2009

An Extended Interoperability Framework for Joint Composability

Andreas Tolk
Old Dominion University, atolk@odu.edu

Charles D. Turnitsa
Old Dominion University, cturnits@odu.edu

Original Publication Citation

Tolk, A., & Turnitsa, C. D. (2009). *An Extended Interoperability Framework for Joint Composability.* Paper presented at the 2009 Fall Simulation Interoperability Workshop, Orlando FL.

An Extended Interoperability Framework for Joint Composability

Andreas Tolk, Ph.D.
Frank Batten College of Engineering & Technology
Old Dominion University
Norfolk, VA 23529
atolk@odu.edu

Charles D. Turnitsa
Virginia Modeling Analysis and Simulation Center
Old Dominion University
Suffolk, VA 23435
cturnits@odu.edu

Keywords:
Joint Common Object Model (JCOM), Levels of Conceptual Interoperability Model (LCIM), Interoperability and Composability

ABSTRACT: *Interoperation of systems is defined by the aspects of integratability, interoperability, and composability. It is therefore needed, to address all levels of interoperation - from conceptual models via implemented systems to the supported infrastructure - accordingly in an interoperation framework.*

Several candidates are available and provide valuable part solution. This paper evaluates the Base Object Models (BOMs), Discrete Event Simulation Specifications (DEVS), Unified Language Model (UML) artifacts as used within the Test and Training Enabling Architecture (TENA), the Object-Process Methodology (OPM), and Conceptual Graphs (CG) regarding their contribution.

Using the Levels of Conceptual Interoperability Model (LCIM), an extended interoperability framework based on the contributions of BOM, DEVS, UML/TENA, OPM, and CG will be proposed and gaps in support of joint composability are indentified.

1 Introduction

The variety and distinction present among all systems that may need to be brought together for the purposes of supporting Joint exercises is so large as to be almost uncountable, especially when also accounting for the various data models and perspectives that each system can espouse. In light of this broad variety, the authors propose that a framework that seeks to describe methods for attaining composability between such systems needs to be broad and (in as much as possible) non-exclusionary. In order to contribute to such a framework, this paper seeks to categorize several widely known and widely used methods for modeling systems and the interoperation between systems. From this categorization, some highlights as to how each method can contribute to an overall framework is identified, and finally these are all described in terms of the Levels of Conceptual Interoperability Model, such that their contribution to a complete conceptual composition can be evaluated.

From the presentation of [1], it can be seen that a goal of the Joint Common Object Model project is the increased interoperability and composability of systems, achieved through methods that make use of increased conceptual expressivity between object models, for the purposes of discovery (for reuse of existing object models) and interoperation (to increase the capability for conceptual expression across existing architectures). The use of meta-models describing object models (OMs), and OM components, from a variety of architectures is key to this process. Such OMs (especially for architectures such as HLA and TENA, but also to include DIS and CTIA) describe the entities (data objects and processes) that are exchanged within their respective architectures. If a meta-modeling technique can be applied to capture the expressed meaning of such OMs, and then operate on that meaning to see how such meaning can be expressed in other OMs, this will greatly facilitate the reuse of existing architectural artifacts (new federations with existing OMs), as well as assist greatly in the design and implementation of new architecture instances (new federations with new OMs).

The criteria to be used for searching across such a meta-modeling technique would come, at least within the training community, from well defined training objectives, expressed in doctrinally/operationally sound language, such as a Joint Mission Thread (JMT). The meta-modeling technique used, must therefore support the representation of the concepts used in defining a training mission, but also be able to translate it into the

elements of existing OM components, for purposes of meaningful search. JMT are intended to be an end-to-end description, relying on a common process model (a conceptual model) that will identify all of the essential activities and operational nodes required to complete the mission being described. Such threads have been mandated by the Joint Battle Management Command and Control road map [2], and are to be developed by Joint Forces Command.

added to an expanding repository of federate OMs. Finally, if no matching, or near-matching, candidates can be found implementation of a new configuration of federate, resulting in an OM that matches the semantic needs of the mission, can be made.

Within the operational scope of JCOM as presented in figure 1, it can be seen that the best components selected for step five in the process may come from different types of federates, employing different

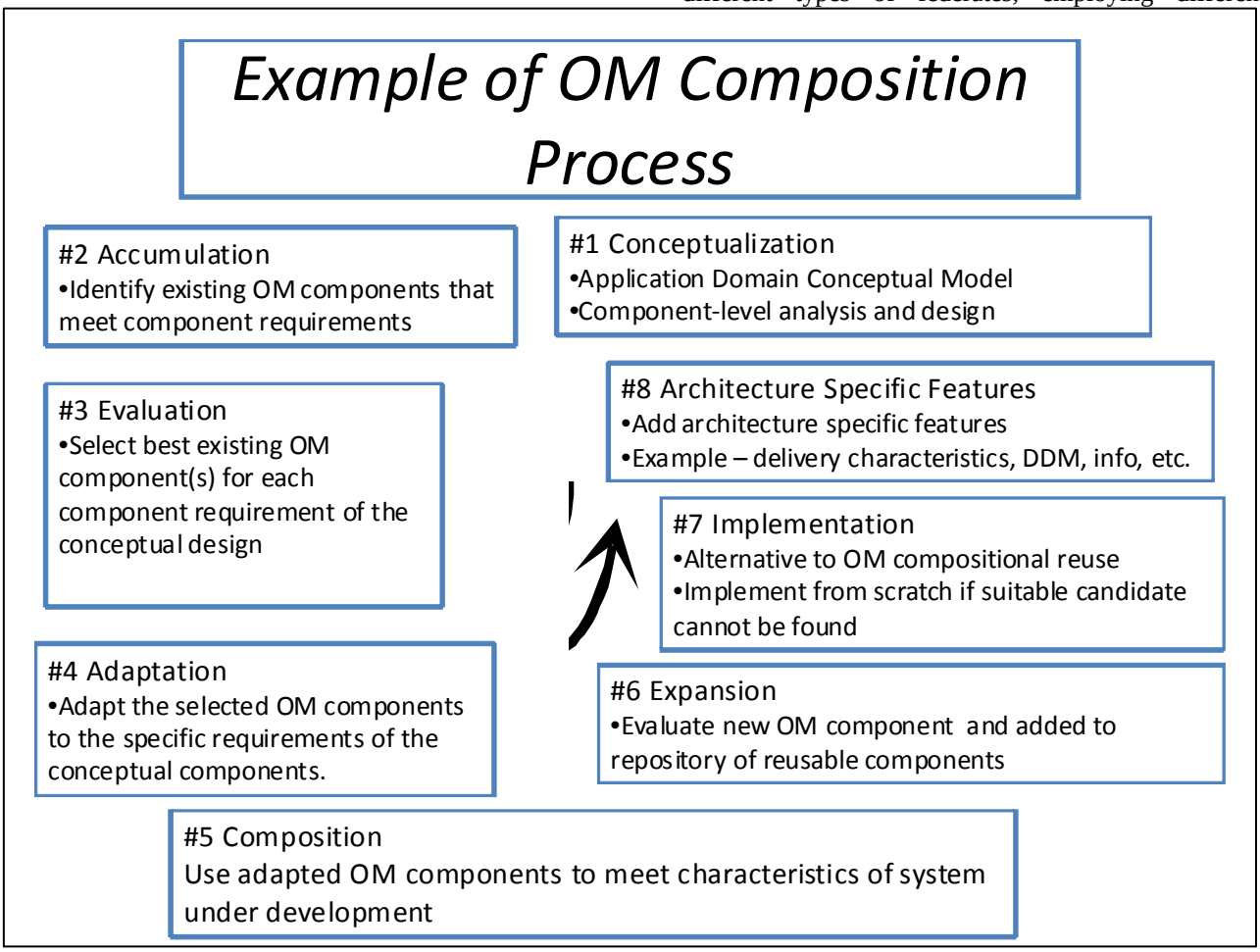

Figure 1- JCOM Composition Process

The JCOM operational picture, from [12], is to begin with a requirement, from training objectives, expressed in a conceptual model of the JMT that make up the training mission. That conceptual model will be used as the basis for a semantic search, resulting in an accumulation of existing object models (representing federates) that meet the approximate needs. From that accumulation, an evaluation of the best fitting candidates can be made. The selected candidates will most likely have to be put through the process of adaptation, bringing them into specific alignment with the composition goals to support the mission. The adapted candidates are now composed. The results are

distributed simulation techniques, and requiring a framework that is both flexible and allows for different levels of conceptual expressivity. One method for describing the different levels of conceptual expressivity is the Levels of Conceptual Interoperability model. Such a model can serve as a guide to the strength of contributing methods, as well as serve as the benchmark to highlight what must be employed to reach the higher levels in the model.

2 Levels of Conceptual Interoperability Model

The Levels of Conceptual Interoperability Model (LCIM) is a model that stratifies the continuum of possible conceptual expressivity within an interoperable coupling of systems. It stratifies such expressivity into a number of layers, and describes each layer. The uses of such a model are varied, but two broad categories exist. The first use of such a model is to rely on the model to describe an interoperability scenario that already exists, and then measures (based on how rich the expression of conceptual meaning between the systems is) how high up the levels of the model the interoperability can be said to be. The second use of the model is to describe the requirements for attaining a particular level, in terms of what a system must be able to express to another system (meaningfully) in order to satisfy that level.

From the LCIM, it can be seen that several levels are each grouped together under a general title – the lowest levels (levels 1,2) represent integratability, the middle levels (levels 3,4) represent interoperability, and the highest levels (levels 5,6) represent composability. The integratability levels describe the means of attaining connection between systems (level 1), and the means of exchanging electronic communications (level 2), as such they don't address the nature of the data being exchanged, nor its meaning, beyond the requirements that it structurally adopt the protocols mandated for its exchange. Such decisions are best left to the technicians who know their systems well and also know the restrictions and requirements for each operational instance where systems must be brought together. So long as such methods can support the higher groupings (interoperability and composability), the details here are not of interest to a framework describing conceptually meaningful exchange.

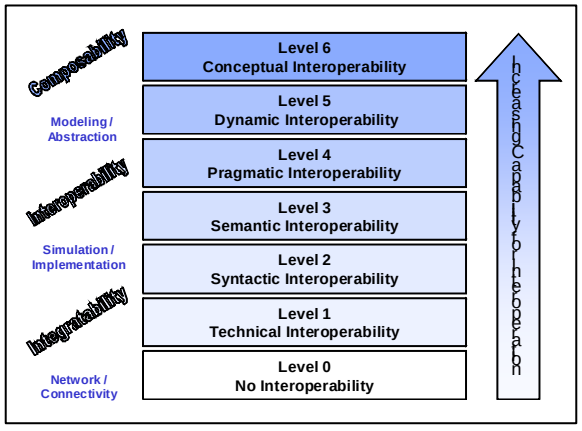

Figure 2 - Levels of Conceptual Interoperability Model

In light of the integratability levels not being addressed by a framework for composability, it then follows that the framework should concentrate on describing the capabilities and requirements at levels 3 and up of the LCIM.

3 Candidate Methods for Interoperation

There are a wide number of methods that currently support interoperation between systems, to a greater or lesser degree of conceptual expressivity. The more common of these methods are described here. Note that there is no prejudice either for or against methods that are used during development or during execution, only that they exist as a standard method agreed to by a community of users.

3.1 Base Object Models

Base Object Models (BOMs) are a description of objects to be simulated, based on a published standard that seeks to describe the object in such a way that interoperability and composability are supported. In short, the standard calls for a XML document describing an object's model in four main areas (and two minor ones). The four main areas are:

- Model identification, which is basically a header section
- Conceptual model, which consists of four elements, which are (1) a pattern description that describes the entity and events (actions) of the object and how they are related, (2) a state machine description of the pattern, (3) the entity description, and (4) the event description.
- Model mapping, which consists of mapping the entities and events identified in the conceptual model to the object classes and interaction classes described in the HLA object model.
- High Level Architecture (HLA) object model, which has three parts – the description of the HLA objects (used to define the BOM entities from the conceptual model), the HLA interactions (used to define the BOM events from the conceptual model), and finally the HLA data types used to help structure the events and interactions.

Also mentioned are two minor areas, which are the Notes and Definitions describing the BOM.

As can be seen from this description of the BOM definition, this is a method for conceptually modeling something so that it can be defined using HLA objects and interactions. For HLA based federations, this is a strong tool for having a standard description of what

HLA elements would be required in order to have a model of something represented. Defined by the layers of the Conceptual Interoperability Model, this seats a BOM description firmly between (entailing some aspects of both) level 3 (semantic interoperability) and level 4 (pragmatic interoperability). The dynamic level is seen as being supported, because the conceptual model, as well as the HLA objects and interactions, within the BOM description can describe a number of changing situations, representing the dynamic states of the system. The reason for this strata is because the BOM describes objects and actions with a rich semantic labeling method (the BOM XML document), which is satisfying level 3, and also gives (through the model mapping that ties together the conceptual model with the HLA object model) a definition of what is meant by the semantic objects (satisfying part of level 4). The reason that all of the pragmatic level is not included here, is because the contextual use of the objects and actions is not described (which is an intended strength of the BOM – it is context free).

3.2 Discrete Event System Specification

The Discrete Event System (DEVS) Specification is a mathematical formal method relied on for describing discrete and continuous systems [3]. The formalism has been extended to cover a large number of related systems, and in many different application areas. This includes extensions to cover continuous systems, and specifications aimed formally addressing the description of discrete event systems for many different types of implementations.

The formal method has been applied to many applications of dynamic system modeling and simulation, including distributed modeling, working with techniques such as HLA [4]. In brief, the formal method consists of a description of a system component expressed in a collection of sets and functions. These include a description of possible inputs, outputs and states of the component, as well as functions describing internal changes, external changes, output and time advance. Additionally there is a function that describes the situation where an internal change and external change are due to occur at the same time (confluence). Such component descriptions can be atomic (intended to be stand-alone) or

In terms of what a DEVS specification of a system can provide to an interoperability framework, it can present the components of the system, their interdependence (especially in terms of component input and output), and terms of component state context (in terms of internal state translation with regard to time advance) [5]. If the component identity descriptions (which are implied but not defined by the formalism) are based on an agreed-to set of tags or names, then LCIM levels 3 (semantic) are well supported. In addition, the richness of state derived situations that can be described within a DEVS specification of a system, also allow it to satisfy LCIM levels 4 and 5 (pragmatic and dynamic).

3.3 Test and Training Enabling Architecture

The Test and Training Enabling Architecture (TENA) is a method for describing objects and processes that are commonly encountered within simulations that work with defense testing environments and training exercises [11] – it has been mainly (but not exclusively) applied to support test range operations. It concentrates on providing a common architecture for different range environments to work together in describing simulated and real range activity (the phrase virtual range is used). This is done by providing for a common object model, as well as a software middleware that all objects can interact with.

The architecture is intended to be interactive with and compliant with HLA, but again, specifically concentrates on interoperability and reuse between ranges. Because of the features described here – that it is specifically for test range work; that it is intended to support simulating a virtual range; that it relies on a common object model and common middleware – it is very much capable of supporting activities that all support a common contextual viewpoint, and for a community that shares a common set of semantics.

The TENA middleware that is used to connect systems to the architecture network includes the provision for a repository of object models (representing the characteristics of the entities and events that are supported in the architecture). This repository and the object models it has described within it are what could be of use to a JCOM effort.

In terms of the LCIM, the TENA contribution to Joint Composability could be viewed as supporting interoperability at levels 4 (pragmatic) and 5 (dynamic). The reasons this appears to be so are (1) the pragmatic level is supported by having everything in the TENA repository being not only semantically labeled in a manner that the whole community agrees to, but also having meaning to other members of the community, and (2) the dynamic level is supported because the context is so well established among the community of users that even with a shifting context, the pragmatics remain known. Outside of an environment based on the TENA middleware, and supported by only systems described by TENA OMs, it appears as if only level 3 and part of level 4 can be supported.

3.4 Unified Modeling Language

The Unified Modeling Language (UML) is a method for describing, through a number of unified graphical methods, the objects and actions that make up a system. As this may also include a system of systems, it is a worthwhile method to consider as a candidate. The different diagramming techniques that make up UML are briefly introduced here:

- Use-Case diagram: provides an illustration of some element of functionality of the described system.
- Class diagram: illustrates the different objects of the system, and describes their relationships with each other.
- Sequence diagram: shows the flow of activities for a use case, or part of a use case. It specifically illustrates the calls that take place between different objects, and their relative timing.
- Statechart diagram: illustrates the different states that an object can be in, and how they are related.
- Activity diagram: shows the specific relationships between different objects describing activity that takes place between them. The illustration describes the activity, indicating which object it belongs to, and also what the next activity to take place will be (and for what object).
- Component diagram: illustrates the actual components of the system, and their functional dependencies and relations to other components in the system (such as software libraries, service components etc).
- Deployment diagram: shows the relationship of deployed (physical) components, and how they are interconnected. Useful for depicting system architecture or network architecture for dispersed components.

What UML specification of systems can bring to a Joint Composability Framework, in terms of providing for interoperability (by LCIM metrics), is partially up to level 5, dynamic, in that it shows the context and timing of the systems and their components, as well as attempting to show some of the dynamic nature of the system (the changes in relationship of the components, over). The depiction of definition of semantically described elements, however, is noticeably lacking from UML. There is no requirement within the diagramming techniques given, to show the structure, scope or resolution of the elements being addressed. One of the reasons that the authors allege that it is not completely satisfying level 5 is that it does not accomplish the goals of level 4 completely – the pragmatic definition of the system, and its components, at a level where the objects and processes are all defined, rather than just given semantic labels.

3.5 Object Process Methodology

The Object Process Methodology (OPM) is a method of describing a system, whereby data objects and states are equaled in their depiction by a similarly robust depiction of the processes that affect those objects, and connect those states. This is done by using a diagramming technique that has two types of elements – entities and links. Entities, within OPM, depict the objects, states, and process of a system. Links are of two varieties for OPM, structural links which relate two or more objects and procedural links which relate an object with a process.

OPM has a relationship to UML, in that both are graphical diagramming techniques, and more than that, certain types of UML diagrams can be mapped to from OPM diagrams. One of the strengths that OPM has in addition to all that UML can provide is its emphasis on the importance to model processes separately from objects, and also to provide for the different relationships that processes can have on objects, their potential states, and their actual state (which is referred to in OPM as an object's status). This gives a good overview of both the static view of the system (objects and states), as well as the dynamic view (processes and effects).

When viewed in light of its expressivity via the LCIM, OPM can describe a system with enough expressivity to show the pragmatic and dynamic (levels 4 and 5). What it offers in addition to some of the methods already discussed that operate at those levels (BOMs, TENA) is that it remains independent from the specific architectures that those systems espouse (HLA in the case of BOM, and the TENA architecture for the latter). Because of this, however, it does not have a specific manner for capturing the definitional attributes of objects and processes that those other architecture specific methods use. Because of this, OPM seems very strongly suited to be a method for connecting other methods at either level 4 or 5 (pragmatic or dynamic), by describing the objects and processes in the methods mandated by the pragmatic and dynamic level definitions, so that various architectures can make use of such a description for composability.

3.6 Conceptual Graphs

Conceptual Graphs (CG) are a method for describing the conceptual entities found within a system, both concepts and relations. They are based on an idea introduce in [6] by Sowa. It is a graphical method used to describe a logical system that is used to express

knowledge. As a potential contributor to a Joint Composability Framework, it would be a strong contender for describing the meaning of objects and of the relationships between objects. One of the original uses where the CG method was employed was to describe the conceptual constructs of relational data bases and data models.

In terms of the LCIM, CG provide a good method for describing the meaning of objects from the semantic level (level 3), so that they could be fully realized at the pragmatic level (level 4). This definition of objects and processes is more from a relationship perspective, than from the perspective of specific attribute maps or enumerated characteristics. There is some work being done, an early report being [7], that indicates the usefulness of CG in describing dynamic systems. A more modern description, reflecting some of the development done in the CG community, is in [14]. Specific usefulness for the JCOM might be in describing the relationships between objects and processes (or actors and roles), as the work in [8] describes.

4 Comparison of Methods

The previous six methods are not each placed in relationship to the LCIM, to describe the appropriate contribution to the overall interoperability/ composability picture that each can make. Of the six methods described here, two are related to specific architectures (BOM and TENA), two are architecture-free graphical representations of systems (UML and OPM), one is a method for describing the pragmatics of a data model (CG), and one (DEVS) is a formalism for describing the changing states within a system (interdependence and context).

Based on the findings of this paper, when expressed in comparison to the LCIM, the various methods described can be seen to address a range of levels of conceptual interoperability. From the figure, there are two types of identifiers describing which levels of the LCIM a method can express, for the purposes of contributing to a Joint Composability Framework. For each method, at the levels supported, the indicator is either a black "XX" representing a complete satisfying match, or a red "PP" representing a partial match.

Some description of what it takes to fully satisfy the levels 3, 4, 5 and 6 of the LCIM is in order here, so that the evaluation, and the contributing analysis, can be understood. The lowest of the levels above integratability, level 3 (semantic), is really the first level of interest to a Joint Composability Framework. As mentioned earlier, the lower (integration) levels (1 and 2) are not of interest to what can be expressed within the framework, as they are the domain of the implementations of a connecting architecture enabling the framework. So starting with level 3, semantic, a description of what is needed will be listed here.

4.1 LCIM Requirements

At the semantic level (level 3 of the LCIM), what is required is for all components and exchanged data to be given a meaningful name that is accepted by all systems that are involved in the interoperability. Note that meaning here is a shared meaning (from a common data model, or a taxonomical directory of terms) that is independent of the particular use of the data.

At the pragmatic level (level 4 of the LCIM), there are two requirements. First, the data elements and components are defined at this level. Commonly, this means that the structure, scope and resolution of such elements (and components that use them) are made explicit. Second, the context of application in which elements are used – both the processes of the system component affecting the elements, and also the place within the operational life span (accumulated time) where the element is intended to be used. This information is made clear by the origin system, and is understood by the receiving system.

At the dynamic level (level 5 of the LCIM), there are additionally two requirements, that are extensions of the pragmatic level. First, the definition of elements and components that took place at the pragmatic level (structure, scope and resolution) is further defined with respect to the dynamic states that change with respect to elapsed time within the operational span of the system. Second, the context of application of how such elements and components are used, and their relations to each other, as these (context and relations) change during the operational span of the system is understood, again with respect to elapsed time and dynamic system states. The state-specific definition of elements and components, and the state-specific definition of context are expressed from the origin system to the receiving system, and also the effects such exchange will have on the receiving system's elements and components (and their context) is understood.

At the conceptual level (level 6 of the LCIM), alignment must take place between systems as to their perspective and frame of reference on the referents being represented to each other. This is so that assumptions and constraints can be made explicit and understood by each other. Much more can be said in regards to the conceptual level, but as the elements of a Joint Composability Framework currently being assessed do not rise to this level of expressivity, this will be left to future work.

4.2 Methods and Their Fit

The specific methods addressed within this paper will now have their comparison to the LCIM explained. A brief mention of the levels that have a complete fit will be made, but more information about the analysis leading to a partial fit indication will be given.

4.2.1 Base Object Models

In the case of Base Object Models, full compliance at the semantic level (level 3 of the LCIM) is indicated, although this is only partly justified. It is in full compliance if one considers the use to which the documented standard indicates (i.e. – as a link between context-free conceptual models, and HLA elements) will be made. The HLA elements being described are well known, at least to the term level, within the federation they are employed. Therefore these elements are sufficiently described for semantic exchange. If the BOM template were to be expanded to other uses, however, then some sort of controlled vocabulary would be have to be employed for the conceptual models within each template, to ensure semantic alignment with other templates.

At the pragmatic level (level 4 of the LCIM), BOMs are evaluated to have a partial fit with the requirements. This is because the definition of the HLA elements (structure, scope, and resolution) would be well known within the federation they are used in. This is the part of the pragmatic level that fits. However, the requirement of the pragmatic level for an explicit depiction of context of application is not supported by BOMs. This is due to the definition of the BOM template as capturing a context-free model of the object being modeled – application specific context is not part of the template. This absence of application specific context is also what prohibits the BOM method from addressing the dynamic level (level 5 of the LCIM).

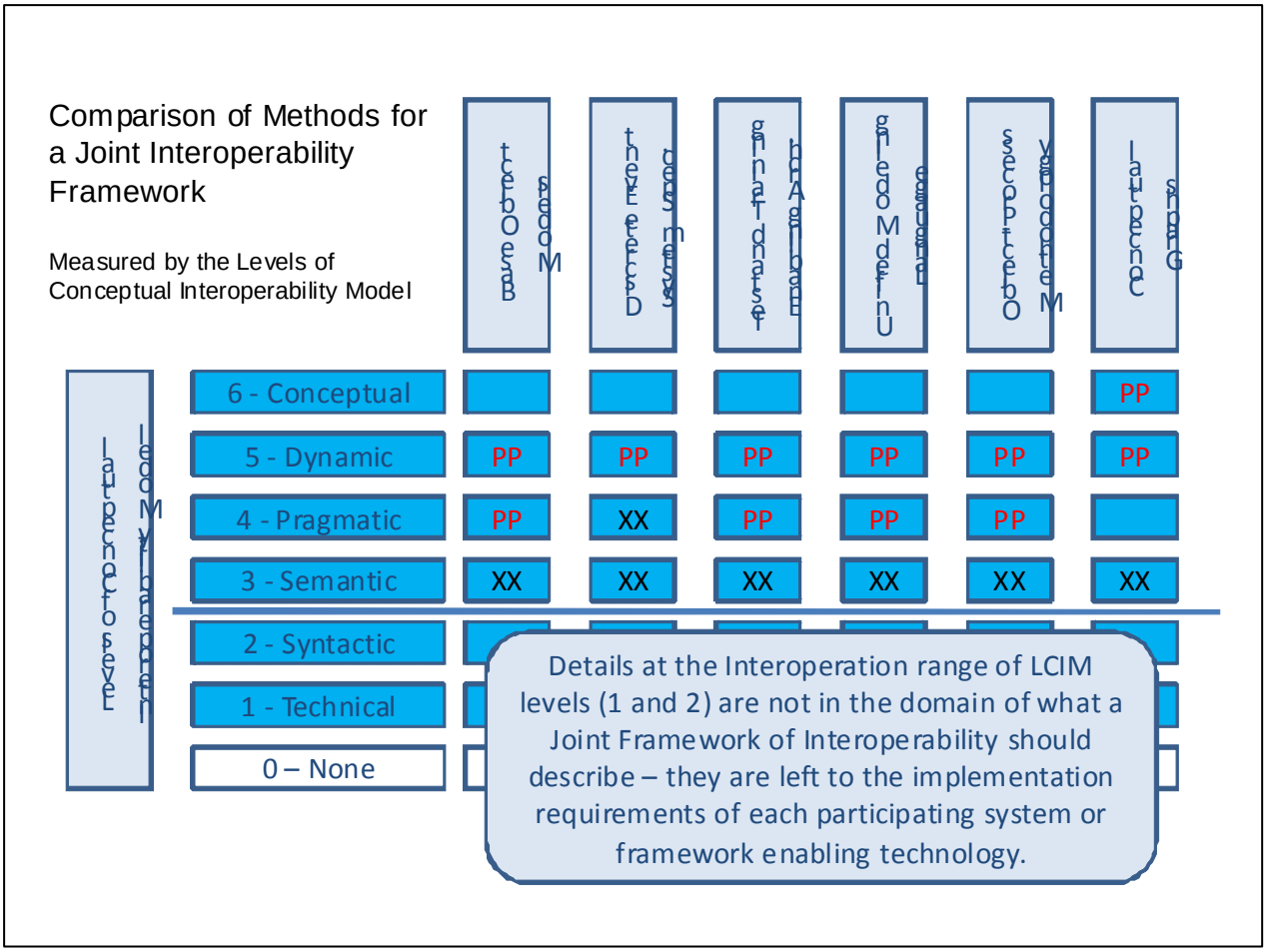

Figure 3 - Comparison of Methods to LCIM

4.2.2 Discrete Event System Specification

For systems described by the DEVS formal method, there are two basic approaches, that of the atomic

DEVS structure, and the coupled DEVS structure. As we are discussing distributed simulation systems (hence the need for interoperability), then the best employment of DEVS will be with coupled structures [9], or to rely on a DEVS specified model for model based engineering [10]. In that light, as all of the DEVS structures are defined in regards to each other (or some central model), then the semantic level (level 3 of the LCIM) is fully satisfied.

As the contextual interactions and definitions of structure, scope and resolution of the elements being described by the formalism are well known amongst all of the DEVS described components, the pragmatic level (level 4 of the LCIM) can be said to be satisfied. The ability of the DEVS specification to show the many different system states, and describe the object-interactions for those states, also shows an ability to support the dynamic level (level 5 of the LCIM). As with BOMs, which are intended to support a specific architecture (HLA), DEVS described components in a system of systems are intended to interoperate with other components that are also described using the DEVS formalism. To adopt DEVS to describe the components implies that all components must be described with DEVS, although this requirement may be true of all approaches (see section 5 below).

4.2.3 Test and Training Enabling Architecture

TENA is listed as being fully compliant with the semantic (level 3 of the LCIM), and partially compliant with the pragmatic and dynamic levels (levels 4 and 5 of the LCIM, respectively) of interoperability. It should be pointed out, while beginning this evaluation that this is in light of the intended use of TENA – that is, to have object models describing the different objects of interest to a TENA federation, and then exchanging that information over TENA middleware. It is possible that a method of translating a TENA object model into a different architecture could be accomplished. In which case the application specific context that comes from relying on all TENA OMs operating over the TENA middleware could likely be absent, prohibiting interoperability at the dynamic level, and severely limiting the fit at the pragmatic level.

If, however, TENA OMs are used as they are intended, then the semantic level is fully satisfied, as all OMs have a common base of terms from which to draw their semantic tagging. Additionally, the TENA middleware supports, fully, the operational context within which components of the OM will be used – thus partially satisfying the pragmatic level (level 4). Further, the dynamic level (level 5) is partially satisfied, when employing TENA OMs, as the changes to operational context (if existent) are well known among represented components of the architecture. What is lacking for both pragmatic and dynamic levels (and why they are both indicated as being only partially compliant) is the depiction of state-specific definition of elements and components (the structure, scope and resolution of such). It is likely that this is due to the immaturity of systems employing the architecture – such context specific definitions do not change – however this is the subject of future classification.

4.2.4 Unified Modeling Langauge

The various diagramming techniques of UML, as a means of directing interoperability between systems described in a Joint Composability Framework, are evaluated to be fully compliant with the semantic level (level 3 of the LCIM). This is because all of the elements and components are well described with informative labels, providing the semantic basis for addressing such elements and components.

Moving beyond the semantic level, while employing the UML method, proves to be somewhat problematic, given the requirements of the LCIM given in section 4.1. While specific application context is well defined, given a full suite of the diagramming techniques, what is missing is the definition of the terms that are semantically labeled. The definition (structure, scope and resolution) of the elements and components is not required by any of the diagramming techniques of UML, so such definition cannot be given (even though the context within which such definition would have pragmatic meaning). Since the definition of objects is not given, then the change to that definition over time (required by the dynamic level of the LCIM) cannot be described.

One of the advantages of UML, however, over the previous methods described (BOMs, DEVS, TENA) is that it can be an implementation neutral guide to applying some of the required artifacts that enable the other methods. BOMs and TENA each describe specific object models, to be implemented over specific architectures, and DEVS requires that all components be described using the same simulation-modeling [5] technique (although a specific architecture is not mandated in the case of DEVS). With UML, there are no implementation requirements, yet it may serve to augment the BOM, TENA or DEVS approaches.

4.2.5 Object Process Methodology

OPM is a diagrammatic approach, similar to UML, however it concentrates on showing (specifically) the relationship between objects (elements of the system) and processes (the means of change and activity for the elements of the system). Because of this, it addresses all of the components of the system in a meaningful

way, which allows for semantic identification of those components, meaning that an OPM described system would fully support the semantic level (level 3 of the LCIM).

The pragmatic level (level 4 of the LCIM) is identified as being partially satisfied by OPM (as with UML, TENA and BOMs). In the case of OPM what is missing is (not surprisingly) a different part of the whole. OPM describes, very well, the specific application context by making the relations between objects, as well as the contextual changes to those objects, clear in a well described manner. The specific changes to context are made clear, as are the specific objects affected by that change, and the framework for describing the specific changes to component and element definition are made (by specifically identifying processes) – but not enabled to the extent that they are made explicit within some of the other techniques (within the OMs of BOM or TENA for instance, or the component description of DEVS). Because of this lack of element and component definitional detail, the dynamic level (level 5 of the LCIM) can only be partially satisfied.

4.2.6 Conceptual Graphs

CGs are analyzed to provide a description of a system fully compliant with the semantic level (level 3 of the LCIM), in that they address all of the system concepts – elements, components, and relations – and identify them with a semantic element. CGs are uniquely (among the methods addressed here) identified as partially supporting levels 5 and 6, because they capture the conceptual meaning of the components of the system, but lack lower level (notably level 4, and part of level 5) support because of the lack of specificity within the representation.

5 The Way Ahead

Clearly, from the methods evaluated here, there is no one method that fully embraces the semantic, pragmatic, and dynamic levels of interoperability. In order to achieve composability, when defined in light of the LCIM, then dynamic interoperability (level 5 of the LCIM) needs to be supported between systems.

In order to achieve dynamic interoperability (or the first level of composability), especially if existing artifacts (OMs, architectures, etc.) are to be relied on, then a merger of existing methods must be attempted. Elements of the methods described here could be relied on. Both OM based architectures (HLA, as seen through the BOMs, and TENA) are very good as describing the definition of elements (part of what is needed at the pragmatic level). The DEVS formal method is very good at identifying the specific model based I/O between components. UML and OPM each give part of the picture of context and relations between elements, as well as indications of how that context changes over the operational span of the system. Merging the best of each of these techniques could result in a Joint Composability Framework that is capable of supporting actual composability (as defined by the LCIM – that is, dynamic interoperability).

A recognized barrier, to be addressed in future research, is the limits that the existing systems that will take part in the federations described here will bring to any interoperability asked of them. It is difficult to mandate dynamic interoperability (complete explicit expression of meaning and definition as it changes from state to state within a system), when such dynamic changes don't take place within the systems we are currently using. However, we will not always be using the same systems that we have today, and it is might do well to look towards the requirements for composing richer systems in the future.

6 References

[1] Lutz, R., Wallace, J., Bowers, A., Cutts, D., Gustavson, P., and Bizub, W., (2009). Common Object Model Components: A first Step Toward LVC Interopeability. *Proceedings of the 2009 Spring Interoperability Workshop (Spring SIW-09)*. San Diego, CA.

[2] US Office of the Secretary of Defense, (2004). Joint Battle Management Command and Control Roadmap, Washington, DC.

[3] Zeigler, B.P., Praehofer, H., and Kim, T.G., (2000). Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems, Second Edition: Academic Press.

[4] Zeigler, B.P., Ball, G., Cho, H., Lee, J.S., and Sarjoughian, H., (1999). Implementation of the DEVS Formalism over the HLA/RTI: Problems and Solutions. *Proceedings of the 1999 Spring Simulation Interoperability Workshop*.

[5] Yilmaz, L., (2004). On the Need for Contextualized Introspective Models to Improve Reuse and Composability of Defense Simulations. *The Journal of Defense Modeling and Simulation* 1:3, pp141-151.

[6] Sowa, J.F., (1984). Conceptual Structures: Information Processing in Mind and Machine, Addison Wesley, Reading, Mass.

[7] Lukose, D. & Mineau, G.W., (1998). A Comparative Study of Dynamic Conceptual Graphs. *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based Systems*

Workshop (KAW-98). B. Gaines & M. Musen (Eds). University of Calgary, Calgary, Alberta, Canada.

[8] Lee, J., Lai, L.F., (1998). Verifying task-based specifications in conceptual graphs. *Information and Software Technology*, 39:14-15, pp913-923, Elsvier Science.

[9] Touraille, L., Traore, M.K., Hill, D.R.C., (2009). On the Interoperability of DEVS Components, Research Report LIMOS/RR-09-04, ISIMA, France.

[10] Schmidt, D.C., (2006). Model-Driven Engineering - guest editor's introduction, *IEEE Computer*, February 2006 (Vol. 39, No. 2) pp. 25-31

[11] Noseworthy, J.R., (2008). The Test and Training Enabling Architecture (TENA) Supporting the Decentralized Development of Distributed Applications and LVC Simulations. *Proceedings of the 2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications*, IEEE Computer Society, Washington, DC, pp. 259-268

[12] Wallace, J., Ceranowicz, A., Powell, E., Lutz, R., Bowers, A., Bizub, W., Cutts, D., Gustavson, P., Rheinsmith, R., McCloud, T. (2009). Object Model Composability and Multi-Architecture LVC Interoperability. *Proceedings of Interservice/ Industry Training, Simulation, and Education Conference*, paper no. 9014, Orlando, FL

[14] Sowa, J.F., (1992). "Conceptual Graphs Summary," in Conceptual Structures: Current Research and Practice, P. Eklund, T. Nagle, J. Nagle, and L. Gerholz, eds., Ellis Horwood, 1992, pp. 3-52

Acknowledgments

The underlying research was partly supported by the JTEOW contract, under Mr. Warren Bizub. The goal of that work is to contribute to a Joint Common Object Model that can support interoperability of a wide field of systems, at a number of different levels of conceptual representation. The authors thank Mr. Bizub for providing this opportunity to support the project with this publication.

Authors' Biographies

CHARLES TURNITSA is a Senior Project Scientist at the Virginia Modeling Analysis and Simulation Center at Old Dominion University. In addition he is also a Ph.D. Candidate, studying under Dr. Andreas Tolk at ODU. He has a M.S. in Electrical and Computer Engineering from that institution.

ANDREAS TOLK is Associate Professor in the Faculty for Modeling, Simulation, and Visualization at the Engineering Management Department of the College of Engineering and Technology at Old Dominion University (ODU) of Norfolk, Virginia. He is affiliated with the Virginia Modeling Analysis & Simulation Center (VMASC). His domain of expertise is the integration of M&S functionality into real world applications based on open standards. He received a Ph.D. and an M.S in Computer Science from the University of the Federal Armed Forces in Munich, Germany.