, left the bureau, turning over his duties to Mark Tanner, who held the position of acting CIO for just three months, until July 2002. He stepped aside for Darwin John, former CIO for the Mormon Church. Chiaradio, who declined to be interviewed for this article, left for a lucrative job in the private sector with BearingPoint Inc., a global consultancy in McLean, Va., and was replaced by W. Wilson Lowery Jr. Within a year, Lowery would replace John.
DOJ (2005)	15 different key IT managers have been involved with the Trilogy project, including 5 CIOs or Acting CIOs and 10 individuals serving as project managers for various aspects of Trilogy. This lack of continuity among IT managers contributed to the lack of effective and timely implementation of the Trilogy project.
Score	Overall Evaluation
1.5	Managed: The customer required periodic face-to-face reviews of project progress and costs.

Exhibit 125: CUST Evaluation

d. Suppliers (SUP)

Suppliers (SUP): The involvement of those who supplied products and/or services to your development project.	
Source of Empirical Evidence	Empirical Evidence
None	There is no mention of SUP in the literature. Because of the problems with the cost and schedule mention in both the PMC and IPM areas it is logical to assume that sub-contractor's would have no more insight than the overall project.
Score	Overall Evaluation
0.0	None: The supplier had no knowledge with respect to his involvement with the project (i.e. fee and delivery date).

Exhibit 126: SUP Evaluation

e. Users (USER)

Users (USER): The involvement of the end user's of the software being developed by your development project.	
Source of Empirical Evidence	Empirical Evidence
Goldstein (2005)	[T]he continued delays in developing the VCF affect the FBI's ability to carry out its critical missions.
Goldstein (2005)	Depew's team also called in people from across the FBI: a dozen in the first few weeks; 40 by the end of November. These "subject matter experts" explained how their divisions or units functioned internally and with the rest of the bureau
McGroddy & Lin (2004)	the advantage of a small-scale prototype is that it can be iteratively developed with strong user feedback and involvement, thus increasing the chances that what is ultimately delivered to the end users meets their needs
Hickerson (2006)	The FBI could not keep up with communicating the expanding requirements for VCF to SAIC developers, and SAIC were just filling in the blanks and making too many key development decisions for themselves
McGroddy & Lin (2004)	the FBI does not appear to employ user-vetted prototypes in its applications development process
Goldstein (2005)	The JAD sessions had produced an exhaustively detailed requirements document.
Goldstein (2005)	To formally define what users needed the VCF to do for them, SAIC embarked on a series of Joint Application Development (JAD) sessions. In these meetings, Depew's team of agents and experts got together with a group of SAIC engineers to hash out what functions the VCF would perform.
FBI (2002)	we found that the specific needs of the users, and of the FBI as a whole, were not adequately defined
Score	Overall Evaluation
1.5	Involved: Users were involved in the development of requirements, validation of design, and user acceptance testing.

Exhibit 127: USER Evaluation

f. Politics (POLT)

Political Involvement (POLT): The extent to which Politics played a role on the software development project.	
Source of Empirical Evidence	Empirical Evidence
Harris (2005)	By late 2004, the writing was on the wall. The FBI's Virtual Case File, a much anticipated program to electronically organize and store mountains of investigative information, was coming unglued. The project was over budget. It was late. And a veritable revolving door of chief information officers and project managers meant that VCF was dangling in the wind with no one to save it.
Harris (2005)	In July 2004, the bureau established an Office of the Chief Information Officer to centrally manage all technology responsibilities, activities, policies and employees. There had been CIOs at the bureau, but they controlled almost nothing. FBI divisions managed their technology investments on their own. They had varied processes and procedures.
Score	Overall Evaluation
0.0	Highly Political: Political behaviors involve external management and party's external to the project and/or parent organization.

Exhibit 128: POLT Evaluation

The overall score for the environment element was 12.4 of 25 possible points.

The overall FBI VCF scored 7.5/50 in the function element, 10.2/25 in the structure element, and 12.4/25 in the environment element. The overall score against the FSE Framework was 30.1 out of 100.

5. RECENT INFORMATION

The prime contractor for the FBI VCF system was Science Applications International Corporation (SAIC) of San Diego, California. The SAIC Vice President in charge of the VCF project was cooperating with the researcher and had arranged for three SAIC managers who worked on the VCF project to participate in the questionnaire surveys. However, one week prior to taking the survey the United States Senate added the following language to the Department of Justice, Science and Related Activities Appropriations Bill (Senate Report 109-280, 2005, p. 35):

The Committee expects the FBI to use all means necessary, including legal action, to recover all erroneous charges from the VCF contractor and,

once recovered, will allow the FBI to apply the recovered resources toward the SENTINEL project.

As a matter of course, the SAIC General Counsel prohibited their employees from further participation in this research. As a result, the survey questionnaire answers include comments from only government or former SAIC employees.

6. CONCLUSION

This case study has presented the important facts surrounding the design and cancellation of the virtual case file system at the Federal Bureau of Investigation. The 1st part of the study provided background material essential in understanding the decision made to adopt a complex information system for handling case files at the FBI. The 2nd part of the study reviewed the scope of the VCF software system. The 3rd part of the case study reviewed the outcome of the effort to design and implement the VCF at the FBI. The 4th part of the case study evaluated the FBI VCF project using the 60 measurement objects in the Function-Structure-Environment (FSE) Framework. The 5th and final part was included to alert the reader to the continuing battle over this important software project.

A complete interpretation of the evaluation against the FSE Framework has been provided in Chapter 6, Discussion of Results. If the reader is interested in additional details related to the FBI VCF system a complete list of references is provided in the next section. While there is no single article that summarizes all of the issues, the most thorough analysis are presented in the article by Goldstein (2005) and the review conducted by the National Research Council (McGroddy & Lin, 2004).

7. FBI VCF CASE STUDY REFERENCES

Alfonsi, B. (2005). "FBI's Virtual Case File Living in Limbo," *IEEE Security and Privacy*, Vol. 3, No. 2, p. 7.

Department of Justice (2005). *The Federal Bureau of Investigation's Management of the Trilogy Information Technology Modernization Project* (Audit Report No. 05-07). Washington, DC: Office of the Inspector General

Federal Bureau of Investigation (2002). *Federal Bureaus of Investigation's Management of Information Technology Investments* (Report No. 03-09). Washington, DC: Office of the Inspector General.

Federal Bureau of Investigation, (2006). *Federal Bureaus of Investigation's Pre-acquisition Planning for and Control Over the Sentinel Case Management System* (Report No. 06-14). Washington, DC: Office of the Inspector General.

General Accountability Office (2003). *FBI Needs and Enterprise Architecture to Guide Its Modernization Activities* (GAO-03-959). Washington: Government Printing Office.

General Accountability Office (2004a). *Information Technology: Foundational Steps Being Taken to Make Needed FBI Systems Modernization Management Improvements* (GAO-04-842). Washington: Government Printing Office.

General Accountability Office (2004b). *FBI Continues to Make Progress in Its Efforts to Transform and Address Priorities. Statement of Laurie E. Ekstrand, Director, Homeland Security and Justice Issues and Randolph C. Hite; Director, Information Technology Architecture and Systems Issues* (GAO-04-578T). Washington: Government Printing Office.

General Accountability Office (2005a). *FBI is Taking Steps to Develop an Enterprise Architecture, but Much Remains to Be Accomplished* (GAO-05-363). Washington: Government Printing Office.

General Accountability Office (2005b). *Information Technology: FBI is Building Management Capabilities Essential to Successful System Deployments, but Challenges Remain: Statement of Randolph C. Hite; Director, Information Technology Architecture and Systems Issues Division* (GAO-05-1014T). Washington: Government Printing Office.

General Accountability Office (2006). *Weak Controls over the Trilogy Project Led to Payment of Questionable Contractor Costs and Missing Assets* (GAO 06-306). Washington: Government Printing Office.

Goldstein, H. (2005). "Who Killed the Virtual Case File?" *IEEE Spectrum*, Vol. 42, No. 6, pp. 24-35.

Harris, S. (2005). "The Turnaround," *Government Executive*, Vol. 37, No. 15, pp. 27-28.

Hickerson, T. B. (2006). *Failure at the Speed of Light: Project Escalation and De-escalation in the Software Industry*, (Master of Arts in Law and Diplomacy Thesis, Tufts University, 2006).

- Kumagai, J. (2003). "Mission impossible?" *IEEE Spectrum*, Vol.40, No.4, pp. 26-31.
- McGroddy, J.C. & Lin, H.S. (Eds.) (2004). *A Review of the FBI's Trilogy Information Technology Modernization Program*. Washington: The National Academy Press.
- United States Senate (2005a). *Statement of Robert S. Mueller, III; Director, Federal Bureau of Investigation* (Before the Appropriations Subcommittee on Commerce, Justice, State and the Judiciary, February 3, 2005). Washington: Government Printing Office.
- United States Senate (2005b). *Record Testimony of Arnold Punaro; Executive Vice President and General Manager – Washington Operations, SAIC* (For the Appropriations Subcommittee on Commerce, Justice, State and the Judiciary, February 3, 2005). Washington: Government Printing Office.
- Zorpette, G. (2002). "Feds Online: New FBI Computers Promise Access, But Will They Be Mole-Proof?" *IEEE Spectrum*, Vol. 39, No.9, pp. 36-37.

VITA

Dr. Adams is an Adjunct Assistant Professor at Virginia Wesleyan College and at the University of Maryland University College. Dr. Adams holds a B.S. in Ceramic Engineering from Rutgers University, an M.S. in Naval Architecture and Marine Engineering and an M.S. in Materials Engineering both from the Massachusetts Institute of Technology, and a Ph.D. in Engineering Management from Old Dominion University. Prior to joining the teaching profession Dr. Adams had more than 25 years of program, financial, and technical project management experience in positions of increasing responsibility in private industry and the military. Dr. Adams was personally responsible for the introduction, development, and maintenance of formal software engineering methods and tools for a large U.S. Department of Defense central design activity responsible for the maintenance of all Navy depot maintenance software systems. As a Project Manager Dr. Adams has been responsible for all aspects of a geographic region of a large information technology company with an annual budget of \$300 million; project manager for two nuclear submarine overhauls each worth in excess of \$240 million; and chief financial officer for a large Naval industrial activity with responsibility for the expenditure of, and accounting for, all customer funding and contracts for 1.5 million man-days of productive effort at a value in excess of \$750 million. Dr. Adams' research activities include software engineering project management frameworks, systems engineering methodologies, system of systems engineering, the implementation of Enterprise Resource Planning (ERP) systems, and the use of enterprise architectures. He is a retired Navy submarine officer, a senior member of the IEEE, a member of the American Society for Engineering Management, the American Association of University Professors, and the United States Naval Institute.