2005

Software Development Project Risk Management: A Literature Review

Kevin MacG. Adams
Old Dominion University

C. Ariel Pinto
Old Dominion University

Original Publication Citation

MacAdams, K. G., & Pinto, C. A. (2005). *Software development project risk management: A literature review*. 26th Annual National Conference of the American Society for Engineering Management 2005 - Organizational Transformation: Opportunities and Challenges, ASEM 2005, Virginia Beach, Virginia, 26-29 October, 2005. (pp. 635-641). American Society for Engineering Management.

SOFTWARE DEVELOPMENT PROJECT RISK MANAGEMENT: A LITERATURE REVIEW

Kevin MacG. Adams, Old Dominion University
C. Ariel Pinto, Old Dominion University

Abstract
The rapid and unprecedented growth in software has brought with it some of the most spectacular and costly project failures in modern history. How risk management is presented in the scholarly journals may give insight into the risk management methods and techniques in use on software development projects. This paper provides a glimpse into the risk management methods, methodologies and techniques available to those who are responsible for software development projects by conducting a non-experimental content analysis. The findings reveal that risk management has not received sufficient attention and does not appear to be widely accepted within the software engineering community.

Introduction
The overall goal of the introduction is to frame the information to be used in the literature review. A non-experimental content analysis will serve as the research method. The content analysis will include a detailed and systematic examination of the contents of the software engineering body of knowledge related to software project risk management in order to identify patterns, themes, or biases that contribute to the current practice of risk management in the software engineering community (Creswell, 2003). The specific characteristics and qualities of the content analysis will focus on: (1) the Software Engineering Body of Knowledge (SWEBOK) which serves to describe the generally accepted body of knowledge for software engineering; and (2) a review of the software engineering scholarly journals for recent articles on risk management and how they relate to software project development.

In the interest of brevity we have excluded: (1) the principal texts used in teaching software engineering, software engineering management, and software risk management; (2) the Project Management Body of Knowledge (PMBOK) which serves to describe the generally accepted body of knowledge for project management; and (3) a review of the scholarly journals on management and project management. These sources are recommended for inclusion in a larger and more comprehensive study of this subject.

Software Engineering Beginnings. The term software engineering was coined at a NATO Science Committee Conference in 1968 (Naur & Randell, 1969). The meeting of 50 top programmers, computer scientists, and captains of industry convened to plot a course out of what had come to be known as the "software crisis." Although the experts could not contrive a road map to guide the industry toward firmer ground, they did coin a name for that distant goal: software engineering (Gibbs, 1994). In the thirty-seven years since the term software engineering was coined some progress has been made toward the development of a profession of software engineering. Two specific actions have particular importance in the movement toward and creation of the discipline of software engineering. They are (1) the creation of the Software Engineering Institute and (2) the participation and leadership of professional societies associated with the emerging discipline of software engineering.

Software Engineering Institute. Early leadership in software engineering was initiated in the public sector. In the late 1970s and early 1980s the United States Department of Defense was the largest user and acquirer of software systems in the world, investing large sums of money developing and maintaining highly sophisticated and enormously complex software systems for a wide variety of command and control, weapons guidance, and defensive early-warning systems. In an effort to acquire and sustain its software-intensive systems with predictable and improved cost, schedule, and quality the Department of Defense initiated a charter for a federally funded research and development center focused on software. In 1984 Carnegie Mellon University was awarded a contract for and established the Software Engineering Institute (SEI) to advance the practice of software engineering. The SEI has made a number of noteworthy contributions in the emergence of software engineering; however, the Capability Maturity Model Integration (CMMII®) has proven to be the SEI's premier work. The CMMI® is a five-stage model that provides software organizations with guidance on how to gain control of their processes for developing and maintaining software and how to evolve toward a culture of software engineering and management excellence. The model's five levels are progressive and made up of process areas (PAs). Risk Management is a

Level 3 PA with the following purpose (CMMI®, 2002, p. 272):

> "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."

The Risk Management PA has three goals supported by seven specific practice requirements. A software developer must show convincing evidence to a group of certified assessors that the seven practice requirements have been successfully implemented, and that objective quality evidence exists to support such a finding. The inclusion of a Risk Management PA in the CMMI® has strongly influenced the adoption and practice of risk management in software engineering.

Professional Societies. Two principal professional societies claim software engineers in their membership: The Institute of Electrical and Electronics Engineers – Computer Society (IEEE-CS), and the Association of Computing Machinery (ACM). In May of 1993 the IEEE-CS Board of Directors approved a motion to "establish a steering committee for evaluating, planning, and coordinating actions related to establishing software engineering as a profession." In August of 1993 the ACM Council followed suit and endorsed the "establishment of a Commission on Software Engineering to address a number of questions relating to: (1) the terminology used to describe software engineering and those who work in the software area; (2) the identification of generally accepted and desired standards of good software practice; and (3) our ability to identify, educate, and train individuals who are competent with software engineering and design." By the end of 1993 the IEEE-CS and the ACM had created a Joint IEEE-CS and ACM Steering Committee for the Establishment of Software Engineering as a Profession. The committee's task was primarily to "establish the appropriate set(s) of criteria and norms for professional practice of software engineering upon which industrial decisions, professional certification and education curricula can be based" (Thompson & Edwards, 2001, p.44). In 1998 the Joint Steering Committee created in 1993 was superseded by a new committee called the IEEE-CS/ACM Software Engineering Coordinating Committee (SWECC). The SWECC was "responsible for coordinating, sponsoring and fostering all the various activities regarding software engineering within the IEEE-CS and ACM's sphere of operation. These included areas such as: standards of practice and ethics, body of knowledge, curriculum guidelines, and exam guidelines" (Thompson & Edwards, 2001, p.45). The Body of Knowledge is particularly noteworthy, with the Trial Version of the Guide to the Software Engineering Body of Knowledge (SWEBOK) being issued in 1999.

Software Engineering Body of Knowledge. The Software Engineering Body of Knowledge or SWEBOK, serves as a compendium and guide to the body of knowledge that has been developing and evolving over the past four decades. The stated purpose of the SWEBOK is to (Abran & Moore, 2004, p. xix):

> "Provide a consensually-validated characterization of the bounds of the software engineering discipline and to provide a topical access to the Body of Knowledge supporting that discipline. The Body of Knowledge is subdivided into ten software engineering Knowledge Areas (KA) plus an additional chapter providing an overview of the Knowledge Areas of strongly related disciplines. The descriptions of the KAs are designed to discriminate among the various important concepts, permitting readers to find their way quickly to subjects of interest."

The knowledge areas (KAs) include the major topics covered in most software engineering texts and curriculums. Users of the SWEBOK have been encouraged to note that the content of the SWEBOK is markedly different from Computer Science. The Editors of the Guide to the SWEBOK go on to make an important distinction between science and engineering (Abran & Moore, 2004, p. xix):

> "Just as electrical engineering is based upon the science of physics, software engineering should be based on computer science. In both cases, though, the emphasis is necessarily different. Scientists extend our knowledge of the laws of nature while engineers apply those laws of nature to build useful artifacts, under a number of constraints. Therefore, the emphasis of the Guide is placed upon the construction of useful software artifacts."

The ten software engineering knowledge areas in the SWEBOK are: (1) software requirements, (2) software design, (3) software construction, (4) software testing, (5) software maintenance, (6) software configuration management, (7) software engineering management, (8) software engineering process, (9) software engineering tools and methods, and (10) software quality.

The seventh area, software engineering management is defined as (Abran & Moore, 2004, p. 8-1):

> "The application of management activities – planning, coordinating, measuring, monitoring, controlling and reporting – to ensure the development and maintenance of software is systematic, disciplined and quantified."

The management of software projects has become a focused discipline within the field of software engineering. Most Master's degree programs in software engineering include a course in software project management as a requirement. Companies that develop and maintain software systems include software project managers as distinct resources. The software engineering management domain is well defined in the SWEBOK and has six topic-based sections: initiation and scope definition, software project planning, software project enactment, review and evaluation, and software engineering measurement. Risk management is a sub-topic in the software project planning section.

Software Engineering Scholarly Journals. Scholarly journals related to software engineering should be a rich source of information on software project risk management. The software engineering literature, like the field of software engineering, is young with the oldest journal being just under 50 years of age. The number of scholarly journals and publications focused solely on software engineering has increased with the emergence of the profession. The two principal professional societies, IEEE-CS and ACM, have contributed significantly and now publish eight peer reviewed journals on software engineering.

1. *Communications of the ACM (CACM).* This is the flagship publication of the ACM and one of the oft-cited magazines in the computing field. Established in 1957 as a vehicle for ACM members to communicate their research findings and ideas, Communications has flourished into the premier computing magazine, internationally renowned and respected for its coverage of both existing and emerging technologies.

2. *IEEE Transactions on Software Engineering (TSE).* An archival journal published monthly. Area of interest includes well-defined theoretical results and empirical studies that have potential impact on the construction, analysis, or management of software. The scope of TSE ranges from the mechanisms through the development of principles to the application of those principles to specific environments.

3. *ACM Transactions on Software Engineering and Methodology (TOSEM).* The purpose of TOSEM is to publish original and significant results in all areas of software engineering research. In general, the systems to which the results apply should be sufficiently complex and long-lived to justify investment in languages, methods, and tools that support specification, design, implementation, validation, documentation, maintenance, reengineering, and other related activities.

4. *IEEE Computer.* Monthly magazine that includes survey and tutorial articles covering a broad range of computer hardware, software, and system design and application. Regular departments present hot topics, product reviews and announcements, media reviews, and professional news and media calendars.

5. *ACM Computing Surveys.* A journal which publishes surveys, tutorials, and special reports on all areas of computing research.

6. *ACM SIGSOFT Engineering Notes.* The ACM Special Interest Group on Software Engineering provides a forum for computing professionals from industry, government and academia to examine principles, practices, and new research results in software engineering.

7. *The Journal of the ACM (JACM).* Aims to provide coverage of the most significant work going on in computer science, broadly construed. Publishes original research papers of lasting value in computer science. To be accepted, a paper must be judged to be truly outstanding in its field and to be of interest to a wide audience. ACM is particularly interested in work at the boundaries, both the boundaries of sub-disciplines of computer science and the boundaries between computer science and other fields.

8. *IEEE Software.* Monthly magazine that includes case studies and surveys on current techniques and new products in software design and development.

However, a large number of software engineering articles also appear within the literature in the field of information systems. A recent study ranked the top 50 Information Systems (IS) journals (Mylonopoulos & Theoharakis, 2001). Seven of the eight ACM and IEEE publications were ranked 2nd, 6th, 13th, 19th, 24th, 26th and 45th, respectively. Only IEEE Software, which contains a mix of articles on research and techniques, was unranked in the study. The fact that software engineering research articles continue to be published in a number of major IS journals requires these journals to be considered in the literature review. In the interest of being able to conduct a productive study of the literature in a reasonable period of time, the journal article literature review for software engineering research methods and designs has been

limited to: (1) the top 10 IS journals reported by Mylonopoulos & Theoharakis (2001) and (2) the eight IEEE and ACM journals. Exhibit 1 is a listing of the journals to be included in the software engineering literature review based on this criterion.

Exhibit 1: Journals in Software Engineering Literature Review

Rank	Title
1	MIS Quarterly
2	Communications of the ACM
3	IS Research
4	Journal of MIS
5	Management Science
6	IEEE Transactions on Software Engineering
7	Harvard Business Review
8	Decision Sciences
9	Decision Support Systems
10	Information and Management
11	ACM Transactions on Software Engineering and Methodology
12	IEEE Computer
13	ACM Computing Surveys
14	ACM SIGSOFT
15	Journal of the ACM
16	IEEE Software

Content Analysis

An essential part of the research process is a thorough review of the related literature. The review will describe the method and techniques found in the literature surrounding the practice of software project management. In this case a review of the formal body of knowledge (SWEBOK) and significant scholarly journals will form the framework for the analysis.

Software Engineering Body of Knowledge (SWEBOK). The SWEBOK has one paragraph devoted to the description of the activity of risk management. The paragraph states (Abran & Moore, 2004, p. 8-5):

> "Risk identification and analysis (what can go wrong, how and why, and what are the likely consequences), critical risk assessment (which are the most significant risks in terms of exposure, which can we do something about in terms of leverage), risk mitigation and contingency planning (formulating a strategy to deal with risks and to manage the risk profile) are all undertaken. Risk assessment methods (for example, decision trees and process simulations) should be used in order to highlight and evaluate risks. Project abandonment policies should also be determined at this point in discussion with all other stakeholders. (Dorfman & Thayer, 2002, Pfleeger, 2001, Pressman, 2004, Reifer, 2002, Somerville, 2005, Thayer, 1997) Software-unique aspects of risk, such as software engineers' tendency to add unwanted features or the risks attendant in software's intangible nature, must influence the project's risk management."

This is a very simplistic view of risk management and refers the user of the SWEBOK to the references cited within the paragraph. Each of these references is a general text on software engineering and gives limited treatment to risk management. The SWEBOK chapter on software engineering management includes a list of further readings. Of the 71 references cited, 17 are papers on risk management and 1 is a text on Software Engineering Risk Management.

Software Engineering Scholarly Journals. A detailed search for articles on risk management and risk analysis in each of the 16 scholarly journals included in Exhibit 2 was conducted. The University of Maryland's Library database was used to access the IEEE Computer Society Digital Library, the ACM Digital Library, Business Source Premier (EBSCO), ScienceDirect (Elsevier B.V.), and ABI/INFORM (Proquest) databases. Two independent key word searches were conducted on each database; the first for risk management and the second for risk analysis. 57 articles were retrieved with 41 categorized as discussing risk management and 16 discussing risk analysis. Exhibit 2 shows the distribution by type and publication for journals that published three or more total articles on risk.

Exhibit 2: Articles by Journal and Risk Category

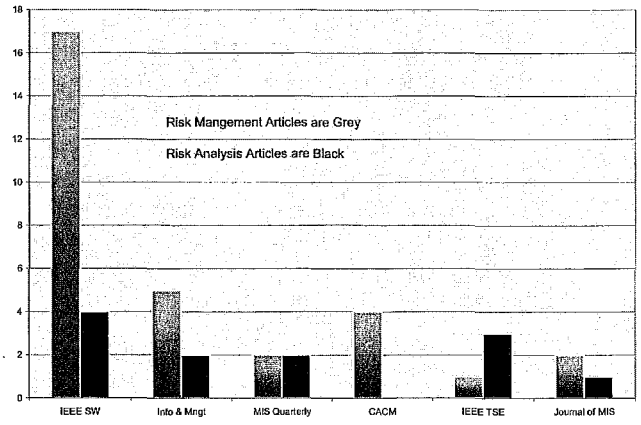
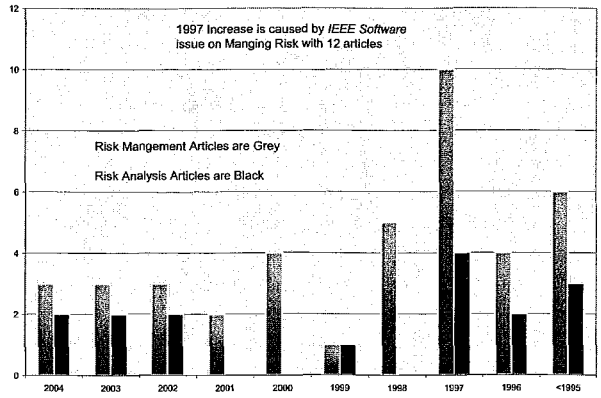


Exhibit 3 depicts the distribution of paper types against the year published. A review of Exhibit 3 shows that for the ten year period from 1994 to present, the major software engineering journals have published, on average; about 2-3 articles per year on either risk management or risk analysis. An interesting fact emerges as a result of this analysis; 11 of the 57 risk management and analysis papers returned in the scholarly journal search are included in the 17 papers in the SWEBOK reading list on risk management.

We were able to obtain 55 of the 57 papers for additional analysis (the two 1993/1994 Information and Management articles were beyond our subscription access rights). Each paper abstract was reviewed to ascertain its applicability for a software development project. Our subjective analysis revealed that 22 of the

Exhibit 3: Journal Article by Risk Category and Year Published



55 papers contained material on risk management (18 papers) and risk analysis (4 papers) that could be applied directly to projects that develop software. Once again it is interesting to note that 5 of the 22 papers were also included on the SWEBOK reading list on risk management. Each of the 22 papers was reviewed in an effort to better understand the risk management methods, methodologies and techniques available to those who are responsible for software development projects. Our analysis shows that for the ten year period from 1994 to present, the major software engineering journals have published, on average, about 1-2 articles per year on either risk management or risk analysis that are applicable to software projects.

The 22 papers covered a wide range of software project risk management topics but fit into one of four generalized areas that deal with risk. The areas are:

Approaches, Models and Frameworks (12 papers, 1 From SWEBOK): Papers in this area deal with risk management through the use of a model, approach or framework.

Implementation (5 papers, 4 from SWEBOK): Papers in this area describe the actual implementation of a risk management program on a software development project.

Methods (3 papers): These papers address a particular risk management method used on a software development project.

Factors (2 papers): These papers discuss specific risk and success factors associated with software development projects.

Four of the five papers included in the SWEBOK reading list fall into the implementation area and one is in the approaches, models and frameworks area. This pattern indicates that the SWEBOK places a heavy emphasis on implementation of risk in software projects. No further patterns were discernable from the data.

Risk Management and Software Projects. A recent study suggests that the net marginal effect of higher CMMI® Levels for software projects is significantly higher product quality, and reduced cycle time and development effort (Harter, Krishnan & Slaughter, 2000). Because Risk Management is a CMMI® Level 3 PA we must assume that Level 3 projects that practice risk management exhibit higher quality, reduced cycle time and development effort and that projects that fail to reach CMMI® Level 3 are the major contributors to the spectacular software project failures reported in the literature (Gibbs, 1994).

There is an additional factor that must be considered for software projects. Because software engineering has adopted many of the established project management practices and techniques, they often rely on techniques that may not work when designing and building software. Spector and Gifford (1986, p. 222) compare bridge design and computer system (including software) design and manufacture. They observe that project management tools in bridge construction are somewhat standardized because most bridges are designed and built using the same in principles; the same can not be said about software projects. We question whether the use of the CMMI® and standard processes has introduced a corresponding degree of project and risk management heterogeneity on software projects?

Finally, the growing popularity of using third-party software Commercial Off-The-Shelf (COTS) components introduces additional uncertainty for software development projects. The unpredictable quality of COTS components and their effect on software projects highlights the need for increased risk management techniques (Sedigh-Ali, Ghafoor & Paul, 2001).

While it is undeniable that software is becoming more ubiquitous, software projects continue to fail. Software development entities have been increasing their CMMI® maturity levels in an effort to use repeatable processes to ensure on-time and on-budget deliveries. Knowing that Risk Management is a level 3 CMMI® PA we would expect to see additional articles in the literature related to risk management. However, the number of risk related article in the scholarly journals continues to remain constant.

Conclusions

A review of the principal software engineering scholarly journals and the body of knowledge reveals few papers that discuss methods and techniques for implementing risk management and/or analysis on software development projects. Twenty-two (22) papers have been cited in the period 1994-2004 and address four generalized areas of that deal with risk on software projects. It is noteworthy that most of the papers deal with approaches, models, methods and frameworks and very few deal with the factors surrounding risk management implementation on software projects. The analysis also shows that the SWEBOK places a heavy emphasis on implementation of risk management in software projects but that these papers, papers that represent the best submissions on software project risk management/analysis, have not received sufficient attention and do not appear to be widely accepted within the software engineering community. We believe that risk management in software engineering is still in flux but that the limited study conducted in this paper provides a solid foundation for engineering managers and a framework for more-in-depth research in software project risk management.

Implications

The implications for extension of the work begun in this paper are twofold. The first is for engineering managers. Engineering managers can benefit directly from the content analysis by reviewing those papers that are applicable to their specific endeavors. They can also point out what elements of risk management are industry best practices and areas where increased utility is required in the practice of risk management while managing engineering projects.

The second is for researchers. The content analysis in this paper indicates the need for additional, focused research in software risk management. Areas include model development, implementation methods and techniques, and proof of concept on pilot projects.

References

Abran, Alain and James W. Moore (Eds.). *Guide to the Software Engineering Body of Knowledge*, 2004 Version. Los Alamitos: The Institute of Electrical and Electronics Engineers. (2004).

CMMI® (2002). *Capability Maturity Model® Integration (CMMI®)* Version 1.1 for Software Engineering. Pittsburgh: Carnegie Mellon University. (2002).

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

Dorfman, Merlin and Richard Thayer (Eds.). *Software Engineering*. New York: Wiley-IEEE Computer Society Press. (2002).

Gibbs, W. Wayt. "Software's Chronic Crisis," *Scientific American*, Vol. 271, No. 3 (September 1994), pp. 86-95.

Harter, Donald E., Mayuram S. Krishnan and Sandra A. Slaughter. "Effects of Process Maturity on Quality, Cycle Time, and Effort in Software Product Development," *Management Science*, Vol. 46, No. 4 (April 2000), pp. 451-466.

Leedy, Paul D. and Jeanne E. Ormrod. *Practical Research: Planning and Design* (7th ed.). Upper Saddle River: Prentice-Hall. (2001).

Mylonopoulos, Nikolaos A. and Vasilis Theoharakis. "Global Perceptions of IS Journals," *Communications of the ACM*, Vol. 44, No. 9 (September 2001), pp. 29-33.

Naur, Peter and Brain Randell (Eds.). *Report on a conference sponsored by the NATO Science Committee at Garmisch, Germany, October 7-11, 1968.* Brussels: North Atlantic Treaty Organization. (1969).

Pfleeger, Shari L. *Software Engineering: Theory and Practice* (2nd ed). Upper Saddle River: Prentice-Hall. (2001).

Pressman, Roger S. *Software Engineering: A Practitioner's Approach* (6th ed.) New York: McGraw-Hill. (2004).

Reifer, Don (Ed.). *Software Management*. New York: Wiley-IEEE Computer Society Press. (2002).

Sedigh-Ali, Sahra, Arif Ghafoor and Raymond A. Paul. "Software Engineering Metrics for COTS-Based Systems," *IEEE Computer*, Vol. 34 No. 5 (May 2001), pp. 44-50.

Sommerville, Ian. *Software Engi*neering (7th ed.) New York: Addison-Wesley. (2005).

Spector, Alfred and David Gifford. "A Computer Science Perspective of Bridge Design,"

Communications of the ACM, Vol. 29, No. 4 (April 1986), pp. 367-283.

Thayer, Richard. (Ed.). *Software Engineering Project Management*. New York: Wiley-IEEE Computer Society Press. (1997).

Thompson, J. Barrie and Helen M. Edwards. "Current Issues in the Area of Software Engineering Professionalism," *ACM Software Engineering Notes*, Vol. 26, No. 6 (November 2001), pp. 44-46.

About the Authors

Kevin MacG. Adams received two M.S. degrees from the Massachusetts Institute of Technology in Naval Architecture and Marine Engineering and Materials Engineering. He holds a B.S. in Ceramic Engineering from Rutgers University. He is currently a Principal Business Systems Analyst at CACI and a Ph.D. student in Engineering Management at Old Dominion University. He has over 25 years of Department of Defense and commercial management experience in Project Management and Software Engineering. He is an adjunct Assistant Professor at the University of Maryland University College where he teaches graduate courses in Software Engineering and at Virginia Wesleyan College where he teaches undergraduate courses in Information Systems.

C. Ariel Pinto received his Ph.D. in Systems Engineering from the University of Virginia and his M.S. and B.S. in Industrial Engineering from the University of the Philippines. Dr. Pinto is an Assistant Professor of engineering management and systems engineering at Old Dominion University. His research focuses on risk management in engineered systems, and systems engineering.