Proceedings of the 2011 Winter Simulation Conference
S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, eds.

USING THE LEVELS OF CONCEPTUAL INTEROPERABILITY MODEL AND MODEL-BASED DATA ENGINEERING TO DEVELOP A MODULAR INTEROPERABILITY FRAMEWORK

Saikou Y. Diallo

Virginia Modeling Analysis & Simulation Center
Old Dominion University
Suffolk, VA 23435, USA

Andreas Tolk

Engineering Management & Systems Engineering
Old Dominion University
Norfolk, VA 23529, USA

Jason Graff
Anthony Barraco

GDIT
General Dynamics
Suffolk, VA 23435, USA

ABSTRACT

This paper describes how to use the Levels of Conceptual Interoperability (LCIM) as the theoretical backbone for developing and implementing an interoperability framework that supports the exchange of XML-based languages used by M&S systems across the web. The principles of Model-based Data Engineering (MBDE) are integrated within the framework to support the interactions between systems across the layers of the LCIM. We present a use case that shows how the framework supports the interoperability of heterogeneous military systems.

1 INTRODUCTION

Interoperability is understood as "the ability of two or more systems or components to exchange information and to use the information that has been exchanged" (IEEE 1990) and remains a great challenge in Modeling and Simulation (M&S) and related fields such as Systems Engineering and Software Engineering. One of the main reasons for interoperability is to support the reuse of existing solutions. However, M&S has the additional challenge of dealing with models which are simplifications of reality in order to answer a modeling question and simulations which are the execution of the model using a simulator. While the interoperability of simulators and simulations has been addressed to some extent, interoperability at the modeling level is starting to garner more attention as the community realizes that ignoring the problem might lead to the creation of a "Frankenstein model" made out of multiple models but without an identity of its own. In order to support interoperability at the modeling level a methodology or approach is needed to 1) understand to separation between model and simulation 2) distinguish between model and valid model and 3) separate simulation from simulator. In the current state of the of art the distinction between the three is theoretical and is not reflected in current interoperability standards and frameworks.

In this paper, we will first distinguish between a model, a language, a simulation and a simulator and show how a language can be used to support the interoperability of models and simulations. We will use the language to derive requirements for an interoperability framework. The LCIM first introduced by Tolk and Muguira (2003) was designed to measure or prescribe the level of interoperability between models and simulation. Model-based Data Engineering (MBDE) (Tolk and Diallo 2003) is an engineering

methodology designed to make systems interoperable. We integrate the LCIM with MBDE to generate a framework that meets the requirements identified earlier and describe an implementation of this framework that supports the interoperability of XML-based languages.

2 MODEL, LANGUAGE, SIMULATION AND SIMULATOR

In order to differentiate between a model a language a simulation and a simulator, we need an apparatus from which we can study these terms and show how they are different and/or related. Model Theory, a branch of mathematics that is focused on the study of objects and their structures using logic provides a unifying approach that can be used in M&S. The following definitions are taken from Weiss and D'Mello (1997).

Definition 1 *A language L is a set consisting of all the logical symbols with perhaps some constant, function and/or relational symbols included*

Definition 2: *A model (or structure) U for a language L is an ordered pair <A, I> where A is a non-empty set and I is an interpretation function with domain the set of all constant, function and relation symbols of L such that a constant symbol is mapped to a constant, a function symbol is mapped to a function and a relation is mapped to a relation*

Definition 3: *A sentence is an assertion that can be assigned the Boolean value of true or false*

Definition 4: *If U is a model of L, the theory of U, denoted ThU, is defined to be the set of all sentences of L which are true in U*

Definition 5: *A finite state machine is a triple {I, S,O} where I is the set of inputs, S is the set of states and O is the set of outputs*

While these definitions originate from model theory, they are useful in M&S in understanding the relationship between model, simulator and simulation. We define these terms in M&S as follows:

Definition 6 *A model in M&S is a language L*

Definition 7 *A simulator is a finite state machine*

Definition 8 *A simulation is the generation of a model by a simulator*

Definition 9 *A referent in M&S is a structure (model in model theory)*

Definition 10 *A model in M&S is valid if and only if it is a Theory of a referent. The model is said to valid under the referent or valid with respect to the referent*

There are several key observations that emerge from using Model theory as the basis for understanding the relationships between a model, a language, a simulator and simulation:

- The language or model in M&S is the common denominator between all of the components of interoperability. In that sense, we can truly say that everything in M&S is a model including the referent (the model of the model). This is a very powerful observation because it allows us to unify interoperability under the modeling umbrella and treat interoperability challenges as modeling challenges.
- The simulator is a model generator. Using Model Theory, we can separate the interoperability of model generators from the interoperability of that which they generate i.e. the model and provide a simple definition for the interoperability of simulators. Further, we can generalize Definition 8 as follows:

Definition 11 *A simulation is the generation of a model by one or more simulators.*

The simulators can be acting in series, parallel or any configuration without affecting the definitions provided. Finally, the definition not only provides a separation between the simulator and the model it also clearly specifies what parts of the simulator are involved in the simulation i.e. the simulator is the part that generates the sentences on the model. Consequently at the practical

level any configuration, software, hardware and operating system that is not directly generating the simulation is not taken into consideration. This means that transport mechanisms, synchronization methods and translation tools are not part of the simulation unless they generate some part of the model. The only thing that generates the model is the finite state machine realization of the model or the finite state equivalent of the language if one exists. This is a very restrictive view of a simulation but one that is formal and very useful in separating technical issues from modeling issues cleanly. Otherwise stated, technical interoperability (the exchange of bits and bytes) is divorced from M&S in the sense that it does not contribute in producing a model.

- Validity and satisfiability: One of the most challenging issues in building an interoperability framework is distinguishing between models and valid models especially if one assumes that validated (in the traditional VV&A meaning) models will cohabitate with models that are not validated. The question is whether validated models become invalid during interoperability and more interestingly whether models that are not validated are assumed to be valid especially when they exhibit apparently valid behavior. This question is rarely answered and in fact federations of models seldom go through the same validation process as individual models. This, of course, is due to the lack of a formal definition of validity and ultimately the lack of a formal theory of validity that would explain this phenomenon. In our case, we assimilate validity with satisfiability simply because it generalizes the idea of validity and formalizes it. It highlights the key aspect of validity as being with respect to something (the referent in M&S) and captures the idea of partial validity by stating that only the parts of the model which are true under that referent are valid. Finally, Definition 10 explains that reuse is the number of theories that a model can generate.

The distinction between model and simulation provided here align very well with the DEVS formalism with the difference that DEVS only deals with finite state machines. Once we have separated the key concepts and established interoperability as a modeling problem excluding the simulator parts as we described earlier, we can safely state that fundamentally a language is necessary and sufficient to ensure interoperability. We can rewrite the interoperability definition as the ability of systems to generate a language. It is important to note that the language need not be valid. We will not delve into the Model Theoretic aspects of this discussion in this paper, but we can use the definitions and findings of Model Theory as applied to M&S to derive the requirements for an interoperability framework.

3 REQUIREMENTS OF AN INTEROPERABILITY FRAMEWORK

An interoperability framework as envisioned in this paper will require models to be developed in the future with interoperability as a basic requirement. However, the framework should also be able to accommodate legacy models among other requirements. In this section we will focus on the specific requirements necessary to support the interoperability of models. Based on the definitions provided in the previous section, it is apparent that in theory a referent can have an infinite number of models which can be generated by an infinite number of simulators which in turn results in an infinite number of simulations. In practice, the number of models of the same referent (combat models in general, tank models, soldier models, etc.) is not quite infinite but still is very large. The same holds true for the number of implementations of these models and the number of ways one can calibrate a given simulator to generate a simulation. Since, this number is very large, any framework that supports particular cases of models will lack the flexibility required to support a rapidly changing environment where doctrine, tactics and procedures are constantly changing which means the number and type of models as well as the interoperability requirements are continuously evolving and changing. This is especially true for standard frameworks such as the Distributed Interactive Simulation (DIS) (IEEE 1998) and the High Level Architecture (HLA) (IEEE 2000) which are now moving into supporting Live, Virtual, Constructive (LCV) requirements (Henninger et al. 2008). By their very nature, these standards do not separate models, simulations, referents and languages. In order to be flexible and adaptive an interoperability framework should be able to:

- Separate referent from model
- Separate referent from valid model
- Separate referent from simulation
- Separate referent from simulator
- Separate model from valid model
- Separate model from simulation
- Separate model from simulator
- Separate simulation from simulator

In addition to these basis requirements, the interoperability framework should have the capability to support the high level requirements shown in Table 1. Each requirements is motivated by a definition or set of definition. The goal or desired feature of the framework is also described as it relates to the capabilities enumerated in the list above. In the military context, the ultimate goal is to generate a framework that can support multiple combat models implementing different aspects of the battlefield (logistics, artillery, etc.) at multiple echelons. The framework should support multiple implementation platforms and message formats including current military messages. The framework should support joint and coalition systems and enable coalition interoperability at the strategic and tactical levels. It should have the flexibility to integrate and support new message types such as the exchange of orders, reports and requests. In terms of interoperability, the proposed framework is network agnostic and therefore can be implemented at multiple levels of security

Table 1: Requirements for an interoperability framework

Requirements	*Origin*	*Explanation*	*Goal*
Support multiple referents	Definition 9	Allow multiple current and future models and simulations to be interoperable	Model agnostic
Support multiple models	Definition 6	Allow multiple languages/dialects to be supported	Language agnostic
Support multiple valid models	Definitions 6, 9, 10	Allow multiple existing/legacy models to be supported	Theory agnostic
Support multiple simulations	Definitions 8,9,10	Allow multiple simulations to coexist in series or parallel	Simulation agnostic
Support multiple simulators	Definition 11	Allow multiple implementations to be interoperable	Implementation agnostic

Having described the requirements for the framework, we need further guidance on what elements are needed to capture it. In order to accomplish this goal, we introduce a formal specification of the LCIM as a guide for describing interoperability and MBDE as a guide on how to accomplish interoperability.

4 INTEGRATING THE LCIM AND MBDE INTO THE FRAMEWORK

The level of conceptual interoperability model (LCIM) was introduced by Tolk and Muguira (2003) in order to establish the degree to which two or more systems interoperate. The latest version of the LCIM as presented by Turnitsa (2005) has seven levels where:

- Level 0: The systems are not connected, no interoperability
- Level 1: The systems can exchange bits and bytes. A physical connection based on a communication protocol is established. The systems are on the level of technical interoperability.
- Level 2: The systems share a common data format and agree on a common syntax. At the level of syntactic interoperability, the bit and bytes exchanged can be grouped to form symbols. At this level, systems share a common reference physical data model instance.
- Level 3: The systems harmonize the meaning of the symbols they exchange. The level of semantic interoperability implies agreement on the definition of terms through a process of disambiguation. The systems share a common reference physical model.
- Level 4: The systems are aware of the context in which the symbols they exchange are used. At this level the systems are aware of all the possible groupings of symbols and how they are related. The level of pragmatic interoperability implies the awareness and sharing of a common reference logical model.
- Level 5: The systems understand the processes that will use the symbols they exchange. At the level of *dynamic interoperability* the assumptions and constraints of processes are described unambiguously and the behavior of systems is predictable during interoperation.
- Level 6: The underlying concepts represented by the symbols are described unambiguously. The level of *conceptual interoperability* implies the alignment of the models represented in systems. The systems share a common reference conceptual model that captures the assumptions and constraints of the corresponding real or imaginary object.

Despite the LCIM being designed for establishing what interoperability is through a categorization, it does not provide 'how' interoperability can be achieved. Considering Model Theory, the LCIM assists us into establishing *how* to achieve interoperability:

- Level 1 by sharing theories under a well-defined and accepted computer model (see definition 10), for instance, TCP/IP or HTTP protocols;
- Level 2 by sharing common interpretation functions (see definition 2) on *A* (constants, functions and relations);
- Level 3 by sharing structures (see definition 2), despite the potential of generating different languages;
- Level 4 by sharing theories in order to share contexts;
- Level 5 by sharing the same sentences (see definition 3) in the same order in order to fulfill synchronicity; and
- Level 6 by the equivalence between models and languages.

MBDE is an engineering process that focuses on capturing data exchange requirements in a federation including its availability, the agreed meaning, its groupings, and the consistency of the groupings within a common reference model. To do so, it is required to know the purpose of the federation in order to identify candidate systems that can be federated and fulfill that purpose. MBDE contains four processes: data administration, data management, data alignment and data transformation.

- Data Administration focuses on the identification of data formats, location, and domain in order to establish unambiguous definitions and classification of entities, properties, values, and metadata.
- Data Management focuses on the identification of logical relations among entities, properties, and values in order to establish the rules to form meaningful groupings. This step is important in that seeks to capture the most mandated properties of all potentially associated entities in order to have strong groupings.
- Data Alignment focuses on the identification of scope (number of unique groupings) and resolution of data (cardinality of the groupings). This is important in order to identify mismatches of scope and resolution between two systems.
- Data Transformation focuses on establishing consistency by identifying and mapping functions that generate valid sentences in a computable model. It is important because it serves as the basis for an implementation.

MDBE helps us establish interoperability of systems and like the LCIM, we can use Model Theory to better formulate that process:

- Data Administration focuses on the identification of the universe A;
- Data Management focuses on the identification of the interpretation function on A;
- Data Alignment focuses on the identification of structures;
- Data Transformation focuses on the identification of language, model and theories.

Unlike the LCIM, MBDE was designed to achieve interoperability but both are assuming that interoperability is engineered and neither makes the distinction between model, valid model, simulation and simulator. The LCIM offers the framework a way to measure interoperability and MBDE provides an approach to achieve that level of the LCIM. MBDE has to be done for every system in order to generate a common language from the bottom up. The idea is that we know in advance what systems we would like to see integrated and we have a scenario and overall model we want to generate. MBDE is used to identify what each systems can produce and what it needs to know. The LCIM is then used to align the models on all levels. A top-down approach is also possible by mandating a common language for all systems and using MBDE as a filtering mechanism to identify the parts of the language that a system can understand. In the next section will describe an implementation of this framework for XML based Languages involving military models that supports both approaches.

5 AN OVERVIEW OF CBMS

The Coalition Battle Management Service (CBMS) is a technical infrastructure that enables the exchange of resources (orders, reports, and requests) between Command and Control (C2) systems, simulation systems and robotic forces. CBMS is a collection of composable web services that can be orchestrated to support the needs of a particular federation. CBMS is currently implemented as a service oriented architecture with an interrupt mechanism, a filtering mechanism and a data distribution mechanism that can be used to support the validation, storage, search and exchange of XML based languages. These languages include but are not limited to the Coalition Battle Management Language (C-BML) (Blais, Galvin and Hieb 2005) and the Military Scenario Definition Language (MSDL) (SISO 2008). CBMS is accessible via any commercially available web browser and uses only next generation XML based technologies in its implementation. In this case, in order to make XML based language standards interoperable, we focus on *what* was said on a message and not on *how* it was said. Hence, CBMS can load a scenario using MSDL and send messages to a simulation from a C2 system using C-BML.

CBMS is architected to take advantage of web principles and technologies because the web as we use it today is the most interoperable environment that replicates the M&S domain. Similarly to computer systems and users on the web, M&S interoperability aims to connect users and systems from heterogene-

ous environments in a parallel and distributed fashion. CBMS is system and environment neutral and follows the principles of a Service Oriented Architecture (SOA) where CBMS provides basic common services (transport, storage, search, filtering) and each system that connects to it becomes a service that is available to any other user or system.

CBMS provides a paradigm shift in interoperability framework by taking the emphasis away from the traditional interface design and alignment approach that often results in rigid solutions. Interface based interoperability requires gateways and bridges to be built ad infinitum in order to foster reuse because the interface is tied to a system or a federation of systems. CBMS places the emphasis on the language and thus creates a document centric paradigm where systems exchange a self contained document that they are free to mine based on their needs and capabilities. Going back to the example of the web, this is very similar to many users accessing a web page where each user is free to read the content that they are interested in, navigate to the pages that they think are relevant all at the same time. Within this paradigm, it is acceptable that some users will not understand the language of the web site (Urdu, English, Pashto), some will understand the language and find the stories relevant and some will not think the stories relevant at all. The same principles are at play within CBMS.

Figure 1 shows an overview of CBMS and its main components namely:

- A transport mechanism: The transport mechanism used in CBMS is the Hyper Text Transport Protocol (HTTP) which is used to transport hypermedia in a distributed environment and collaborative environment. HTTP is ubiquitous across the web and thus provides us a solid and reliable basis for technical interoperability;
- A storage mechanism: Instead of providing a Relational Database Management System (RDBMS) as is traditionally the case, CBMS stores every document in its native XML format. The only requirement is that the document be syntactically valid which respect to a schema. The storage mechanism allows us to support syntactic interoperability between documents that share a common structure. The storage mechanism is designed to support document searching and after action review (AAR) in the case of military scenarios.
- An artificial intelligence component: This component is designed to facilitate the move towards semantic interoperability by providing support for ontology, reasoning and the design of domain specific languages. This component allows the specification of a common interpretation of terms and structures of a language.

In terms of a bottom up approach CBMS can support the interoperability of systems that natively speak language such as MSDL or C-BML. It can also support a language that is generated through MBDE in order to support specific systems in a given federation. For instance , CBMS can support the exchange of Link messages if they are expressed in XML and at the same time support the integration of Google Maps to provide a common operational picture through the Keyhole Markup Language (KML).

In Figure 2, we show how CBMS further borrows from the principles of the web. CBMS uses the Representational State Transfer (REST) which constrains information exchange to the ability to get a message, put a message, post a message and delete a message. The RESTful implementation allows CBMS to be interface and content agnostic. In addition, CBMS uses Server Sent Events to allows systems and users to predefine content of interest which is delivered as soon as it is available. In terms of military systems, CBMS allows systems to filter out content that is not of interest (for instance events that occur outside the area of operation that are not of immediate interest) and conversely to only share information that is relevant to other users (this is a generalization of the publish subscribe paradigm). Because it uses HTTP, CBMS can easily support web applications, browsers and any HTTP compatible device.

Table 2 shows how the CBMS architecture implements the interoperability framework described in this paper in order to support C2 and simulation interoperability:

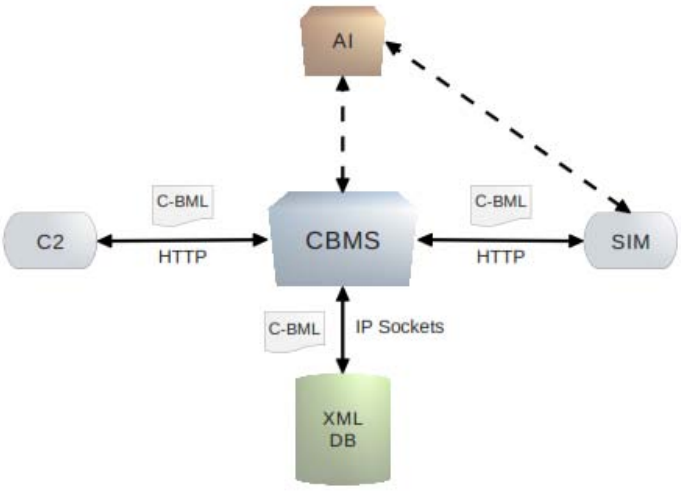

Figure 1: CBMS component integration

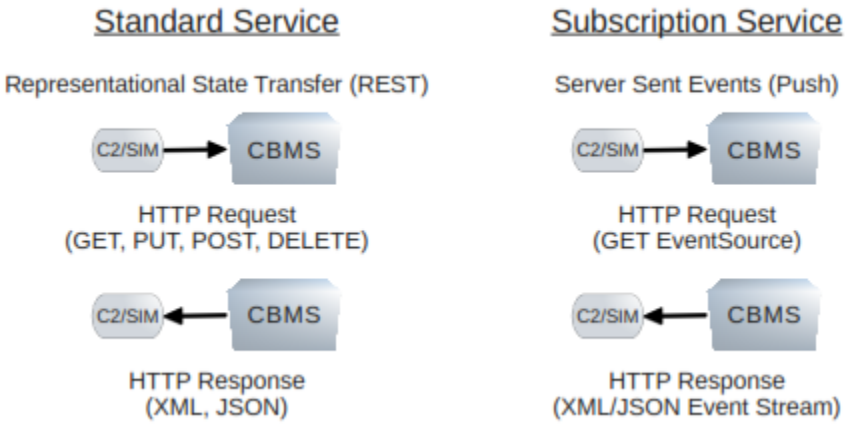

Figure 2: CBMS implementation overview

The CBMS framework focuses on the exchange of languages between systems. This language is contained within the document that is being exchanged. Each system that is connected to CBMS represents a referent (user) or a model that can generate and/or consume a language. If two or more systems are syntactically interoperable it means that they speak the same language and CBMS can support the exchange of this language. If a language if a theory of this model it is said to be valid for this model or the models are semantically interoperable. A s simple example let us assume that there are several tank models that want to exchange information about their respective positions. Each tank can broadcast its position is latitude and longitude which makes them syntactically interoperable. If they, in addition agree on a common reference system and the same definition for latitude and longitude, they share a common theory and we can say that latitude and longitude are theories of positions for the tanks. CBMS can then be used to exchange the position information with the added capability of filtering the information to the position of a tank of interest i.e. it is not necessary to get information on all tank positions. Further, if a tank wants to get the position update only between two phase lines and at a given time or time interval CBMS the capability to support such an exchange.

Table 2: CBMS and the interoperability framework

Requirements	*CBMS service*	*Explanation*	*Goal*
Support multiple referents	Transport mechanism, subscription	Allows any referent to send receive or communicate	Model agnostic
Support multiple models	Syntactic validation, Subscription	Allow multiple languages/dialects to be supported	Language agnostic
Support multiple valid models	Transport mechanism	Allow multiple existing/legacy models to be supported	Theory agnostic
Support multiple simulations	Transport mechanism, Storage	Allow multiple simulations to coexist in series or parallel	Simulation agnostic
Support multiple simulators	Subscription, Storage	Allow multiple implementations to be interoperable	Implementation agnostic

CBMS provides syntactic validity but does not take a position on the semantics of a language or the validity of a model. In other words, CBMS abides by the eight rules formulated in section 3 which gives it the ability to be flexible i.e. any model can connect to the framework and speak any language at any point in time. It is also adaptive because it supports proprietary languages, standard languages and future languages regardless of the domain and application area. The only requirement is that the language is structured. Let us examine in details how CBMS deals with the high level requirements we expressed in section 3:

- Separate referent from model: The model in CBMS is represented by the language and the referent is the object that the language refers to. For instance the string "tank" can be used to refer to a tank. In CBMS, the tank model is a service that can use CBMS to talk about tanks;
- Separate referent from valid model. In CBMS, the language is not assumed to be intrinsically valid. In keeping with our example, the sentences about tanks are neither true nor false. This evaluation is done with respect to a referent;
- Separate referent from simulation: CBMS does not assume that a simulation is valid with respect to a referent. In keeping with our example, if a tank receives an order to move, CBMS does not assume that the tank will in fact move as intended or in a way consistent with the referent. The role of CBMS is to make sure that the right tank received the order;
- Separate referent from simulator: CBMS does not assume a particular implementation and is not interface driven. Staying with the tank example, a tank' s movement can be generated by a random number generator, a probability density function, a differential equation etc;
- Separate model from valid model: CBMS does not assume a given language is a theory of a referent. This is in keeping with rules 1) and 2). Simply stated, some sentences about tanks might not be true in some models of tanks or some sentences about tanks have no meaning (neither true nor false) in some models of tanks;

- Separate model from simulation: CBMS does not assume a simulation is generating a valid model. In other words, CBMS does not take position as to whether the position of a tank at given moment is correct or is generated using a correct algorithm.
- Separate model from simulator: CBMS does not assume that a model can be generated by only one simulator or a simulator can generate only one model. For instance, we do not assume that because there are sentences about a tank there is only one algorithm that can can generate such a tank;
- Separate simulation from simulator: Finally, CBMS does not assume that every model can be simulated and every simulator can simulate a model. In other words, there are simulators that cannot generate certain tank behaviors even though there exist a set of sentences describing that behavior.

6 CONCLUSION

In this paper we presented the need to separate the notions of model, simulation, valid model and simulator in order to better understand how to make models interoperable. We provided definitions for these terms using Model Theory and showed that interoperability is theory generation. We use these definitions to define an interoperability framework and provide a brief description of CBMS which is an implementation of this framework in support of C2 to simulation interoperability. CBMS supports current messaging formats such as USMTF (at least the parts that have an XML equivalent) or the link family of messages as well as HLA and other existing standards. Emerging standards such as C-BML and MSDL are natively XML based and this trend will continue in the future . In the future, models have to be developed to be interoperable. Frameworks such as the one proposed in this paper will be extremely useful when models are expressed in standard and/or proprietary languages and are actually fully expressive instead of providing limited interfaces.

REFERENCES

Blais C., K. Galvin and M. Hieb. 2005. “Coalition Battle Management Language (C-BML) study group report.” In *Proceedings of the IEEE Fall Simulation Interoperability Workshop.* IEEE CS Press.

Fielding, R., J. Gettys, J. Mogul, H. Nielsen, L. Masinter, P. Leach, and R. Berners-Lee. 1999. RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1. http://tools.ietf.org/html/rfc2616.

Henninger, A., D. Cutts, M. Loper, R. Lutz, R. Richbourg, R. Saunders and S. Swensin. 2008. “Live Virtual Constructive Architecture Roadmap (LVCAF) Final Report.” DoD Office of Security Review (Case No. 09-S-2412)/ M&S CO Project No. 06OC-TR-001.

IEEE. 1990. “A Compilation of IEEE Standard Computer Glossaries”. New York: IEEE Press.

IEEE. 1998. “Standard for Distributed Interactive Simulation”. IEEE Std 1278-1998.

IEEE. 2000. “Standard for modeling and simulation (M&S) high level architecture (HLA)–framework and rules.” IEEE Std 1516-2000.

SISO. 2008. “Simulation Interoperability Standards Organization (SISO) Standard for: Military Scenario Definition Language” SISO-STD-007-2008.

Tolk A. and S. Diallo. 2005. “Model-based data engineering for web services.” *IEEE Internet Computing.* 9(4): 65–70

Tolk, A. and J. Muguira. 2003. “The Levels of Conceptual Interoperability Model (LCIM).” In *Proceedings of IEEE Fall Simulation Interoperability Workshop*. IEEE CS Press

Turnitsa, C. 2005)\. “Extending the Levels of Conceptual Interoperability Model.” In *Proceedings of IEEE Summer Computer Simulation Conference.* IEEE CS Press

Weiss, W. and C. D'Mello. 1997. Fundamentals of Model Theory. University of Toronto, Toronto, ON.

AUTHOR BIOGRAPHIES

SAIKOU Y. DIALLO is Research Assistant Professor at the Virginia Modeling, Analysis and Simulation Center at Old Dominion University. He received his M.S and Ph.D. in Modeling and Simulation from Old Dominion University. His email address is <sdiallo@odu.edu>.

ANDREAS TOLK is Professor of Engineering Management and Systems Engineering at Old Dominion University. He is also affiliated with the Virginia Modeling Analysis and Simulation Center. He holds a M.S. and Ph.D. in Computer Science from the University of the Federal Armed Forces in Munich, Germany. His email address is <atolk@odu.edu>.

JASON GRAFF is an engineer with General Dynamics. He has worked as a technology consultant for the past two decades in the defense, education, electronics, gaming, government, insurance, online retail and gas/oil industries. Jason holds a B.S. in Management Information Systems from the University of Wisconsin. His e-mail address is <jason.graff@gdit.com>.

ANTHONY BARRACO is a Software Engineer for General Dynamics currently responsible for the design, code and documentation of projects which include the Coalition Battle Management System (CBMS) and the Network Effects Emulation System (NE2S). He has seven years professional experience, specializing in object oriented programming. He received his B.S. in Computer Science from the University of Central Florida. His e-mail address is <anthony.barraco@gdit.com>.