Flight and Ground Vehicle Simulation Course

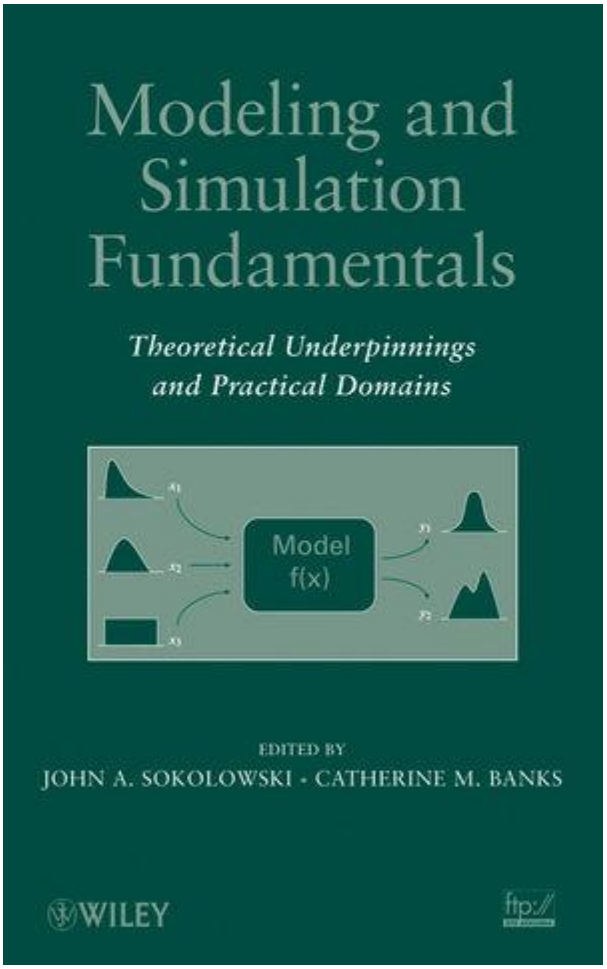
Distributed Simulation/ High Level Architecture Overview: Engineering Principles of Combat Modeling and Distributed Simulation

Dr. Andreas Tolk
andreas.tolk@gmail.com

Chapter 12

Interoperability and Composability

Andreas Tolk



Modeling and Simulation Fundamentals

Theoretical Underpinnings and Practical Domains

EDITED BY

JOHN A. SOKOLOWSKI • CATHERINE M. BANKS

WILEY

ftp://

Modeling and Simulation Fundamentals: Theoretical Underpinnings and Practical Domains
John A. Sokolowski, Catherine M. Banks
ISBN: 978-0-470-48674-0

Hardcover
456 pages
April 2010

Extracted from Tolk A: “Interoperability and Composability,” in Sokolowski JA, Banks CM: *Modeling and Simulation Fundamentals: Theoretical Underpinnings and Practical Domain, pp.403-433*
© John Wiley 2010

The following pages are extractions from the chapter 12 on “Interoperability and Composability” by Andreas Tolk, prepared for the John Wiley book *“Modeling and Simulation Fundamentals: Theoretical Underpinnings and Practical Domains,”* edited by John A. Sokolowski and Catherine M. Banks.

Introduction

For many modeling and simulation (M&S) developers, questions regarding the future interoperability and composability of their solution are not the main concern during design and development. They design their M&S system or application to solve a special problem and provide a solution. There is nothing wrong with this perception. However, there are many reasons why it is preferable to design interoperability and composability from the early phases on, e.g., by using open standards for the communication of information or by using standardized interfaces to common services. The main driving factor for this is the wish to enable the *reuse of existing solutions*. Why should we invest something into rewriting a solution that already exists?

The second aspect is that of *modularity*, in particular when dealing with complex systems. While it may be possible to use and evaluate small models as a whole, large and complex system rapidly become too big to be handled in one block. Development and testing for such systems should be conducted in modules; however, these modules need to be interoperable and composable to allow bringing them back into a common system.

The aspect of *reducing costs* is also playing a significant role. The idea is to reduce development cost by reducing reliable solutions and avoiding to “reinvent the wheel” in new models. However, again this assumes that the components can be identified, selected, composed, and orchestrated.

In addition, the growing connectivity of real world problems is reflected in the requirement to *compose cross-domain solutions* as well. Examples are, among others, the evaluation of interdependencies between the transportation systems and possible energy support in the domain of homeland security, or the analysis and support of common operations of several nations with several branches of their armed forces hand in hand with non-military and often even non-government organizations for the organizations like the North Atlantic Treaty Organization (NATO) or the European Defense Agency (EDA). Other examples from medical simulation can easily be derived for biological and medical simulations, where similar problems are observed when composing models on the enzyme, cell, or organism level with each other. The common challenge of these compositions is that such joint operations are more than just the parallel execution of part solutions. Synergistic effects need to be taken into consideration, as the whole new operation is likely to be more than just the sum of its part solution.

The growing connectivity requiring interoperable and composable parts is also reflected in the ideas of service-oriented architectures (SOA) and system of systems. In both cases solutions are composed on-the-fly reusing preexisting services that provide the required functionality. While traditionally engineers conduct the evaluation and adaptation of existing solutions to make them fit for a new environment, these engineering tasks need now to be conducted by machines, such as intelligent agents. This requires that all information needed to allow for

- the identification of applicable services,
- the selection of the best subset for the given task,
- the composition of these services to produce the solution and
- the orchestration of their execution

must be provided in machine-readable form as annotations. Consequently, services and systems must be annotated with information on their interoperability and composability characteristics to allow and enable their composition on-the-fly.

Finally, interoperability and composability challenges are not limited to M&S applications and services. Many M&S application areas as defined earlier in this book require the interoperation of M&S systems and operational infrastructure, such as traffic information systems and evacuation models, or military command and control systems and combat simulation systems.

Extracted from Tolk A: "Interoperability and Composability," in Sokolowski JA, Banks CM: *Modeling and Simulation Fundamentals: Theoretical Underpinnings and Practical Domain, pp.403-433*
© John Wiley 2010

This chapter will focus on the technical challenges of interoperability and composability, current proposed standardized solutions, and ongoing related research. It will not deal with business models supporting the idea. It will also leave out security aspects (you don't want your opponent or completion to use your best tools for his solutions) and questions of intellectual property out. These are valuable research fields on there own.

…

Students and scholars of the topics of interoperability and composability are highly encouraged to use this chapter as a first step towards underlying ideas and methods. It has implications for nearly all domains captured in this book, in particular for distributed simulation development and validation, verification, and accreditation (VV&A). However, it also implies new views on conceptual modeling beyond establish needs as well as the need for extended annotations of M&S services in service-oriented architectures. It also implies the need for new M&S standards as current solution are to implementation focused. We will focus on these issues in the appropriate paragraphs of this chapter.

Defining Interoperability and Composability

It is good practice to start discussions on the need for unambiguous definitions with respective definitions of terms that are used. We will start with the more traditional definitions used by IEEE and other organizations before looking at ongoing research on layered models of interoperations that are applied to improve the community understanding of what interoperability and composability are and how they can be reached.

Selected Interoperability Definitions

IEEE defines interoperability as the ability of two or more systems or components to exchange information and to use the information that has been exchanged [1]. This simple definition has already a number of implications:

- Interoperability is defined between two or more systems. As such, it includes peer-to-peer solutions as well as hub solutions.
- Interoperability allows systems to exchange information. This means that systems must be able to produce the required information as well as to consume the provided information. In particular when information is encapsulated, this may be challenging, which explains that it is necessary to take interoperability requirements into account early enough, so that the design does not hide information from accessibility.
- Interoperability allows systems to use the information in the receiving system. This implies some common understanding that is shared between sender and receiver. If a system "just listens" to provided information but ignores everything it cannot use, this is not an interoperable solution.

Other organizations, like the Software Engineering Institute (SEI) of Carnegie Mellon, are stricter in their definition. In [2], SEI defines interoperability as *the capability of two or more components or component implementations to interact*. The notion of taking action based on the received information is the new element in this view.

The U.S. Department of Defense adds the component of efficiency to the collaboration and defines interoperability as *the ability of systems, units, or forces to provide data, information, materiel, and services to and accept the same from other systems, units, or forces, and to use the data, information, materiel, and services so exchanged to enable them to operate effectively together* [3]. The same directive furthermore states that joint concepts and integrated architectures shall be used to characterize the interoperations of interest.

In summary, interoperability is understood as the ability of systems to effectively collaborating together on the implementation level to reach a general common objective. To this end, they exchange information that both sides understand well enough to make use of it in the receiving system. Interoperability is a characteristic of a group of systems.

Selected Composability Definitions

In the M&S community, the term composability is also used to address similar issues. Petty and Weisel documented various definitions in [4]. Examples for definitions of composability are the following.

Harkrider and Lunceford define composability as *the ability to create, configure, initialize, test, and validate an exercise by logically assembling a unique simulation execution from a pool of reusable system components in order to meet a specific set of objectives* [5]. They introduce the aspect of logically assembling and – as such – emphasize the necessary for a common basis for the conceptual models that describe the underlying logic.

Pratt, Ragusa, and von der Lippe approach the challenge of composability from the common architecture perspective. They define it as *the ability to build new things from existing pieces* [6]. These existing pieces, however, are components of a common architecture, or at least can be captured in a common architecture framework.

Kasputis and Ng emphasize the simulation view. They define composability as *the ability to compose models/modules across a variety of application domains, levels of resolution and time scales* [7].

In their work, Petty and Weisel recommend the following definition: *Composability is the capability to select and assemble simulation components in various combinations into valid simulation systems to satisfy specific user requirements* [4]. They also observe that composability deals with the composition of M&S applications using components that exist in the community (e.g., in a common repository). The composition is driven by requirements defining the intended use of the desired composition. Their definition became a common basis of composability research within the community. Composability therefore resides in the models, dealing with the conceptualizations and how they can support a set of requirements.

In comparison, interoperability is seen as the broader, technical principle of interacting systems based on information exchange while composability deals with the selection and composition of preexisting domain solutions to fulfill user requirements. This idea to distinguish between interoperability of implementation or simulation systems and composability of conceptualizations or simulation models is also the result of current layered approaches.

Towards a Layered Model of Interoperation

Several researchers introduced layered models to better understand the theoretical underpinnings of interoperation, not only in M&S. Computer science has a tradition of using layered models to better understand the concepts underlying successful interoperation on the implementation level. One of the better known examples is the International Organization for Standards (ISO)/Open System Interconnect (OSI) reference model that introduced seven layers of interconnection, each with well defined protocols and responsibilities [8]. This section uses the idea of introducing a reference model with well defined layers of interoperation to better deal with challenges of interoperability of simulation systems and composability of simulation models.

Dahmann introduced the idea of distinguishing between substantive and technical interoperability [9]. In her presentation, technical interoperability ensures connectivity and distributed computation while substantive interoperability ensures the effective collaboration of the simulation systems contributing to the common goal.

Petty built on this idea in his lectures and short courses [10]. He explicitly distinguishes between the implemented model representing substantive interoperability and layers for protocols, the communication layers, and hardware representing technical interoperability.

Tolk and Muguira [11] introduced the first version of a layered model for substantive interoperability, which was very data centric. In this first model, they distinguished between system specific data, documented data, aligned static data, aligned dynamic data, and harmonized data. These categories describe gradual improvements of interoperability and composition. While system specific data result in independent systems with proprietary interfaces, documented data allow for ad-hoc peer-to-peer federations. If these data follow a common model, they are statically aligned and allow for easier collaboration. If their use in the systems is also understood, the data are dynamically aligned as well, and systems can be integrated. Finally, when assumptions and constraints regarding the data and their use are captured as well, the data are harmonized, allowing a unified view.

Using the responding articles of Hoffmann [12] and Page, Briggs, and Tufarolo [13], the LCIM was improved into its current form, which was successfully applied in various application domains, which are not limited to M&S applications, as reported in [14]. The main improvement was to adapt the names of the layers of interoperation to the terms known from the computer linguistic spectrum regarding the increasing level of understanding based on the information exchanged [12]. In addition, Page, Briggs, and Tufarolo proposed to clearly distinguish between the three governing concepts of interoperation:

- Integratability contends with the physical/technical realms of connections between systems, which include hardware and firmware, protocols, networks, etc.
- Interoperability contends with the software and implementation details of interoperations; this includes exchange of data elements via interfaces, the use of middleware, mapping to common information exchange models, etc.
- Composability contends with the alignment of issues on the modeling level. The underlying models are purposeful abstractions of reality used for the conceptualization being implemented by the resulting systems.

In summary, successful interoperation of solutions requires integratability of infrastructures, interoperability of systems, and composability of models. Successful standards for interoperable solutions must address all three categories.

The Levels of Conceptual Interoperability Model

The current version of the LCIM was first published in [15]. In this and the following papers, the LCIM exposes six layers of interoperation, namely:

- The technical layer deals with infrastructure and network challenges, enabling systems to exchange carriers of information. This is the domain of integratability.
- The syntactic layer deals with challenges to interpret and structure the information to form symbols within protocols. This layer belongs to the domain of interoperability.
- The semantic layer provides a common understanding of the information exchange. On this level, the pieces of information that can be composed to objects, messages, and other higher structures are identified. It represents the aligned static data.
- The pragmatic layer recognizes the patterns in which data are organized for the information exchange, which are in particular the inputs and outputs of procedures and methods to be called. This is the context in which data are exchanged as applicable information. These groups are often referred to as (business) objects. It represents the aligned dynamic data.
- The *dynamic layer* recognizes various system states, including the possibility for agile and adaptive systems. The same business object exchanged with different systems can trigger very different state changes. It is also possible that the same information sent to the same system at different times can trigger different responses.
- Finally, assumptions, constraints, and simplifications need to be captured. This happens in the *conceptual layer*. This layer represents the harmonized data.

The following figure shows the LCIM in connection with the three interoperation categories as defined in [13]. The figure adds an additional basis level 0 in which no interoperation takes place and where no interoperability has been established.

The LCIM is unique regarding the dynamic and conceptual level. The viewpoint of the LCIM is to distinguish clearly between the three interoperation categories – integratability, interoperability, composability – and their related concepts within infrastructures, simulations, and models of the systems or services.

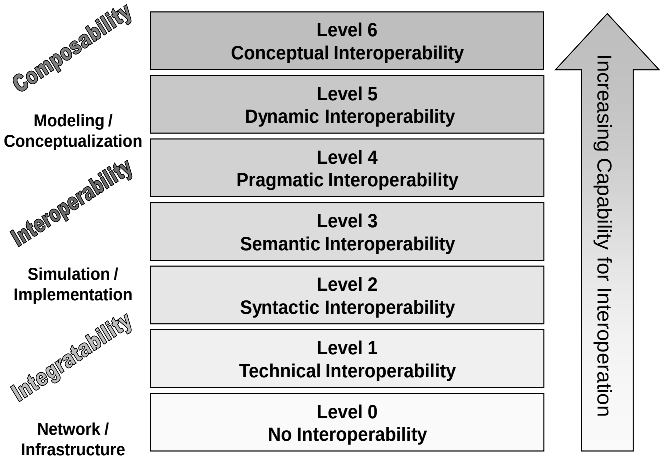

Composability
Modeling / Conceptualization
Interoperability
Simulation / Implementation
Integratability
Network / Infrastructure
Level 6
Conceptual Interoperability
Level 5
Dynamic Interoperability
Level 4
Pragmatic Interoperability
Level 3
Semantic Interoperability
Level 2
Syntactic Interoperability
Level 1
Technical Interoperability
Level 0
No Interoperability
Increasing Capability for Interoperation

Figure 12-1: Levels of Conceptual Interoperability Model

Alternative Layered Views

Although the LCIM has been successfully applied in various domains [14], alternative layered models exist that are of interest and at a similar maturity level. Of particular interest is the following model that finds application in the net-centric environment.

Zeigler, Kim, and Praehofer [15] propose the following architecture for M&S that also comprises six layers. They define these layers as follows:

- The *Network Layer* contains the infrastructure including computer and network.
- The Execution Layer comprises the software used to implement the simulation. This includes protocols, databases, etc.
- The Modeling Layer captures the formalism for the model behavior.
- The Design and Search Layer supports the design of systems based on architectural constraints, comparable to the ideas captured in [6] and mentioned earlier in this chapter.
- The *Decision Layer* applies the capability to search, select, and execute large model sets in support of what-if analyses.
- The *Collaboration Layer* allows experts – or intelligent agents in support of experts – to introduce viewpoints and individual perspectives to achieve the overall goal.

The LCIM maps well to the network, execution, and modeling layer that deal with infrastructure, simulation, and model. The upper three layers are meta-layers that capture the intended and current use of the model, including architectural constraints, which are not dealt with by the LCIM. Using this architecture for M&S, Zeigler and Hammonds define syntax, semantics, and pragmatics as linguistic levels in a slight different way. They define in [16]:

- *Syntax focuses on structure and adherence to the rules that govern that structure, such as XML*
- *Semantics consists of low-level and high-level parts. Low-level semantics focus on definition of attributes and terms, high-level semantics on the combined meaning of multiple terms.*
- *Pragmatics deals with the use of data in relation to its structure and the context of the application (why is the system applied).*

These definitions are different from the similar terms introduced in the LCIM. In particular the pragmatics as defined by Zeigler and Hammonds represent the context of the application; in the LCIM, pragmatics is the context of data exchange within the application.

Zeigler and Hammonds associate these linguistic levels with the architecture for M&S in [16] as shown in the following figure. The difference in the definition of pragmatics becomes obvious, as the intended use capture in the linguistic definition of the term is mapped to the meta-layers of the architecture for M&S.

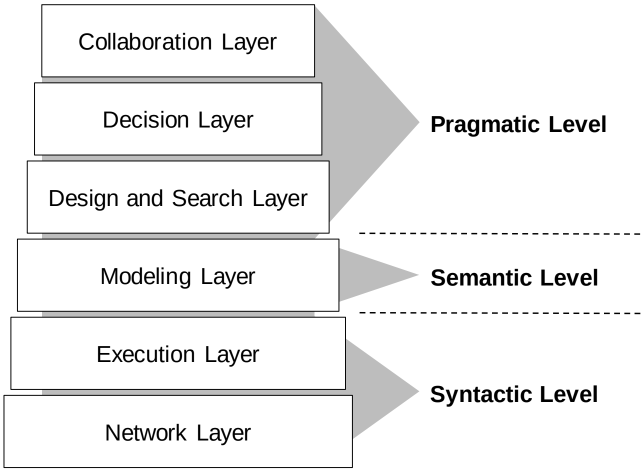


Figure 12-2: Association between Architecture for M&S and Linguistic Levels

Both viewpoints are valid and offer a different perspective of the challenges of interoperability and composability. While the LCIM is unique in defining the dynamic and the agility of systems in the dynamic layer as well as the assumption, constraints, and simplifications in the conceptual layer, the approach of Zeigler and Hammond introduces the intended and current use in for of linguistic pragmatics as an additional challenge.

…

Summarizing Observations on Standardization Efforts and Alternatives

DIS and HLA are well established IEEE standards that have been successfully applied in world wide distributed simulation experimentation. BOM is a SISO standard that introduces the idea of conceptual models of representing entities, events, and patterns of interplay in support of reusability and composability.

XML, RDF/RDFS, and OWL/OWLS provide means that can be applied in support of interoperability and composability of M&S applications. In particular when M&S applications need to interoperate with operational systems that do not follow M&S interoperability standards, this knowledge becomes important.

There are other possibilities that may be considered for special application domains, among these the *Test and Training Enabling Architecture* (TENA) [25]. TENA is not a standard as those described here but an integrated approach to develop distributed simulation for military testing and training, including the integration of live system on test ranges. TENA supports the integration of HLA and DIS based systems, but is neither HLA nor DIS based. Following the TENA philosophy, interoperability requires a common architecture, which is TENA, an ability to meaningfully communicate, which requires a common language provided by the TENA Object Model and a common communication mechanism, which is the TENA Middleware and Logical Range Data Archive. In addition, a common context in form of a common understanding of the environment, a common understanding of time as provided by TENA Middleware, and a common technical process as provided by TENA processes is needed. The specialization of test and training in the military domain is the strength as well as the weakness of TENA, as test and training is well supported, but the transition to other domains requires significant changes to the object model, the data archives, and even the middleware.

Another branch not evaluated in this chapter but worth to be considered is the use of the *Discrete Event System Specification* (DEVS) as documented in [15]. DEVS is a formalism rooted in systems theory. Using this formalism consistently improves reuse and composability, but goes beyond the scope of this chapter.

…

Summary

This chapter introduced the student and scholar to the concepts needed to understand the challenges of integratability, interoperability, and composability. It motivated why it is necessary to distinguish between interoperability of simulation systems focusing on aspects of their implementation and composability of simulation models focusing on aspects of their conceptualization. The LCIM was introduced to systematically evaluate in prescriptive and descriptive applications the various layers of interoperation. The LCIM also identifies artifacts needed to annotate systems and service allowing identifying applicable potential solutions, selecting the best candidates, composing the selected candidates to provide the solution, and orchestrating their execution.

Following these theoretic concepts, current interoperability standardization efforts were introduced: IEEE 1278 DIS [17], IEEE 1516 HLA [18], and SISO-STD-003-2006 BOM [19]. In addition to these M&S specific standards, web-based standards were described as well: XML, RDF/RDFS, and OWL/OWL-S. When applying the LCIM descriptively the elusiveness of the conceptual level becomes apparent.

Current research evaluates engineering methods in support of enabling interoperation in complex systems: data, process, and constraint engineering. This is ongoing research that contributes to the next generation of interoperability standards.

Although this chapter focuses on the technical challenges of interoperability and composability, technical solutions do only support interoperability as a point-solution in time. To ensure interoperability and composability over the life time of a system or a federation, management processes are needed that are integrated into project management as well as into strategic project management [31]. What management processes are needed and which artifacts need to be produced to support them are topics of ongoing research.

References

[1] Institute of Electrical and Electronics Engineers. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York, NY; 1990.

[2] Software Engineering Institute. SEI Open Systems Glossary. Carnegie Mellon, Pittsburgh, PE. Available at http://www.sei.cmu.edu/opensystems/glossary.html. Accessed 2009 May 15.

[3] U.S. Department of Defense (DoD) Directive 5000.01. The Defense Acquisition System. Certified as current as of November 20, 2007 (former DoDD 5000.1, October 23, 2004).

[4] Petty MD, Weisel EW. A Composability Lexicon. In Proceedings of the Spring Simulation Interoperability Workshop, March 30 - April 4, 2003, Orlando, FL; pp. 181-187.

[5] Harkrider SM, Lunceford WH. Modeling and Simulation Composability. In Proceedings of the Interservice/Industry Training, Simulation and Education Conference, 29 November - 2 December 1999, Orlando FL.

[6] Pratt DR, Ragusa LC, von der Lippe S. Composability as an Architecture Driver. In Proceedings of the Interservice/Industry Training, Simulation and Education Conference, 29 November - 2 December 1999, Orlando FL.

[7] Kasputis S, Ng HC. Composable simulations. In Proceedings of the Winter Simulation Conference, December 10-13, 2000, Orlando FL; pp. 1577-1584.

[8] International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 10731:1994. Information technology – Open Systems Interconnection – Basic Reference Model – Conventions for the definition of OSI services. ISO Press, 1994.

[9] Dahmann JS. High Level Architecture Interoperability Challenges. Presentation at the NATO Modeling & Simulation Conference, 25-29 October 1999, Norfolk VA. NATO RTA Publications.

Extracted from Tolk A: “Interoperability and Composability,” in Sokolowski JA, Banks CM: *Modeling and Simulation Fundamentals: Theoretical Underpinnings and Practical Domain, pp.403-433*
© John Wiley 2010

[10] Petty MD. Interoperability and Composability. *Modeling & Simulation Curriculum* of Old Dominion University, Old Dominion University, Norfolk, VA; 2002.

[11] Tolk A, Muguira JA. The Levels of Conceptual Interoperability Model (LCIM). In *Proceedings of the Simulation Interoperability Workshop*, 14-19 September 2003, Orlando, FL.

[12] Hofmann M. Challenges of Model Interoperation in Military Simulations. *SIMULATION* 2004;80:659-667.

[13] Page EH, Briggs R, Tufarolo JA. Toward a Family of Maturity Models for the Simulation Interconnection Problem. In Proceedings of the Simulation Interoperability Workshop, 18-23 April 2004, Arlington, VA.

[14] Tolk A, Turnitsa CD, Diallo SY. Implied Ontological Representation within the Levels of Conceptual Interoperability Model. *International Journal of Intelligent Decision Technologies 2008;*2(1): 3-19.

[15] Zeigler BP, Kim TG, Praehofer H. *Theory of Modeling and Simulation, 2nd ed.* New York: Academic Press, 2000.

[16] Zeigler BP, Hammonds PE. *Model and Simulation-based Data Engineering*. Elsevier Science & Technology Books, New York: Academic Press, 2007.

[17] Institute of Electrical and Electronics Engineers. *IEEE 1278 Standard for Distributed Interactive Simulation*, IEEE publication, Washington, DC.

[18] Institute of Electrical and Electronics Engineers. *IEEE 1516 Standard for Modeling and Simulation High Level Architecture*, IEEE publication, Washington, DC.

[19] Simulation Interoperability Standards Organizations. *SISO-STD-003-2006 Base Object Model (BOM) Template Specification;SISO-STD-003.1-2006 Guide for BOM Use and Implementation.* Available at http://www.sisostds.org. Accessed 2009 May 15.

…

[25] Noseworthy JR. The Test and Training Enabling Architecture (TENA) Supporting the Decentralized Development of Distributed Applications and LVC Simulations. In *Proceedings of the 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications*, 27 – 29 October 2008, Vancouver, Canada; pp. 259-268, 2008.

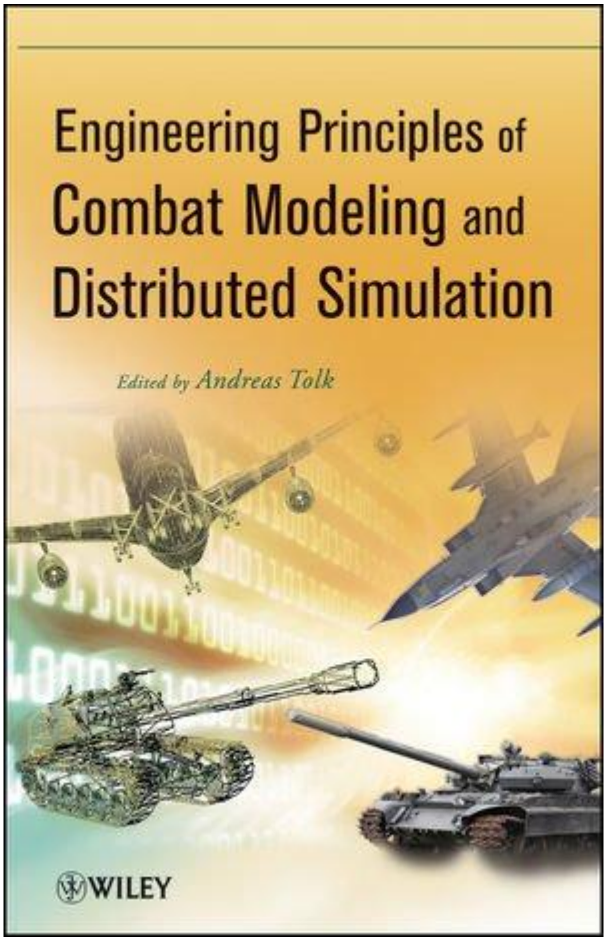
…

[31] Tolk A, Landaeta RE, Kewley RH, and Litwin TT. Utilizing Strategic Project Management Processes and the NATO Code of Best Practice to Improve Management of Experimentation Events. In *Proceedings of the International Command and Control Research and Technology Symposium*, 15–17 June 2009, Washington, DC, USA.

Chapter 12

Standards for Distributed Simulation

Andreas Tolk


Engineering Principles of Combat Modeling and Distributed Simulation
Edited by Andreas Tolk
WILEY

Engineering Principles of Combat Modeling and Distributed Simulation
Andreas Tolk
ISBN: 978-0-470-87429-5

Hardcover
936 pages
March 2012

Extracted from Tolk A: "Standards for Distributed Simulation," in Tolk A: *Engineering Principles of Combat Modeling and Distributed Simulation, pp.209-242, © John Wiley 2012*

The following pages are extractions from the chapter 12 on "Standards for Distributed Simulation" by Andreas Tolk, prepared for the John Wiley book *"Engineering Principles of Combat Modeling and Distributed Simulation,"* edited by Andreas Tolk.

What are Standards for Distributed Simulation?

As described in the last chapter on the challenges of distributed simulation, the tasks to be conducted by the simulation engineering are all driven by the problem of the customer. Every standard that helps

- to capture the objectives of an exercise or another simulation task,
- to derive a conceptual model that can serve as a blue print to guide the simulation engineer through further decisions,
- to identify potential simulation solutions based on the available documentation,
- to select the best simulation solutions to implement a specific solution for the problem of the customer,
- to compose the solutions into a new system – or a federation – which includes the identification of multiresolution and time challenges,
- to integrate networks and infrastructures (including using proxies, brokers and protocol solutions),
- to make the simulation systems interoperable (including using gateways) and identify or develop an information exchange model,
- to ensure that the models are composable,
- to ensure that data needed for initialization are available or can be obtained,
- and all other tasks and subtasks described so far in this book …

… every standard is relevant for distributed simulation. As such, all standards that support distributed systems and computer engineering are relevant. Network standards and standards supporting the Internet are potential candidates.

To give an example, the *Extensible Modeling and Simulation Framework* (XMSF) initiative kicked off by George Mason University in Fairfax, VA, Naval Postgraduate School in Monterey, CA, Old Dominion University in Norfolk, VA, and the Science Applications International Corporation (SAIC) in San Diego, CA, looked at applying World Wide Web standards in support of Internet based distributed simulation of the future (Brutzman et al., 2002; Blais et al., 2005). A recently conducted peer study showed standards of the semantic web can support distributed simulation better and are expected to be applied more (Strassburger et al., 2008), and recent research shows that this indeed may be the case.

In this chapter, the focus will be on two topics, namely (1) standards officially recognized as modeling and simulation standards and (2) standards that are repeatedly successfully applied in support of distributed simulation in numerous modeling and simulation conferences. Both choices can neither be complete nor exclusive, but they are a compilation that should provide a good starting point for simulation engineers looking for support of their tasks.

Modeling and Simulation Standards

The viewpoint taken in this book to identify modeling and simulation standards is driven by purely practical views: to start looking at modeling and simulation standards, the standards supported by the Simulation Interoperability Standards Organization (SISO) build the initial group.

The reason is simple. SISO states in its vision that it will serve the global community of modeling and simulation professionals, providing an open forum for the collegial exchange of ideas, the examination and advancement of M&S-related technologies and practices, and the development of standards and other products that enable greater M&S capability, interoperability, credibility, reuse, and cost-effectiveness (SISO, 2010). Furthermore, the Institute of Electrical and Electronics Engineers (IEEE) Computer Society Standards Activities Board voted in November 2003 to unanimously grant the SISO Standards Activities Committee (SAC) status as a recognized IEEE Sponsor Committee. The SISO SAC Chair serves as SISO's primary contact for all IEEE Standards activities. In addition, SISO maintains a Liaison Member relationship with Sub-Committee 24 (SC 24) of the Joint Technical Committee 1 (JTC 1) of the

International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC). Starting with the standards recognized by SISO as modeling and simulation standards is therefore easily justifiable.

On their website, SISO enumerates IEEE standards, SISO standards, and ISO standards supporting modeling and simulation, which will be covered at least in form of an overview here as well. As of June 2011, the following standards were enumerated. As SISO meets at least two times per year in the USA and one time in Europe, this is a list that is constantly updated. The reader is therefore encouraged to check for updates regularly to ensure that support from the latest developments.

- IEEE Standards
    - High Level Architecture (will be dealt in more detail in this chapter as well as in Chapter 19)
        - IEEE Standard 1516 - Framework and Rules
        - IEEE Standard 1516.1 - Federate Interface Specification
        - IEEE Standard 1516.2 - Object Model Template (OMT) Specification
        - IEEE Standard 1516.3 - Federation Development and Execution Process (FEDEP) Recommended Practice
        - IEEE Standard 1516.4 - Recommended Practice for Verification, Validation, and Accreditation of a Federation—An Overlay to the High Level Architecture Federation Development and Execution Process
    - Distributed Interactive Simulation (will be dealt in more detail in this chapter)
        - IEEE 1278.1 - IEEE Standard for Distributed Interactive Simulation - Application Protocols
        - IEEE 1278.1A - IEEE Standard for Distributed Interactive Simulation - Supplement to Application Protocols - Enumeration and Bit-encoded Values
        - IEEE 1278.2 - IEEE Standard for Distributed Interactive Simulation - Communication Services and Profiles
        - IEEE 1278.3 - IEEE Standard for Distributed Interactive Simulation Exercise Management & Feedback (EMF) - Recommended Practice
        - IEEE 1278.4 - IEEE Standard for Distributed Interactive Simulation - Verification Validation & Accreditation
- ISO/IEC Standards
    - SEDRIS (see Chapter 6)
        - ISO/IEC 18023-1, SEDRIS -- Part 1: Functional specification
        - ISO/IEC 18023-2, SEDRIS -- Part 2: Abstract transmittal format
        - ISO/IEC 18023-3, SEDRIS -- Part 3: Transmittal format binary encoding
        - ISO/IEC 18024-4, SEDRIS language bindings -- Part 4: C
        - ISO/IEC 18025, Environmental Data Coding Specification (EDCS)
        - ISO/IEC 18041-4, EDCS language bindings -- Part 4: C
        - ISO/IEC 18026, Spatial Reference Model (SRM)
        - ISO/IEC 18042-4, SRM language bindings -- Part 4: C
- SISO Standards
    - Approved Standards
        - SISO-STD-001-1999: Guidance, Rationale, & Interoperability Modalities for the RPR FOM (GRIM 1.0) (will be dealt in more detail in this chapter)
        - SISO-STD-001.1-1999: Real-time Platform Reference Federation Object Model (RPR FOM 1.0**)** (will be dealt in more detail in this chapter)
        - SISO-STD-002-2006: Standard for: Link16 Simulations (see also Chapter 23)
        - SISO-STD-003-2006: Base Object Model (BOM) Template Specification (see also Chapter 19)
        - SISO-STD-003.1-2006: Guide for BOM Use and Implementation (see also Chapter 19)
        - SISO-STD-004-2004: Dynamic Link Compatible HLA API Standard for the HLA Interface Specification Version 1.3
        - SISO-STD-004.1-2004: Dynamic Link Compatible HLA API Standard for the HLA Interface Specification (IEEE 1516.1 Version).

        - SISO-STD-005-200X: Link 11 A/B (see also chapter 23)
        - SISO-STD-006-2010: Commercial Off-the-Shelf (COTS) Simulation Package Interoperability (CSPI) Reference Models
        - SISO-STD-007-2008: Military Scenario Definition Language (MSDL) (see also Chapter 24)
        - SISO-STD-008-2010: Core Manufacturing Simulation Data (CMSD)
    - Product Development Groups (PDG), Product Support Groups (PSG), and Standing Support Groups (SSG)
      PDGs are developing a new standard in a community effort. Based on a product nomination, the group reaches a consensus on the standard and recommends the solution to the SAC. In order to work on regular updates once the standard is accepted, a PSG or SSG may be formed. PSG are meeting regularly, SSG only in case of need.
        - C-BML - Coalition - Battle Management Language
        - CMSD - Core Manufacturing Simulation Data
        - CSPI - Commercial Off-the-Shelf Simulation Package Interoperability
        - DDCA - Distributed Debrief Control Architecture
        - DIS - Distributed Interactive Simulation Extension
        - DSEEP - Distributed Simulation Engineering and Execution Process
        - DMAO - DSEEP Multi-Architecture Overlay
        - EPLRS/SADL - Enhanced Position Location Reporting System including Situational Awareness Data Link Simulation Standard
        - FEAT - Federation Engineering Agreements Template
        - GM-VV - Generic Methodology for VV&A in the M&S Domain
        - Link 11 A/B - Link 11 A/B Network Simulation Standard
        - MSDL - Military Scenario Definition Language
        - RPR FOM - Real-Time Platform Reference Federation Object Model
        - SCM - Simulation Conceptual Modeling
        - SRML - Simulation Reference Markup Language

In addition, SISO also sponsors Study Groups (SG) that are made up of experts in the field from academia, industry, and government to evaluate the need and applicability for standardized modeling and simulation solutions for focused topics. These study groups create a final report that often results in the development of a Product Nomination (PN) for a PDG. The standardization process including definitions of roles and responsibilities of the PDG and the SAC and all officers is documented in the Balloted Products Development and Support Process (SISO, 2008).

Other Standards in Support of Distributed Simulation

It is much harder to define other standards successfully applied in support of distributed simulation, as there is no organization that tracks such developments. Some professional organizations of interest to the simulation engineers are captured in an appendix to this book, but there is not a central website or point of contact that can be visited for information. The approach taken here was to evaluate the last five years of modeling and simulation workshops and identify other standards that were documented as successful in these proceedings repeatedly by more than one group. The evaluated workshops are

- Winter Simulation Conference (WSC) of the American Statistical Association (ASA), Association for Computing Machinery (ACM), Institute of Electrical and Electronics Engineers (IEEE), Institute for Operations Research and the Management Sciences: Simulation Society (INFORMS), Institute of Industrial Engineers (IIE), National Institute of Standards and Technology (NIST), and SCS
- Principles of Advanced Distributed Simulation (PADS) of ACM, IEEE, and SCS,
- Spring and Summer Simulation Multi-conferences of the Society for Modeling and Simulation International (SCS), and
- Spring, Euro, and Fall Simulation Interoperability Workshops (SIW) of SISO

Even with this approach it is likely that this summary only captures a small fraction of standards that can support distributed simulation. As stated before, every standard that supports distributed systems supports by definition distributed simulation as well. New programming languages and maybe even simulation languages are developed, web-enabled development tools become more and more important, and so on. Taking these constraints into account, the application of the following three standard categories was documented repeatedly in the domain of modeling and simulation related research events:

(1) Using the discrete event system specification (DEVS) and its variants,

(2) Using modeling languages and architecture languages of the application domain, and

(3) Applying web standards, in particular semantic web standards, in support of distributed simulation.

Many more topics could have made the list, such as data and process modeling issues, but that would have blown this chapter out of proportion. As it should be self-evident that a simulation engineer has a solid programming background and knows “the usual” programming language well enough that algorithms can be read and understood, it is also a natural necessity to continuously observe the domain of distributed systems and evaluate new ideas regarding their applicability to solve challenges in the domain of distributed simulation.

Discrete Event System Simulation Formalism

We understand formalism in this book as a description of something in formal mathematical logical terms so that a machine can read and potentially understand it. Although simulation is not limited to discrete event simulation – we introduced several alternatives in Chapter 4 – this modeling paradigm is predominant in current combat models. A formal representation is therefore helpful.

The Discrete Event System Simulation (DEVS) formalism was developed by the research group around Bernard Zeigler in support of establishing a theory of modeling and simulation. The latest edition has been published by Zeigler et al. (2000). The formalism builds models from the bottom up.

…

The formalism can be used for very practical applications, as shown in many examples by Wainer (2009). The application to combat modeling task is shown in Chapter 21. Atomic and coupled DEVS together build the classic sequential DEVS approach. With the advancements in computer technology, there have also been numerous extensions, also explained with examples in Wainer (2009), such as parallel DEVS, dynamic DEVS, cellular DEVS, and more.

It is important to distinguish between the DEVS formalism and *DEVS implementations* based on this formalism. Several simulation development frameworks have been proposed based on the DEVS formalism that are united by the common roots, but that does not mean that they are interoperable. Recently, a DEVS standardization effort was launched that tries to establish a community of DEVS practice that works on common standards to support interoperable implementations as well. Examples for several DEVS implementations as well as resulting interoperability challenges are given by Wainer et al. (2010). Most implementations use Java, C++, or C# as programming languages. Examples for DEVS implementations and platforms are

- adevs: A C++ library for constructing discrete event simulations based on the parallel DEVS and dynamic DEVS
- CD++: An environment for developments based of DEVS and cellular DEVS
- CoSMo-Sim: An integrated developer environment for DEVS and parallel DEVS, also supports cellular automata and XML based models
- DEVS++: Open Source Library for C++
- DEVS#: Open Source Library for C#
- DEVSJAVA and DEVS Suite: Development environment for JAVA
- DEVSSOA: Service oriented Architectures based on DEVSML (DEVS Modeling Language), and
- DEVSim++: Extension on the basic simulator for DEVS, used in Chapter 21

This list is just a small subset. Many of these environments are open source, in particular those developed by the academic community. An addition, Mittal (2010) developed the DEVS Unified Process which connects DEVS to system architecture frameworks.

The DEVS formalism and DEVS implementations have been extensively dealt with in proceedings by academicians and practitioners in many simulation domains. Many implementations and development environments are open source and therefore often used. The simulation engineers for combat engineering should at least know of these developments. In particular when it comes to the need to federate current combat simulation solutions with new simulation domains needed for new military tasks – such as human, social, cultural, and behavior models as discussed later in this book – it is more than likely that some of them will follow the DEVS formalism and make use of its development environments and implementation. A small group is already looking into DEVS/HLA applications and interoperability challenges, but this research is still in its beginning.

Modeling and Architecture Languages

The more general block of modeling and architecture languages is in particular needed to communicate models and implementations among different stake holders, such as communications between customers, potential users of the product, and engineers, but also between simulation engineers active in different phases of the life cycle of a simulation. We will look at different phases in a following chapter of this book in more detail. Modeling and architecture languages are used to specify descriptions and documentations of systems and architectures based on a set of agreed and aligned artifacts, such as tables, diagrams, or figures. It is highly desirable that these artifacts are based on a common repository containing all described elements only once to ensure consistency in descriptions.

Examples we want to have a look at in this section of the chapter are the Unified Modeling Language (UML), the System Modeling Language (SysML), and the U.S. Department of Defense Architecture Framework (DoDAF). Again, this is only a very limited subset, but it will be used to address the main ideas the simulation engineer has to known in order to apply comparable solutions to facilitate work. In addition, such modeling languages have become often the *lingua franca* between engineers and simply have to belong to the tool set of a simulation engineer supporting combat modeling.

UML and SysML are closely related. They are both governed by the Object Management Group (OMG) and build the basis for many community standards. They even overlap significantly, as SysML was defined as a subset of UML that was extended to better support system modeling while UML was defined mainly in support of software engineering (although UML was applied for many other approaches as well, such as business modeling). UML artifacts are also used to describe several DoDAF artifacts. The descriptions here cannot replace a more detailed introduction or a tutorial, but it should be sufficient to motivate to more deeply deal with these topics.

UML uses different artifacts and diagrams that can be used to provide different points of views on one system. The latest official version of UML is 2.3, but in March 2011 the beta version of 2.4 was released. UML is continuously improved and enriched by a huge variety of users and developers.

The traditional view of systems modeling distinguishes between static views describing the structures and functional views describing the behavior exposed. These views describe what entities are modeled (as classes describing the types and objects describing the instances of things), how entities react when they receive input (as the state changes, which can be broken down in case the entity is a composite made up of several smaller entities), and how entities interact with each other (in form of activities in which more than one entity work together and exchange messages with each other).

…

UML has been applied in a variety of communities and has become something like the lingua franca between modelers that specify software intensive systems. As such, the simulation engineer needs to be at least able to read and interpret the artifacts. The main disadvantage of UML, however, is the focus on object-oriented software systems. While UML easily can be interpreted to support other systems as well, the software engineering roots often shine through.

These led to the development of SysML, also under the umbrella of the Object Management Group. SysML is a general modeling language for systems and gets more and more support from systems engineers. There is a huge overlap between UML and SysML, but the focus is on the system, not on the software. As such, SysML started by stripping all software specific diagrams from UML, modify the other in case of need, and adding system specific new diagrams where needed.

…

The simulation engineer can use SysML specifications not only for the development of federations, but many weapon systems and combat support systems are documented in SysML. A feature of particular interest is the traceability of system component functionality to requirements, which can be a significant help when simulation developers have to decide which details should be included into a model. If, e.g., a component of a system hosts many of the functionalities driven by high priority requirements it may be wise to represent this component explicitly to allow for better damage evaluations in the light of original requirements.

Extracted from Tolk A: “Standards for Distributed Simulation,” in Tolk A: *Engineering Principles of Combat Modeling and Distributed Simulation, pp.209-242, © John Wiley 2012*

The last standard a simulation engineer supporting combat modeling to be explicitly mentioned in this section is DoDAF. A later chapter will deal with this topic in some more detail, but under a slightly different viewpoint.

DoDAF is currently published and applied in different versions. The most current version is DoDAF 2.0, but many organizations are still using DoDAF 1.5. We will have a look at both versions to show the main difference. As all systems used within the US Department of Defense principally should be documented using the artifacts defined by DoDAF, it is very helpful for the simulation engineer to know the framework. Additional arguments for the use of DoDAF in the context of Combat Modeling and Distributed Simulation are compiled by Atkinson (2004).

DoDAF is rooted in earlier systems engineering approaches. The C4ISR Architecture Framework was created in response to the passage of the Clinger-Cohen Act and addressed in the 1995 Deputy Secretary of Defense directive that a DoD wide effort be undertaken to define and develop a better means and process for ensuring that C4ISR capabilities were interoperable and met the needs of the warfighter. The first version was published in June 1996, rapidly followed by version 2.0 in December 1997. The second version was the result of the continued development effort by the C4ISR Architecture Working Group and was mandated for all C4ISR architecture descriptions in a February 1998 memorandum by the Architecture Coordination Council, co-chaired by the Under Secretary of Defense for Acquisition and Technology (USD A&T), the Assistant Secretary of Defense for Command, Control, Communications, and Intelligence (ASD C3I), and the Command, Control, Communications, and Computer Systems Directorate, Joint Staff (J6).

In August 2003, the DoD Architecture Framework Version 1.0 was released. It restructured the C4ISR Framework v2.0 to offer guidance, product descriptions, and supplementary information in two volumes and a desk book. The objective was to broaden the applicability of architecture tenets and practices to all mission areas rather than just the C4ISR community. Therefore, the first version explicitly addressed usage, integrated architectures, DoD and federal policies, value of architectures, architecture measures, DoD decision support processes, development techniques, and analytical techniques.

In April 2007, the DoD Architecture Framework, Version1.5 was released as an evolution of the first version. It reflects and leverages the experience that the DoD components have gained in developing and using architecture descriptions. However, it was designed as a transitional version providing additional guidance on how to reflect net-centric concepts within architecture descriptions, and including information on architecture data management and federating architectures through the DoD. The second version also incorporates the Core Architecture Data Model (CADM), a data model that consistently described all modeled aspects as reflected in various artifacts in support of providing a repository. It also emphasized the use of UML based artifacts to express the different views needed to describe the system.

Since May 2009, DoDAF Version 2.0 has been available. This version is defined as the overarching, comprehensive framework and conceptual model enabling the development of architectures to facilitate DoD managers at all levels to make key decisions more effectively through organized information sharing across DoD, Joint Capability Areas, Component, and Program boundaries. This new version provides an overarching set of architecture concepts, guidance, best practices, and methods to enable and facilitate architecture development. The focus is the fit-for-purpose concept which allows defining individual view points that support special needs. To this end, the artifact focus has to shift from the different diagrams used in various views and that are harmonized based on a common repository to a common data and metadata model that drives different view points, including user defined ones.

…

DoDAF 2.0 extends the view categories significantly. This version also uses the term view point to emphasize that not the view itself should be the product, but the data that are represented via each view point. The CADM of earlier version has been replaced by the DoDAF Meta-model (DM2) that supports the architecture content of the DoD core processes by design. While CADM was derived from the various views that needed to be aligned to avoid inconsistencies, DM2 was designed to capture the relationships needed to support qualitative and quantitative analysis. The resulting ability to relate architectural descriptions, based on a common architecture vocabulary as used by decision makers and war fighter as envisioned in the NATO COBP, allows better analysis across architectural description. The price is a more complex data model underlying the view points. As all core processes are in the focus of this version, the scope had to be moved from a single group of systems towards DoD as an enterprise.

…

For examples the interested reader is referred to the original DoDAF documents that are freely distributed via the Internet. It is worth to point out that NATO has the NATO Architecture Framework (NAF), and the UK has the Ministry of Defense Architecture Framework (MoDAF). In order to support a modeling method in support of these main frameworks, OMG defined the Unified Profile for DoDAF/MODAF (UPDM) mainly using UML products as the common denominator. Other nations support similar efforts, such as the Department of National Defence and Canadian Forces Architecture Framework (DNDAF) of Canada.

The better the simulation engineer knows these modeling and architecture languages, the easier is communication in the international community as well as within the community of engineers with different professions that support an exercise, or maybe in real operations in which the simulation engineer's expertise is required to ensure success.

Semantic Web Standards

Tim Berners-Lee coined the term "Semantic web." The semantic web is understood by the community as a web of data. While the hyperlinked documents of the Internet were merely designed to support humans, the idea of the semantic web is to add machine-readable and machine-interpretable metadata to the websites that allow machines to support finding data, make connections and relations between websites, etc.

Several methods and technologies are needed to make this vision a reality. The needed standards and tools as currently understood to build the semantic web stack are shown in the following figure.

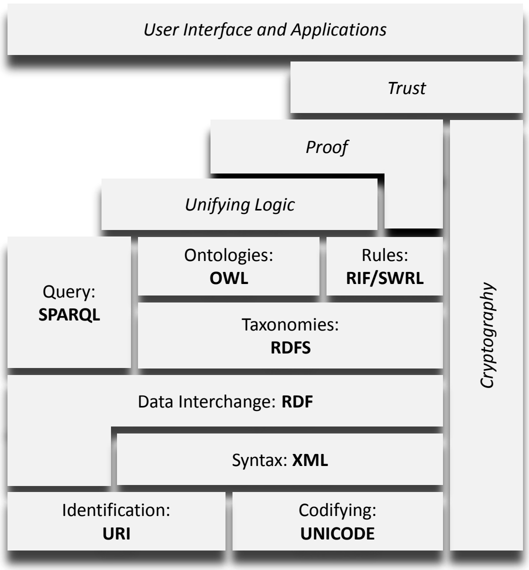


Figure 1: Semantic Web Stack

The set of standards that make up the lower left block of the stack (and that are written in bold font) allowed moving towards a new level of machine support and automatic reasoning in the recent years was never been possible before. In addition, the technologies are continuously further developed and exchanged within the semantic web community to foster agreement and improvement.

- The basis for general communication in the web is the use of a machine and vendor independent codification allowing the interpretation of signals as symbols. This is supported by the use of *UNICODE* as the lowest building block.
- The second foundational leg is the use of *Uniform Resource Identifiers (URI)*. Everything on the web becomes a resource and can be addressed as such in a standardized way.
- The *Extensible Markup Language (XML)* provides the elementary syntax for resources, which means allows communicating structure within a resource that can be search for, extracted, etc. It is

possible to provide and restrict the structure and content of elements through the use of an XML Schema. Another approach that is seemingly gaining more and more support is Turtle, which is *XML* independent, providing an alternative.

- The *Resource Description Framework (RDF)* allows to structure resources and their relations similar to using a data model. Most of these frameworks use XML, but that is not longer necessary, as alternative exist. However, whenever data need to be exchanged in a consistent way, which includes the idea of transactions on one or several resources, support of a common RDF is needed. RDF organizes resources as triples, which provide the structure for higher operations.
- While RDF is more a data model, the use of an *RDF Schema (RDFS)* compares to the definition of a taxonomy of terms describing properties and classes of RDF resources. It allows building hierarchies and complex relationship using well-defining types of associations connecting well-defined structured tagged with common terms. While RDF works with general triples, RDFS standardizes the most important of these triples as recognized by the community.
- The Web Ontology Language (OWL) is an extension of RDFS. It adds more standardized terms describing resources, properties, and relations, such as cardinality, equality, typing, enumerations, and more. OWL allows reasoning over these extensions, if all supporting resources follow the standard accordingly.
- Queries about resources do not require that the resources are documented in RDFS and OWL. The SPARQL Protocol and RDF Query Language (SPARQL) allows searching for triples, conjunctions of triples, disjunctions of triples, and optional patterns. As such, SPARQL works with RDF, RDFS, and OWL.
- The Rule Interchange Format (RIF) is not yet standardized, but only recommended. It allows the communication of constraints within the semantic web stack to ensure consistency between the layers. An alternative for OWL based resources is the Semantic Web Rule Language (SWRL) that utilizes description logic subsets of OWL (the dialect OWL-DL).

The use of OWL should not be decided without proper consideration of the issues of decidability and computational complexity. Not all dialects of OWL are decidable, and even if they are decidable, their application can lead to non-polynomial computing time. Several papers deal with these challenges. The simulation engineers should focus on OWL-DL (which is decidable) and preferably even the profiles defined for OWL 2: OWL 2 EL, OWL 2 QL, and OWL 2 RL. For more information, a good review of available semantic web tools and standards and their computational constraints is given by Hitzler, Krötzsch, and Rudolph (2009).

Building a unifying logic that supports all alternatives that becomes the foundation of solid proofs is an ongoing effort. Together with cryptography (which was originally envisioned to be based on XML as well, but this idea is no longer supported generally) the proof layer becomes the basis for trusted information providers that can support the applications. The importance of these ideas and protocols becomes immediately obvious when we go back to the data discussions in the NATO COBP, the need to initialize simulation from trusted sources, the need for common initialization, and more. If the infrastructure that supports the distributed simulation is based on the semantic web stack, many of the tasks of the simulation engineer are taken care of. Vice versa, the simulation engineer can use personal experiences in distributed simulation to enhance the ideas accordingly. Tolk (2006) contributed to enhance the vision by actively pushing the idea of semantic web based modeling and simulation for the next generation of the web. An interesting perspective on future directions for semantic systems is given by Sowa (2011).

To practically apply these ideas, the simulation engineer has two immediate options: (a) using the concept to better describe modeling and simulation resources using metadata that is stored in registries, and (b) using web services to access modeling and simulation resources that are available. This connects directly back to the four tasks in support of distributed simulation: to identify applicable solutions, to select the best options, to compose the best solution, and to orchestrate the execution.

The community already started to work on identifying metadata that can be used to describe modeling and simulation resources, which includes data sources, simulation systems, and more (US DoD M&S CO, 2008). The approach is summarized by Gustavson et al. (2009). The purpose of this specification is to standardize on the set of metadata used to describe resources in Modeling and Simulation Resource Repository (MSRR) nodes and similar applications, and to ensure that the product metadata templates will align with the DoD Discovery Metadata Specification (DDMS) as part of the Global Information Grid (GIG)/Net-Centric Data Strategy. The idea was generalized in support of a Modeling and Simulation

Information System (MSIS) for the US DoD. Similar approaches are conducted in other countries as well. Several MSRRs are already in use, and there is potential to merge these different approaches based on a common set of metadata. Current efforts, application of the metadata, tools, and the guiding vision were recently presented by Gustavson et al. (2009).

To access and utilize modeling and simulation services, Web services are the method of choice in most cases. In general, Web services are services that can be accessed via the Internet based on the specifications of the service published as well. The uses for military combat modeling and distributed simulation applications were described by Tolk et al. (2006), the general ideas were covered by the already mentioned XMSF projects (Brutzman et al., 2002; Blais et al., 2005).

There are several different categories of Web services. The earlier versions were based on a combination of the using XML to define the data to be exchanged, the Simple Object Access Protocol (SOAP) to access the service, the Web Service Description Language (WSDL) – which is based on XML – to describe the access characteristics of the service, and Universal Description Discovery and Integration (UDDI) –also based on XML – to support publication and discovery. The general idea was that a Web service provider described the service characteristics in WSDL and posted them to the UDDI server. Whoever needed to find such a service downloaded the descriptions from the UDDI and looked for applicable solutions based on the WSDL description. If a service was found and accessed, the searcher used XML to provide the needed data and accessed the service via SOAP. The disadvantage was that this procedure, from the logic very close to a remote procedure call (RPC), required a lot of knowledge regarding the service interface, the information exchange requirements, etc.

To overcome these constraints, Representational State Transfer (REST) methods were applied to define RESTful services. The idea of REST is to generalize the interface as much as possible and instead put the needed information into the data exchanged, not the application programming interface. This allows making component interactions scalable, deploying them independently, reducing latency, and enforcing security. As REST allows building wrappers around existing systems, such encapsulation of legacy systems supports migration towards this new infrastructure. To allow for these approaches, RESTful services expose all services as resources that are addressed via URI. They are connected via uniform connectors or channels that are used to exchange messages that comprise all information in form of metadata and data that the receiving service needs to act on the message. This allows every RESTful service to provide a uniform interface that addresses the same fundamentals. To these fundamentals belong the following:

- Within the messages, resources are identified by URIs and can be manipulated by the receiver;
- Each message includes enough information to describe how to process the message;
- The receiver uses hypermedia to make state transitions; and
- All transactions are handed by the providing server.

RESTful services have been successfully applied in simulation infrastructures. Examples are given, among others, by Al-Zoubi and Wainer (2009).

The simulation engineers should observe the developments in the domain of distributed computing. These tools and standard can support tasks and may even develop to the point that they can replace the specific simulation interoperability standards given in the last section of this chapter. In any case they can support these efforts of distributed simulation significantly.

Of additional and increasing interest is the use of standards supporting the gaming industry. This topic will be dealt with in more detail in the context of Chapter 17 of this book.

Modeling and Simulation Interoperability Standards

This last section gives an overview of the two IEEE Modeling and Simulation Interoperability Standards: the IEEE 1278 Distributed Interactive Simulation (DIS), and IEEE 1516 High Level Architecture (HLA). A more technology-oriented overview is given by Tolk (2010). These standards are dealt with in more detail under different viewpoints also in other chapters of this book, including the history as well as their role for multiresolution challenges.

The detail given in this section cannot replace a comprehensive introduction. Readers interested in a comprehensive guide to DIS are referred to Neyland (1997). For the HLA, Kuhl et al. (2000) is a good place to start, but there are also very good tutorials available during interoperability focused workshops, such as the Simulation Interoperability Workshops of SISO.

Another effort that needs to be mentioned is the *Test and Training Enabling Architecture (TENA)*. Although not being a *de jure* standard, the ideas supported by TENA are pivotal to interoperable simulation systems. TENA is described in detail in Chapter 20 of this book.

IEEE 1278 Distributed Interactive Simulation

In the 1980s, the Defense Advanced Research Project Agency (DARPA) and the US Army initiated the development of a simulation network (SIMNET) that allowed coupling former stand-alone simulators to support better training. In this prototype, SIMNET showed how to combine individual tank simulators of the Combined Arms Tactical Training System (CATT) to enable tank crews to operate side-by-side in a common synthetic battle space. The individual simulators represented weapon systems on this common virtual battlefield that had a well defined set of actions and interactions: tanks could move, observe, shot at each other, exchange radio communication, etc. Individual activities led to status changes that were communicated via status reports. Interactions were communicated via messages.

…

As the set of information exchange specifications could be well defined, this resulted in the idea to standardize these messages, which led to the IEEE1278 Distributed Interactive Simulation (DIS) standard: the Protocol Data Units (PDUs) captured syntactically and semantically all possible actions and interactions based on the idea that individual simulators represent individual weapon platforms. Only later, instead of individual platforms also groups and aggregates (like platoons or companies) were accepted as receivers and producers of such PDUs, but these groups were understood as individual entities in the battle space as well.

Following the principles learned in SIMNET, the DIS community defined and standardized PDUs for all sorts of possible events that could happen during such a military training. Whenever a preconceived event happens – such as one tank firing at another, two system colliding, artillery ammunition being used to shoot into special area, a report being transmitted using radio, a jammer being used to suppress the use of communication or detection devices, and more – the appropriate PDU is selected from the list of available PDUs and used to transmit the standardized information describing this event. Within a PDU, syntax and semantics are merged into the information exchange specification.

…

DIS is still successfully used and supported by a large user community. As mentioned earlier in this chapter there are still standardization efforts going on under the umbrella of SISO. Furthermore, the Realtime-Platform-Reference Federation Object Model (RPR-FOM) activities described in the following section migrated many DIS solutions into the new IEEE 1516 world. Also, DIS-HLA Gateways are often applied to integrate the functionality provided by DIS conform simulation systems into HLA federations.

IEEE 1516 High Level Architecture

Although Chapter 19 will deal with the High Level Architecture in more detail, a short summary will be given to allow for a better presentation of the follow-on ideas in the next section. HLA was developed to unify various distributed simulation approaches within the US DoD and has been adopted by NATO as well. Under the lead of the US Defense Modeling and Simulation Office (DMSO), several prototypes were developed and distributed in several versions. The last version that was submitted to IEEE for standardization was the HLA 1.3 NG. Many US companies are still using this version to this day, as many tools were freely distributed by DMSO. The international standardization group under IEEE improved this version by bringing it up-to-date regarding supported standards and generalizing some of the ideas. For example, the Backus-Naur-nomenclature used in 1.3 NG was replaced by XML. Furthermore, hard-coded enumerations were replaced by reconfigurable solutions and configuration tables. The result was the IEEE 1516 – 2000 HLA, which became the main version implemented in Europe and parts of Asia and Australia. Although most differences between both versions were of editorial nature, gateways are needed to connect federations that are based on different versions.

As every IEEE standard, HLA was reviewed and updated after 10 years. Under the title HLA Evolved, this work was conducted under the lead of SISO. The result was the updated version of the standard: IEEE 1516 – 2010 HLA. The main difference between this new version and the older ones is that the information exchange model became modular, so that part of the information exchange agreement can be changed during runtime. Furthermore, dynamic link capabilities, extended XML support, increased fault tolerance, and web-based standards were integrated into the concepts supported by HLA-based federations. At the point in time that this book was written, only some early adopters used the 1516-2010, but several supporters and organizations already announced the decision to go straight to this new standard version when they update their federations.

The objective of defining the HLA was to define a general purpose architecture for distributed computer simulation systems. It defines a federation made up out of federates, which are the simulation systems, and the connection middleware that allows the information exchange between the simulation systems.

…

RPR-FOM and GRIM

When HLA was introduced to the combat modeling and simulation community, most distributed simulation systems supported DIS in a real-time environment. Many systems had just been modified to support working not only as stand-alone simulators but to be part of a common virtual battle space. In particular the community of virtual training simulators saw no real need to conduct another potentially expensive conversion to support a new simulation interoperability protocol.

In order to facilitate the integration of these simulators without expensive conversions of the systems, two approaches were used: the use of DIS-HLA Gateways as discussed by Steel (2000), and the development of a technical standard in form of the Realtime-Platform-Reference Federated Object Model (RPR-FOM) and the supporting Guidance, Rationale, and Interoperability Modalities for the RPR-FOM (GRIM). We will focus on the second approach in this section.

The RPR-FOM and GRIM was developed under the leadership of SISO with the goal to implement the DIS PDU structures within HLA objects and attributes and interaction and parameters as well as to provide an intelligent translation of the concepts used in DIS to an HLA environment. While the RPR-FOM defines the information exchange means needed by DIS simulation systems, the GRIM documents the guidelines on how to use them most efficiently. As the DIS standards evolves continuously, so does the RPR-FOM and the GRIM as well. As mentioned earlier in this chapter, RPR-FOM and GRIM are SISO standards in version 1.0, but versions 2.0 and 3.0 supporting new developments and agreements in DIS are worked on by standardization groups already.

…

References

Al-Zoubi, Khaldoon, and Wainer, Gabriel (2009). Performing Distributed Simulation with RESTful Web-Services Approach. Proceedings of the Winter Simulation Conference, Austin, TX, pp. 1323-1334

Atkinson, Ken (2004). Modeling and Simulation Foundation for Capabilities Based Planning. *Proceedings Spring Simulation Interoperability Workshop*, IEEE CS Press, Orlando, FL

Blais, Curtis L., Brutzman, Don, Drake, David, Moen, Dennis M., Morse, Katherine L., Pullen, J. Mark, and Tolk, Andreas (2005). Extensible Modeling and Simulation Framework (XMSF) 2004 Project Summary Report. Final Report NPS-MV-05-002, Naval Postgraduate School, Monterey, CA

Brutzman, Don, Zyda, Michael, Pullen, J. Mark, and Morse, Katherine L. (2002). Extensible Modeling and Simulation Framework (XMSF) Challenges for Web-Based Modeling and Simulation. Workshop Report, Naval Postgraduate School, Monterey, CA

…

Gustavson, Paul, Nikolai, Ali, Scrudder, Roy, Blais, Curtis L., and Daehler-Wilking, Richard (2009). Discovery and Reuse of Modeling and Simulation Assets. Modeling and Simulation Information Analysis Center (MSIAC) Journal, Volume 4, Number 2, pp. 11-20

Hitzler, Pascal, Krötzsch, Markus, and Rudolph, Sebastian (2009). Foundations of Semantic Web Technologies. Chapman & Hall/CRC

Institute of Electrical and Electronics Engineers. IEEE 1278 Standard for Distributed Interactive Simulation, IEEE publication, Washington, DC.

Institute of Electrical and Electronics Engineers. IEEE 1516 Standard for Modeling and Simulation High Level Architecture, IEEE publication, Washington, DC.

Kuhl, Frederick, Dahmann, Judith, and Weatherly, Richard. (2000) Creating Computer Simulation Systems: An Introduction to the High Level Architecture. Prentice Hall PTR

Extracted from Tolk A: “Standards for Distributed Simulation,” in Tolk A: *Engineering Principles of Combat Modeling and Distributed Simulation, pp.209-242, © John Wiley 2012*

Mittal, Saurabh (2010). Agile Net-Centric Systems Using DEVS Unified Process. In: Intelligence-based Systems Engineering, edited by Andreas Tolk and Lakhmi Jain, ISRL 10, Springer, Berlin, pp. 159–199

Neyland, David L. (1997). Virtual Combat: A Guide to Distributed Interactive Simulation. Stackpole Books

…

SISO (2008). Balloted Products Development and Support Process. Administrative Document SISO-ADM-003-2008, Orlando, FL

SISO (2010). The SISO Vision. Administrative Document SISO-ADM-004-2010, Orlando, FL

SISO website: http:www.sisostds.org (last visited June 2011)

Sowa, John F. (2011). Future Directions for Semantic Systems. In: Intelligence-based Systems Engineering; edited by Tolk A., and Jain, L.C. Springer-Verlag, ISRL 10, Berlin Heidelberg, Germany

…

Strassburger, Steffen, Schulze, Thomas, and Fujimoto, Richard (2008). Future Trends in Distributed Simulation and Distributed Virtual Environments: Results of a Peer Study. Proceedings of the Winter Simulation Conference, Miami, FL, pp. 777-785

Tolk, Andreas (2006). What comes after the Semantic Web, PADS Implications for the Dynamic Web. Proceedings of the 20th ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation, Singapore, IEEE CS press, pp. 55-62

…

US Department of Defense (2007). DoD Architecture Framework Version 1.5. Volume I: Definitions and Guidelines; Volume II: Product Descriptions; Volume III: Architecture Data Description. US DoD Chief Information Officer, Washington, DC

US Department of Defense (2009). DoD Architecture Framework Version 2.0. Volume I: Introduction, Overview, and Concepts: Manager’s Guide; Volume II: Architectural Data and Models: Architect’s Guide; Volume III: DoDAF Meta-model: Physical Exchange Specification Developer’s Guide. US DoD Chief Information Officer, Washington, DC

US Department of Defense Modeling and Simulation Coordination Office (US DoD M&S CO) (2008). Modeling and Simulation (M&S) Community of Interest (COI) Discovery Metadata Specification (MSC-DMS) Version 1.1. Alexandria, VA

Wainer, Gabriel A. (2009). Discrete-Event Modeling and Simulation: A Practitioner's Approach. CRC Taylor & Francis, Boca Raton, FL

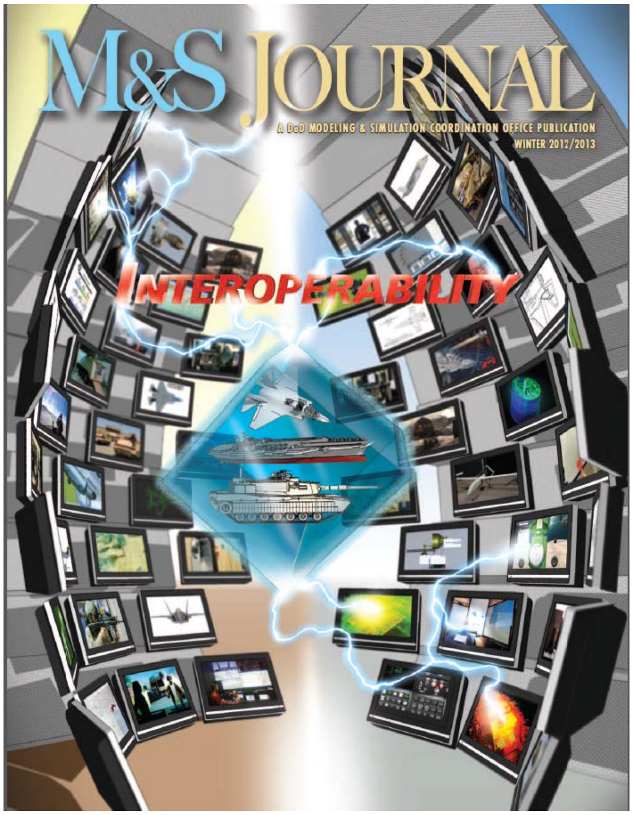
Wainer, Gabriel A., Al-Zoubi, Khaldoon, Mittal, Saurabh, Risco Martín, Jose Luis, Sarjoughian, Hessam, Zeigler, Bernard P. (2010). Standardizing DEVS Simulation Middleware. In: Discrete-Event Modeling and Simulation: Theory and Applications, edited by Wainer, G., and Mosterman, P., CRC Press, Taylor and Francis, pp. 459-493

Zeigler, Bernard P., Praehofer, Herbert, and Kim, Tag Gon (2000). Theory of Modeling and Simulation, Second Edition. Academic Press, San Diego, CA

ARTICLE PUBLISHED IN THE

M&S JOURNAL WINTER 2012

SPECIAL ISSUE ON INTEROPERABILITY



Andreas Tolk, Saikou Diallo, Jose Padilla, and Charles Turnitsa:
"How is M&S Interoperability different from other Interoperability Domains?"
M&S Journal, Winter 2012-2013; pp. 5-14

How is M&S Interoperability different from other Interoperability Domains?

AUTHORS

Dr. Andreas Tolk
Engineering Management
& Systems Engineering
242B Kaufman Hall
Old Dominion University
Norfolk, VA 23529, USA
atolk@odu.edu

Dr. Saikou Y. Diallo, Dr. Jose J. Padilla
Virginia Modeling Analysis
& Simulation Center
Old Dominion University
1030 University Blvd
Suffolk, VA 23435, USA
sdiallo@odu.edu; jpadilla@odu.edu

Dr. Charles D. Turnitsa
TSYS School of Computer Science
Columbus State University
Columbus, GA 31907, USA
cturnitsa@gmail.com

Keywords

M&S interoperability standards

Abstract

During every standard workshop or event, the examples of working interoperability solutions are used to motivate for ‘plug and play’ standards for M&S as well, like standardized batteries for electronics, or the use of XML to exchange data between heterogeneous systems. While these are successful applications of standards, they are off the mark regarding M&S interoperability. The challenge of M&S is that the product that needs to be made interoperable is not the service or the system alone, but the model behind it as well. The paper shows that the alignment of conceptualizations is the real problem that is not yet dealt with in current interoperability standards.

1 Introduction

TO ANSWER THE QUESTIONS OF HOW AND WHY MODELING AND SIMULATION (M&S) INTEROPERABILITY ARE DIFFERENT FROM OTHER INTEROPERABILITY DOMAINS, WE HAVE TO GAIN A BETTER UNDERSTANDING OF WHAT MAKES M&S SPECIAL FIRST. IN OTHER WORDS, WE NEED TO UNDERSTAND THE EPISTEMOLOGY OF M&S AND ANSWER THE QUESTION IF AND HOW IT IS DIFFERENT FROM OTHER RELATED INTEROPERABILITY DOMAINS. TO ANSWER THIS QUESTION, WE FURTHERMORE LIMIT OUR DISCOURSE AND FOCUS ON M&S SUPPORTING COMPUTER SIMULATIONS AND INFORMATION TECHNOLOGY (IT) INTEROPERABILITY DOMAINS.

2 Current Interoperability Standards

One of the most often used examples for solved interoperability challenges are batteries. There is hardly a workshop on interoperability in which it is not used: based on the standard that defines measurements like size, electronic data, voltage, and ampere, the same battery can power a radio, flashlight, night vision goggles, or the proverbial toy bunny. Another example closer to software is the use of the Extensible Markup Language (XML) to exchange data between heterogeneous systems. The XML standard uses basic standard foundations, so that many heterogeneous systems can support them easily (like being fully Unicode compliant), but is extensible to support complex information exchange needs.

The final examples of working interoperability solutions are web services and cloud computing. Although different in their implementation, the underlying conceptual ideas are comparable: a service is well defined by its interface (input and output parameters) and, if necessary, by additional constraints, such as timing, synchronization points, and more. The semantic markup for services OWL-S [1] defines three categories needed to describe services (as shown in figure 1):

- With the *ServiceProfile*, the service presents "what the service does." As specified in OWL-S, [1] this includes the description of what is accomplished by the service, limitations on service applicability and quality of service, and requirements that the service requester must satisfy to use the service successfully.
- Within the *ServiceGrounding* definition, the service supports different ways "how to access it." In this part, communication protocols, message formats, and other service-specific details such as port numbers are specified.
- Finally, a service is described by a *ServiceModel* that defines "how the service works." This description fulfills the tasks of detailing the semantic content of requests, the conditions under which particular outcomes will occur, and, where necessary, the step by step processes leading to those outcomes.

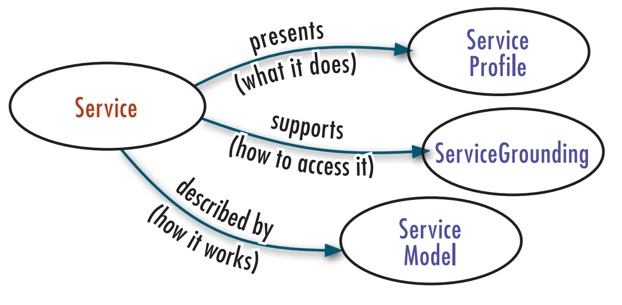


Figure 1: OWL-S

The authors showed in "Ontology Driven Interoperability – M&S Applications," [2] that OWL-S is one of the most advanced available standards supporting interoperability for M&S applications. These findings were based on research conducted in support of the Extensible Modeling and Simulation Framework (XMSF) initiative that evaluated the applicability of web-based standards to drive interoperability for M&S [3, 4]. All these standards are applied successfully, including in the M&S domain.

In addition, we have M&S specific solutions that successfully have been standardized via SISO and IEEE, namely the Distributed Interactive Simulation (DIS) protocol [5], standardized in IEEE1278, and the Modeling and Simulation High Level Architecture (HLA) [6], standardized in IEEE1516. Despite significant success stories, M&S interoperability standards seem to have "hit the wall." In recent years, no break-through has been accomplished. Instead, we look at gradual improvements, but the promised "plug and play" functionality, as suggested by the battery example, is still a dream. What is this wall? In the next section, we will have a look at where we are and how we got there, and this may help to better understand where the current challenge lies.

3 A Brief Historical Overview

In order to better understand the current view on M&S interoperability standards it is necessary to review the history of distributed simulation.

The use of simulators and simulations in the armed forces has a long history, including the use of strategic games, life exercises, and board games. However, with the advance of computers, a new era of computer simulation and simulators began. The birth of simulation standards can be seen with the creation of the Simulator Network SIMNET, which was a project of the Defense Advanced Research Project Agency (DARPA). Developed between 1980 and 1990 in collaboration with DARPA and the U.S. Army, SIMNET showed how to combine individual tank simulators of the Combined Arms Tactical Training System (CATT) to enable tank crews to operate side by side in a common synthetic battle space. The individual simulators represented weapon systems on this common virtual battlefield that had a well defined set of actions and interactions: tanks could move, observe, shoot at each other, exchange radio communication, etc. Individual activities led to status changes that were communicated via status reports. Interactions were communicated via messages.

If two tanks engaged in a duel, the order of activities and the data to be exchanged between these entities were well defined. The shooter decided to engage the victim. He moved his weapon system, and potentially platform components like a turret and a cannon into the best direction, always updating his status, so that other simulators could update their visualization showing that the tank/turret/cannon is moving. He shot at the victim. This data was sent to everyone as well. All observing systems could visualize the shooting (smoke, flash, etc.). The victim also received information on the ammunition shot at him such as velocity, angle, etc. The victim computed the result of this engagement – like catastrophic kill, movement kill, firepower kill, etc. – and communicated the result. All observers, including the shooter, updated their visualization of the victim (like being on fire, smoking, or no effect beside the impact explosion). Based on his assessment of the effect, the shooter could reengage, or continue with a new task. The tasks of who is doing what based on what data was well understood by those simulators embedded as individual independent entities in the common battle space.

As the set of information exchange specifications could be well defined, this resulted in the idea to standardize these messages, which led to the IEEE1278 Distributed Interactive Simulation (DIS) standard: the Protocol Data Units (PDUs) captured syntactically and semantically all possible actions and interactions based on the idea that individual simulators represent individual weapon platforms. Only later, instead of individual platforms also groups and aggregates (like platoons or companies) were accepted as receivers and producers of such PDUs, but these groups were understood as individual entities in the battle space as well. DIS is still successfully used and supported by a large user community.

In parallel to the simulator community that serviced the tactical level training needs, higher commands started to use computer assisted exercises (CAX) to support their command post exercises as well. Ever since Baron von Reisswitz introduced the "Kriegsspiel" during his tenure as war counselor in Prussia in 1811, [see figure 2] combat models were used to train command post officers. These exercise support games had well defined units with well defined interactions, all ruled by very detailed tables enumerating in detail the effects of each possible interaction.

Figure 2: Kriegsspiel (War Game)

The computer based successors also required a distributed capability, in particular to support higher command training of distributed facilities. As the earlier war games, these computer simulations represented aggregates on the operational level, like battalions and brigades. Again, they were interpreted as individual entities on the battlefield. MITRE developed the Aggregate Level Simulation Protocol (ALSP) to exchange information between these simulation systems.

However, unlike the simulator solution, in ALSP several units were represented in each system. When these systems were connected, the protocol ensured that each simulated aggregate had exactly one simulation system that was responsible for updates. In all other simulation systems, the respective aggregate was "ghosted," which means that a simulation object was instantiated in the simulation system, but it was tagged to be controlled by another system and was only used to make decisions for the aggregates controlled by the system, e.g., where to place surveillance radars in the surveillance simulation systems based on the distribution of tanks in the combat simulation system.

As the diversity of aggregates were higher than that of platforms and in addition differed from exercise to exercise, ALSP did not standardize the messages to be exchanged. Instead, ALSP standardized the syntax to be used, but allowed to specify the semantics (meaning of information exchange) in special formats that today would be described as metadata allowing the interpretation of the exchanged data. While during the time of "das Kriegsspiel" the possible units were limited to a set of categories supported by all armies, such as infantry, cavalry, artillery, scouts, etc.), ALSP provided a frame to communicate the participating entities (or better aggregates), possible interactions, and the effects of such interactions.

The High Level Architecture (HLA) was developed to replace both approaches – DIS and ALSP – with a new and merging approach. Originally developed within the U.S. DoD, the final version HLA 1.3 NG was handed to IEEE for international standardization, resulting in the IEEE 1516-2000 and was only recently updated to the HLA evolved standard IEEE 1516-2010. Significantly influenced by recent new methods developed in computer science in general and software engineering in particular, a very flexible protocol was developed providing more flexibility and configurability than both of its predecessors.

The HLA interoperability standard was focused to maximize the flexibility for all kinds of M&S application domains and supported M&S paradigms. The information exchange requirements of a federation are captured in the Federation Object Model (FOM). This model defines all persistent objects and their attributes and transient objects and their parameters that can be exchanged between participating simulations. While persistent objects have to be created and then are updated (and the responsibility can be switched between the participating simulation systems during runtime), transient objects are like messages created in case of need and only used once.

Six service groups are provided as a result of generalizing the synchronization challenges ensuring that all the required information needed is delivered at the right time to the right simulation system. The purpose of *Federation Management* is to determine the federation. Federates join and leave the federation using the functions defined in this group. The purpose of *Declaration Management* is to identify which federate can publish and/or subscribe to which information exchange elements. This defines the type of information that can be shared. The purpose of *Object Management* is managing the instances of shareable objects that actually are shared in the federation. Sending, receiving and updating belong to this group. The purpose of *Data Distribution Management* is to ensure the efficiency of information exchange. By adding additional filters this group ensures that only data of interest are broadcasted. The purpose of *Time Management* is the synchronization of federates. The purpose of *Ownership Management* is to enable the transfer of responsibility for instances or attributes between federates.

HLA significantly increased the flexibility of simulation federation definitions. Instead of being limited to predefined information exchange groups, the developer can specify the objects and interactions and can even support different time model philosophies. It neither assumes the level of resolution nor does HLA assume the partition of the battle space into tactical unit or the phasing of a supported operation. HLA supports component level simulation, platform level simulation, and all levels of aggregation

4 What makes M&S Special?

The last section showed the development of M&S interoperability standards with an increase in flexibility and support of different M&S paradigms. *However, the mental model behind all these developments remained the idea of one shared virtual battle space that was populated by individual independent aggregates and/or platforms that interact with each other.*

These individuals, or group of individuals, were well defined by their own actions and interactions with each other, which could be represented by boundaries around the individuals – or a group thereof – being the boundaries of the simulation system that was responsible for their simulation and the specification of data that could be exchanged via these interfaces. The individual becomes a black box that can represent a simulated system or a live system, as long as the interface specifications are fulfilled. They build a perfect participation of the battle space and what goes on within it.

However, with the introduction of the flexibility provided by HLA, we opened Pandora's Box. While DIS enforced the one shared battle space view by defining syntax and semantics of the PDUs, and while ALSP ensured with the ghost concept that simulated entities are only available once (and merely reflected in other simulation), HLA said farewell to this paradigm.

The interoperability view of HLA is indeed that the same objects are represented in two simulations, and that these objects are represented as the persistent objects in the FOM. If an attribute changes in one of the representing systems, the attribute change is communicated via updates. Nonetheless, we have as many instances of the same object as we have implementing simulations.

As every participating simulation has been developed for a special purpose, it is unlikely that the representations are going to be identical. Actually, it is very likely that the scope will be different, which means that attributes needed to describe the object in one context are meaningless and therefore not even modeled in another context. A simulation system written to support combat operations will use a different model to represent a main battle tank than a simulation system written to support logistics. A radio modeled for support of communications of dismounted infantry will look different than one modeled to be evaluated in the light of electronic warfare.

As all models are simplifications and abstractions of a perception of reality in order to support a certain task, they have to be different. And as simulations are implementations of models, the implemented objects will look different as well:

- Simplification takes things away. Even if we start with a common definition of a real object, we will chop off different aspects of this real object in the process of simplification. Therefore, we end up with different scopes.
- Abstraction in general leads to models with different structures and resolutions. Again, even when starting with identical observations, the detail represented in two models is likely to be different. Even worse, if aggregation is part of the abstraction process, the resulting aggregates may look very different, resulting in different structures.

To show the challenges deriving from abstraction, we already introduced the example of 'number world' and 'letter world' in "Federated Ontologies Supporting a Merged Worldview for Distributed Systems," [7]: a system exposes the six observables *a1, a2, a3, b1, b2,* and *b3*. In letter world, the three observables *a1, a2,* and *a3* are abstracted into attributes of *A,* and *b1, b2,* and *b3* are abstracted into attributes of *B*. In number world, the abstraction of *a1* and *b1* results in One, *a2* and *b2* in *Two*, and *a3* and *b3* in *Three*. Both are plausible models, but they are quite different. While on this lowest level the common attributes are still derivable, supporting the information exchange between the abstractions, what if the resolution for the model is changed and only *A, B, One, Two, a*nd *Three* remain in the models?

Even when starting from an agreed description of reality that comprises all possible attributes that a participating simulation may be interested in, the process of simplification and abstraction is going to produce very different modeling results. Furthermore, not everything going on in the real world referent is observable, even when perfect sensors are assumed. Then it depends on additional assumptions how to model these "hidden" attributes, and as no reference for them can exist by definition, different models may easily result from observing the same system with the same sensors.

It becomes worse when we take the aspect of perception into account. In this paper, perception is the physical-cognitive process of observing reality and building a conceptualization of the observation.

- The physical aspect defines what attributes of an object are observable with the sensoric system of the observer, or more general, the information about the object that can be obtained in the process of perception (this can include gaining insight from literature, discussions with colleagues, etc.).
- The cognitive aspect is shaped by the education and the knowledge of the observer. In order to conceptualize the observation the observer needs to have an internal model he can map this observation to. A physician will see more in an x-ray than a layman. An educated mechanic sees more in an engine than a novice. The subject matter expert has more internal models to explain an observation in his field than others do.

Physical and cognitive perception will therefore shape the model and resulting simulation significantly, even more than simplification and abstraction does, as perception results in a different starting point: We no longer can assume that everybody starts from a common reality, we all have individual perceptions thereof! This common conceptual starting point, however, is the necessary requirement and builds the conceptual foundation for developing a common information exchange specification between simulation systems.

As long as we are starting to support a common theory, like we did in the successful example of a common battle space following the laws of Newtonian physics, we can always track our models and resulting simulations back to the common ground defined by this theory. We can observe with more accuracy, we can model with higher resolution, and we can add "missing" attributes (those that are described in the theory, but not used in individual models). Actually, following the philosophy of science, ***a simulation system is an***

executable hypothesis or – once proven to be valid – an executable theory![1]

Mathematically, a simulation system is a production system representing the theory: starting with the initialization data, we apply production rules encoded as functions, procedures, and methods. Every state that is simulated is a valid state represented by the theory encoded as the simulation. This is equivalent to assigning TRUE and FALSE values to such states: if a certain state can be produced (and we can even add the constraint of 'within a given time') it is true, otherwise it is false. The M&S interoperability challenge comprises the task to ensure the logical equivalence of all representations of an object in the federation.

Again, we can start with assuming that we start from the common ground of a common and accepted description of reality in the form of an object model that can serve as the *Übermodell* from which all simulation representations can be derived by pruning and aggregating. We show in [8] how to apply model-based data engineering to construct the model from the information exchange needs, but this algorithm and similar ones only work if we can assure that all models started from the same common ground. And even then, strange effects can be observed.

To better address the challenges, a formal approach to simulation interoperability [9] has been developed and applied. Without going into the mathematical details, this approach showed significant shortcomings of our current M&S interoperability approaches. From the data modeling theory, we know two categories of dependencies of two objects *A* and *B*:

- *A* is existential dependent on *B* if *A* can only exist if, and only if, *B* exist.
- A is transformational dependent on *B* if *A* needs to be changed if *B* is changed.

None of our current standards support this kind of dependency. We can have a perfect FOM communication in every aspect of *A* and *B*, but we cannot communicate the dependencies. If now two simulation systems implement *A* and *B* identical despite the dependency, we can end up with two versions of truth in the same federation, if we delete or change *B*: in the simulation system that implements the dependency, the deletion or change of *B* implies the deletion or change of *A* as well; but that is not the case in the system that does not implement the dependency. While *A* continues to exist in one federate, it ceases to exist in another, and all under valid current standard conditions.

So far, all of the examples can be understood as examples that someone made a mistake: an important detail was not implemented, a model was over-simplified, an important relation was overlooked, etc. In addition, our focus has been on physical-technical models. As these models have a common referent, this 'real world' can always be used to find out if a model is sufficient or 'realistic.' The assumption here is, however, that truth exists on its own, it is independent of the observer, and reality is separated from the individual who observes it. The traditional scientific method is rooted in this world view called positivism. There exists one world and one truth, and it is possible to find this truth by observation and experimentation. This world view worked well for Newtonian physics and the physical-technical models that model it based on this common ground of a common theory.

However, the M&S community is currently starting to look into better approaches to support human, social, cultural, and behavioral modeling. Davis summarizes his research in as follows: *"Fortunately, the social science literature has a great deal to offer. However, the literature is fragmented along boundaries between academic disciplines, between basic and applied research, and between qualitative and quantitative research. … Realistically, the research base is not mature enough to support a coherent expression of the body of knowledge. The uncertainties and disagreements are profound, on both subject-area facts and even the nature of evidence and the appropriateness of different methodologies. Those hoping to find a nicely compiled body of knowledge that can be used to write computer models will be disappointed. Further, they will often find that there are multiple competing "theories." And, even if a particular "theory" is chosen, it will be found upon inspection to involve numerous variants and uncertainties."*[10] These findings are supported by other researchers as well.

[1] Using the scientific method, a hypothesis becomes a theory only after it has been repeatedly tested and confirmed via real world data using experiments. This is in contrast to the every day use of the term "theory," where it is often understood as a collection of ideas that are not yet proven. In both cases, however, internal consistency is mandatory. In the rest of this paper, we will assume that our models are indeed grounded in theory that has been proven to be relevant and is backed by empiric evidence to avoid having to discriminate explicitly between hypothesis and theory. Whenever this is not the case, it only amplifies the implications of misuse of current practice.

To make things worse for the M&S engineer, we no longer deal with positivism in this domain, but with interpretivism. Interpretivism holds the belief that truth is a construct of the observer. Reality is relative and cannot be separated from the individual who observes it. The majority of social and human sciences subscribe to interpretivism. That means that we have to take the aspect of perception into account when evaluating if two simulation systems can operate together.

If two simulation systems implement competing theories, they can never become interoperable, as the underlying mathematical production systems produce different versions of truth. This does not make one of them wrong or the other solution better. It is a fact of life and no interoperability standard can solve this challenge: we simply do not know, and in some cases even cannot know what is needed to solve the conflict between competing theories. The challenge of the M&S engineer and of supporting M&S interoperability standards is to ensure that no competing theories (and following competing simulation systems) are federated to produce a common federation model.

In summary, our challenges lay often on the modeling side. It is understood that while modeling targets the conceptualization, simulation challenges mainly focus on implementation, in other words, modeling resides on the abstraction level, whereas simulation resides on the implementation level. Our interoperability problems are derived from the abstraction level, but our standards only focus the implementation level.

5 Implications

One of the first things to do about these challenges is to raise the awareness regarding them [11]. It would be naïve to apply standards that were developed for physical-technical models based on a common theory representing the positivistic worldview to integrate socio-psychological models derived from competing theories representing interpretivism and expect valid results. As pointed out in "Towards Methodological Approaches to meet the Challenges of Human, Social, Cultural, and Behavioral (HSCB) Modeling," [10], the best way ahead may be to live with contradicting models. It is highly unlikely that we will be able to address all problems with one common approach based on a common theory resulting in a consistent federation. It is much more likely that the multi-simulation approach based on multi-resolution, multi-stage, and multi-models envisioned by Yilmaz et al. [12] needs to be exploited to support the analysis of these multi-facetted challenges we are faced with as a community.

Generally, it is necessary to focus more on the abstraction level (the modeling) when building federations than on the implementation side. Our approaches to M&S interoperability have been shaped by software engineering and computer engineering principles that are necessary, but not sufficient. The alignment of conceptual constraints is not supported enough by the current approaches and standards. As we are connecting simulated things we need transparency of what we are simulating, as the real world referent use in other interoperability domains has been replaced in the modeling phase by its representing conceptualization in the M&S interoperability domain.

It is worth mentioning that it is possible to apply competing methods in one federation if they are coupled via a common theory. For example, two agents implementing competing theories can be coupled by purely exchanging their actions in the physical world. The underlying conceptual model, however, is well aware that one agent implements one theory, the other agent implements another theory. If we know the agents run into oscillating states or produce inconsistent results, this is part of the underlying common conceptual model that allows for this to happen, as both theories are contained in their agents.

Another aspect is the applicability of current methods for validation and verification to human, social, cultural, and behavioral modeling. As pointed out in the paper, there are many competing hypotheses, and the dearth of real-world data as well as the epistemological nature of simulation forcing us into interpretivism. However, as in interpretivism truth is subjective to the observer and not objective for the observation, validation becomes relative as well. As a consequence, socio-psychological hypotheses may remain in general objectively untestable and cannot graduate into general common theories. This challenge increases with the complexity of proposed solutions and the number of participating hypotheses, resulting in uncertainties and risks adverse to successful application of federated approaches.

The fundamental difference between M&S systems and other software systems is that M&S adds the level of conceptualization to what needs to be aligned. While other software systems connect with the real thing or support the real thing, in M&S systems the "conceptualization is the real thing" that is simulated: the model is the reality of the simulation. If we use technical means to make two simulations interoperable on the implementation level that are based on competing theories, we merge things together that do not belong together, and instead of creating a solution, the result is a conceptual chimera … or worse. However, it is well known that conceptual problems cannot be solved with technical solutions. More work is needed to make sure that the next generation of M&S interoperability standards contributes towards a solution of this category of challenges we are just becoming aware of.

Summary

After all this explanation we still did not have the answer to the question posted in the title of this paper: *How is M&S Interoperability different from other Interoperability Domains?* The answer is simple: M&S ***interoperability requires interoperability of the simulations*** – that is provided by the software engineering standards we focused on so far, including mediation of data representations, conversion of different unit of measures, mappings between different styles of enumeration, etc. – as well as ***composability of the models*** [13]. We have to ensure transparency of our conceptualizations, as they represent the real world references for the simulation. While other interoperability domains connect real things and can refer to the same real world referents, M&S interoperability connects conceptualizations, and we have to understand what the participating systems concepts look like in order to operate together. The same real world referent can have different conceptualization in different models.

The Levels of Conceptual Interoperability Model (LCIM) [14] addresses these issues for some time. Only interoperability domains that are model-driven have the second challenge.

- The battery is plugged into the system and either connects to the socket or does not. As long as power is left it is provided. The battery does not need a model of what it is powering.
- A web service that connects the fill out order for books with the inventory list of Amazon doesn't need a common model: it connects the real list with the real database. Integratability and Interoperability is all it has to be concerned about. The ordered book is either there, or it is not.
- If two simulation systems exchange data, they need to support common concepts of a model. As such, there is a conceptual overlap of the models implemented by the simulation systems. Within this overlapping area, the six interrogatives Who, What, Where, When, Why, and How need to be consistent.

In other words, for the simulation systems, the implemented model is reality. In order to couple two simulation systems, there needs to be an overlap; otherwise both systems have nothing in common to exchange data about. This overlap must be consistent, which means that the results of computations regarding the research questions must be identical. If this is not the case, we end up with two versions of truth in the federation. This problem of model-based reality is unique to M&S. Consequently, the application of software engineering standards cannot solve this problem. Therefore, a new generation of M&S standards needs to support the alignment of models to support and ensure not only interoperability, but also composability, in a form that allows the automation of such processes wherever possible.

This new generation of M&S standards must ensure the transparency of models, not only the mediation of simulations. While standards for real components can focus exclusively on the exchange of data, model-based components must ensure that the same concepts are represented consistently in all participating components. This problem does not occur outside of the model-based world. If the same real world referent is modeled or changed inconsistently in model-based components, this introduces inconsistencies on the conceptual level that are not necessarily observable. While in real components the real world reference exist only once, in model-based components the concept of this one real component can exist independently in every component.

Even more importantly may become the recognition that simulations are implemented theories, as it is the case when human behavior is modeled and implemented. As long as the simulation systems to be federated support consistent theories, the upcoming interoperability challenges can be resolved. In new application domains, such as the emerging domain of HSCB, many conflicting theories exist. This is a conceptual block that cannot be solved by M&S interoperability standards. Federating such models into one common federation must lead to inconsistencies and meaningless results! Instead, alternative uses of alternative theories need to be supported by new approaches like the proposed Multisimulations [12].

This requires a domain of new standard efforts: the efficient and effective support of exploratory analysis under uncertainty and disagreement, and supporting development of strategies that are flexible, adaptive, and robust, as requested by Davis in [10]. SISO should address these challenges in respective efforts.

Although current standards are not sufficient, they are necessary and are building a strong foundation new approaches can extend. The authors made first recommendations in "Conceptual Modeling for Composition of Model-based Complex Systems" [8] and "Using a Formal Approach to Simulation Interoperability to Specify Languages for Ambassador Agents," [9], extending the work presented in [12]. It is now time to focus on building better tools to support the work of the M&S engineer sufficiently well to help avoid mistakes and guide him/her to better solutions in support of the customer not only in the military domain.

Acknowledgement

Significant parts of the underlying research were funded by the US Joint Forces Command Training Directorate and the US Modeling and Simulation Coordination Office.

The authors furthermore thank Hans U. Mair, Johns Hopkins University/Applied Physics Laboratory, for his critical review and constructive remarks regarding theories, validity, and hypotheses.

The paper was originally published as paper 11S-SIW-008 during the 2011 Spring Simulation Interoperability Workshop in Boston, MA, April 4-8, 2011. It has been awarded with a SIWzie and is on the Recommended Reading List of the workshop

References

[1] David Martin, Mark Burstein, Jerry Hobbs, Ora Lassila, Sheila McIlraith, Srini Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan, and Katia Sycara (2004). "OWL-S: Semantic Markup for Web Services," *W3C Member Submission 22 November 2004,* last accessed January 2011 at http://www.w3.org/Submission/OWL-S

[2] Andreas Tolk, Saikou Diallo, and Charles Turnitsa (2006). "Ontology Driven Interoperability – M&S Applications," *Whitepaper in support of the I/ITSEC Tutorial 2548,* VMASC Report, Old Dominion University, Suffolk, VA

[3] Don Brutzman, Michael Zyda, J. Mark Pullen, and Katherine L. Morse (2002). "Extensible Modeling and Simulation Framework (XMSF) - Challenges for Web-Based Modeling and Simulation," *Workshop Report*, Naval Postgraduate School, Monterey, CA, 22 October 2002

[4] Curtis Blais, Don Brutzman, David Drake, Dennis Moen, Katherine Morse, Mark Pullen, and Andreas Tolk (2005). "Extensible Modeling and Simulation Framework (XMSF) 2004 Project Summary Report," *Project Report NPS-MV-05-002*, Naval Postgraduate School, Monterey, CA, 28 February 2005

[5] *Institute of Electrical and Electronics Engineers. IEEE 1278 Standard for Distributed Interactive Simulation,* IEEE publication, Washington, DC.

[6] *Institute of Electrical and Electronics Engineers. IEEE 1516 Standard for Modeling and Simulation* High Level Architecture, IEEE publication, Washington, DC.

[7] Charles D. Turnitsa, and Andreas Tolk (2007). "Federated Ontologies Supporting a Merged Worldview for Distributed Systems," A*ssociation for Advancements in Artificial Intelligence (AAAI) Fall Symposium, Technical Report FS-07-06*, AAAI Press, Menlo Park, CA, pp. 116-119

[8] Andreas Tolk, Saikou Y. Diallo, Robert D. King, Charles D. Turnitsa, and Jose Padilla (2010). "Conceptual Modeling for Composition of Model-based Complex Systems" *in* Stewart Robinson, Roger Brooks, Kathy Kotiadis, and Durk-Jouke van der Zee (Eds.) *Conceptual Modeling for Discrete-Event Simulation,* CRC Press, pp. 355-381

References (continued)

[9] Andreas Tolk, and Saikou Diallo (2010). "Using a Formal Approach to Simulation Interoperability to Specify Languages for Ambassador Agents," *Proceedings of the 2010 Winter Simulation Conference,* B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan (Eds.), IEEE CS Press, pp. 359-370

[10] Andreas Tolk, Paul K. Davis, Wim Huiskamp, Gary L. Klein, Harald Schaub, and James A. Wall (2010). "Towards Methodological Approaches to meet the Challenges of Human, Social, Cultural, and Behavioral (HSCB) Modeling," *Proceedings of the 2010 Winter Simulation Conference,* B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan (Eds.), IEEE CS Press, pp. 912-924

[11] Andreas Tolk (2010). "M&S Body of Knowledge: Progress Report and Look Ahead," *SCS M&S Magazine*, Vol. 1, No. 4, October

[12] Levent Yilmaz, Tuncer Ören, Alvin Lim, and Simon Bowen (2007). "Requirements and Design Principles for Multi-simulation with Multiresolution, Multistage Multimodels." *Proceedings of the 2007 Winter Simulation Conference,* S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, (Eds.), IEEE CS Press, pp. 823-832

[13] Andreas Tolk (2006). "What comes after the Semantic Web, PADS Implications for the Dynamic Web," *Proceedings of the 20th ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS 2006),* Singapore, May 2006, pp. 55-62

[14] Andreas Tolk, Charles Turnitsa, and Saikou Diallo (2008). "Implied Ontological Representation within the Levels of Conceptual Interoperability Model," *Journal of Intelligent Decision Technologies (IDT)*, 2(1):3-19

Authors' Biographies

Andreas Tolk

Andreas Tolk is Associate Professor for Engineering Management and Systems Engineering at Old Dominion University. He is senior member of IEEE and SCS. His email is atolk@odu.edu.

Saikou Diallo

Saikou Diallo is Assistant Research Professor at the Virginia Modeling Analysis and Simulation Center of Old Dominion University. His email is sdiallo@odu.edu.

Jose Padilla

Jose Padilla is Assistant Research Professor at the Virginia Modeling Analysis and Simulation Center of Old Dominion University His email is jpadilla@odu.edu.

Chuck Turnitsa

Chuck Turnitsa is faculty member at the TSYS School for Computer Science at Columbus State University. His email is cturnitsa@gmail.com.

WSC'16

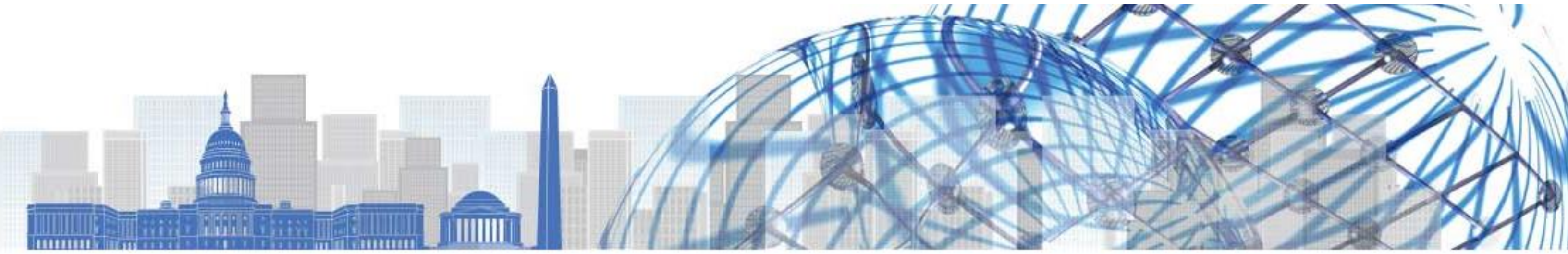
Simulating Complex Service Systems
December 11-14 | Washington, D.C.

Andreas Tolk

Tutorial on the Engineering Principles of Combat Modeling and Distributed Simulation

Proceedings of the 2016 Winter Simulation Conference,
edited by T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick

Arlington, VA, Dec 2016, pp. 255-269



Proceedings of the 2016 Winter Simulation Conference
T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, eds.

TUTORIAL ON THE ENGINEERING PRINCIPLES OF COMBAT MODELING AND DISTRIBUTED SIMULATION

Andreas Tolk

Simulation Engineering
The MITRE Corporation
903 Enterprise Pkwy Suite 200
Hampton, VA 20666, USA

ABSTRACT

This advanced tutorial introduces the engineering principles of combat modeling and distributed simulation. It starts with the historical context and introduces terms and definitions as well as guidelines of interest in this domain. The combat modeling section introduces the main concepts for modeling of the environment, movement, effects, sensing, communications, and decision making. The distributed simulation section focuses on the challenges of current simulation interoperability standards that support dealing with them. Overall, the tutorial shall introduce the scholar to the operational view (what needs to be modeled), the conceptual view (how to do combat modeling), and the technical view (how to conduct distributed simulation).

1 INTRODUCTION

Combat modeling and distributed simulation are very challenging and interesting topics. I have been teaching a graduate course on this topic for several years at the Old Dominion University in Norfolk, VA. Over the development of the course, more and more students not working in combat modeling related domains joined me, as the complexity of challenges of their domains could often be mapped to the topics of this course, that in many aspects became a course for engineering managers and system engineers in charge of complex simulation-based projects. After the first couple of iterations I decided that I needed a textbook that addresses all the various challenges, which with the help of friends who are experts in their related domains I finally finished some years ago (Tolk 2012). Following the experiences made in teaching this topic for a diverse student body, I structured the topic into four sections.

First, I provide a historical context for the domain of combat modeling and distributed simulation. As discussed in (Page and Smith 1998), the military domain has its own language and is often separated from other simulation experts. The second section therefore explains the general concepts by providing the terms and definitions as well as the universe of discourse for combat models and distributed simulation systems. The next section introduces the referential domain of combat modeling. It looks into concepts and models to cope with the situated environment used to describe the virtual battle space, how to move in it, which effects can occur, and what models are used to represent sensing, communicating, and making decisions. In combat models, these often boil down to move, shoot, look, and communicate. The final section deals with the methodological domain of distributed simulation, including discussions of supporting simulation interoperability standards, such as the Distributed Interactive Simulation (DIS) protocol (IEEE 2012) and the High Level Architecture (HLA) (IEEE 2010).

The objective of this tutorial is that the participant will understand the main principles of combat modeling and distributed simulation. He will know the basic algorithms, constraints, and application areas, and the interplay between the different challenges. Through methods from the fields of operations research,

978-1-5090-4486-3/16/$31.00 ©2016 IEEE

computer science, and engineering, participants are guided through the history, current training practices, and modern methodology related to combat modeling and distributed simulation systems. The tutorial intends to provide a comprehensive overview of the engineering principles and state-of-the-art methods needed to address the many facets of combat modeling and distributed simulation addressing the operational view – what needs to be modeled – the conceptional or referential view – how to model the resulting propertied concepts, activities, and effects — and the technical or methodological view – how to implement and use a distributed simulation solution.

2 HISTORY

To better understand the state of the art regarding combat modeling and distributed simulation it is beneficial to know where we are coming from. The military domain has a long standing history of using games to educate their members in strategic thinking. Games like the Indian Chaturanga, the Chinese game Go, and Chess were often played by nobility to prepare the future decision makers for their tasks. It may be of interest that the idea of maneuvers was not really known before the Roman armies trained their soldiers in ”bloodless battles” first documented around 100 B.C. Most soldiers before were ”trained on the job.” The idea of using maneuvers gave the Romans a huge advantage and became a building block for all future military organizations.

Another significant step was conducted by the Prussians. Baron von Reisswitz introduced the ”Kriegsspiel” in 1811. As the war counselor in Prussia he used a representation of the terrain, different blocks representing the different army branches – like infantry, cavalry, and artillery – guided by a rulebook on movement and attrition to educate his officers. His son introduced the idea of paper maps, standardized figures, and better rules in 1824 and was so successful that the Prussian Chief of Staff von Muffling ordered the use of wargames throughout the Prussian Army. The Prussian successes in battle did lead many other nations to adopt wargaming. Major Livermore improved the attrition models in 1883 by incorporating historical data to validate his numbers and tried to introduce the idea of wargaming to General W. T. Sherman, the Commanding General of the U.S. Army. However, still under the impression of the brutal encounters of the Civil War that required many more human factors than could be captured in wargaming, he discouraged the use of this approach by stating: ”Men are not wooden blocks!” This hindered the use of wargames in the USA until the modern days, when nations like Germany and Japan proved the value of this approach on the battlegrounds of two world wars. From the 1930s, through WWII, and on into the Cold War, the armies of the world (including the United States) developed and employed many different forms of tabletop wargaming. The complexity of the games required, often, large staffs of referees, many complicated charts and tables governing the actions units in the game could be ordered to undertake, and complex calculations concerning the adjudication of not only combat, but logistics operations and integrated movement functions (airlifts, littoral landings, etc).

The first modern simulators were flight simulators, starting with the Link simulator of the 1930s (first released by Ed Link in 1929 as a prototype). This was a simple device that was intended to give training pilots a feel of what it was like to handle the controls in a moving platform, before actually attempting to fly a plane. The company is still in existence as L3, and is still in the Flight Simulator business. The first fully instrumented Link unit was sold to US Navy in 1931 for $1,500. Shortly thereafter, the Army took delivery of its Link Trainers in 1934. Not only were they credited with saving vast sums of money and time, they also saved the lives of pilots during training, and after (with the skills taught). According to a report to the US House of Representatives, these trainers were estimated to have saved the Army Air Corp at least 524 lives, $129,613,105 and 30,692,263 man hours in one year.

In the 1960s as computers became more powerful, and the explosion of ideas concerning input and output devices began, the idea of visualizing data came back around, and this time it was taken seriously. All along this time, vehicle simulators became more and more complex and realistic. Finally, in the 1970s and 1980s the first ground vehicle simulators became available.

In parallel to the simulators, simulations became more powerful as well (definitions for both will be given in section 3.1). They represented more and more complex combat situations in more and more challenging terrains to decision makers in the headquarters in so-called computer assisted exercises (CAX) (Cayirci and Marincic 2009). Constructive simulation systems stimulated common operational pictures and received orders from the command and control systems. The supporting de facto standard was the Aggregate Level Simulation Protocol (ALSP). This protocol supported a world-wide federation of systems in the USA, Germany, and Korea and was very successful. To close the gap between the simulator and the simulation world, the High Level Architecture (HLA) was introduced and internationally standardized as IEEE1516.

Today, the use of simulators and simulations in common federations is the rule for military training and education. The biggest and most expensive exercise may be Millennium Challenge 2002. It brought simulators and simulations from all over the nation together for a three week long event (July 24–Aug. 15, 2002), was sponsored by U.S. Joint Forces Command, and has been estimated to have cost approximately $250 million.

3 GENERAL CONCEPTS

The concepts, terms, and scenario elements of combat modeling were introduced to the Winter Simulation Conference in (Page and Smith 1998). One of the particular challenges in this domain are the military terms and abbreviations, but also the special terms used often uniquely in the simulation descriptions. We can only deal with a very limited subset here and have to refer to additional literature for the interested reader.

3.1 Terms and Definitions

We already used several terms in the last section that may not be familiar to a scholar or researcher of M&S in other application domains than defense, but some concepts are also shared. In this section, the main terms and definitions of concepts are introduced. This is not an easy endeavor. Tuncer Ören compiled a lexicon of thousands of M&S related terms (Ören 2011). In the context of this tutorial, the Glossary of Military Terms (U.S. D.O.D. Joint Staff 2010) as well as the Department of Defense (DoD) Modeling and Simulation Glossary are of particular interest.

Models are target driven, purposeful abstractions and simplifications of a perception of reality. The perception will be shaped by cognitive, physical, and legal constraints. *Simulations* are the execution of models over time, in many cases using computers to execute a programmed version of the model to do so. If the resulting device is used to provide stimuli and feedback to an individual or a group of trainees, this device is a *simulator*. Typical examples are driving simulators or battle simulators that provide a realistic virtual environment in which individuals or crews can train. If a system explicitly provides stimuli for a predefined target system in a predefined structure via predefined interfaces, we talk about a *stimulator*. They are often used to generate test cases for new system, e.g., to check if a new battle command system can handle the required number of incoming messages as specified.

In the same context, *live, virtual, constructive (LVC) simulation* is defined. The easiest way to understand the three concepts is to look at people, systems, and the operation. If real people use the real systems to participate in a simulated operation, then we are talking about live simulations. If real people use simulated systems or simulators to participate in a simulated operation, then we are talking about virtual simulations. If simulated people use simulated systems to participate in a simulated operation, then we are talking about constructive simulations. For military training events it is often advantageous to include all three types to support the needs of the trainees.

The model hierarchy of military simulations is often depicted as a pyramid. On the top, theater/campaign level models allow to analyze force structures or force designs and provide training for high–level decision makers and their staff. The next level are mission or battle level models that are used for doctrine, mission planning, force employment, and force modernization. Many CAX events are also supported by this level of models and simulation systems. The difference between theater and mission level is the scope of the simulated engagement. The activities to defend Western Europe during the Cold War or the military activities conducted within the Operation Iraqi Freedom are theater level, while main events like the 1942–1943 Battle for Stalingrad or the 2003 Battle for Baghdad are examples for the mission level. Tactical improvements as well as weapon system level engagements are covered by engagement level models representing one on one or many to many duels. Finally, the engineering level provides high-resolution support for Research & Development of weapon system components. The effect of new ammunition or new types of armor are simulated on this level.

Another set of terms used to describe the different resolution levels are *entity level simulation* versus *aggregate level simulation*. On the entity level, high resolution models are used representing individual weapon systems, often simulating all military processes individually. If several systems are combined into a unit that becomes the simulated entity, we talk about aggregated simulation. If only weapon systems of the same type are aggregated, we call this a homogeneous unit, otherwise it is a heterogeneous unit.

One of the main challenges in decision making is coping with uncertainties and minimizing the associated risks. If a simulation does not contain any random parameters and always produces the same output for a given input, it is *deterministic*. If probabilistic components are used to represent not only point estimates but to generate variations in the simulation following the laws of statistics, the simulation becomes *stochastic.*

When addressing the universe of discourse for a simulation system, we have to address *scope, resolution, and structure.* Resolution answers the question of how detailed something is modeled in the simulation system. The more detail is added, the higher the resolution. Scope answers the question about what is represented in the simulation system. What has been recognized as important in the viewpoint of one simulation may have been seen to be neglectable in another simulation system. Structure describes how observed details are grouped into concepts. The same attributes can be used to describe different concepts in two simulation models, resulting in different structures that are used to categorize the observations of the real system. These terms are often subsumed under the challenge of multi–resolution modeling.

The next three terms are often confused as well: *fidelity, resolution,*and *credibility.*Fidelity of a simulation is the accuracy of the representation when compared to the real world system represented. A simulation is said to have fidelity if it accurately corresponds to or represents the item or experience it was created to emulate. As discussed above, resolution of a model or a simulation is the degree of detail and precision used in the representation of real world aspects in a model or simulation. Credibility is the level of trust of the user of the model. This level can vary and is user dependent as well as application domain dependent. Credibility is the quality or power of inspiring belief, or the capacity for belief.

The engineering methods of *verification, validation, and accreditation (VV&A)* help determining if a simulation is correct and usable to solve a given problem. Validation is defined as the processes of determining the degree to which a model or simulation is an accurate representation of the real world from the perspective of the intended uses. The scope is therefore the behavioral or representational accuracy. It answers the question: Did we build the correct simulation? Verification is the process of determining that a model or simulation implementation accurately represents the conceptual description and specifications. The scope is transformational accuracy. It answers the question: Did we build the simulation correctly? Accreditation is the official determination that a model or simulation is acceptable to use for a specific purpose. While this is required for military simulations in the USA, other nations are often talking about acceptance and do not apply the same formal process. A good overview of the state of the art for general V&V is given in (Sargent 2013).

3.2 Scenario Elements

In this subsection, some military terms are introduced to describe the elements that most likely will make up a scenario. In order to support the military, the simulation engineer has to understand what the mission is (the big picture), what capabilities are required to conduct the mission successfully, what relations are needed to conduct the tasks in an orchestrated manner, what system can be modeled that provide the needed capabilities as well as the communication, and what the time constraints are. In other word, what is needed for an effective and efficient conducting of the mission?

It is pivotal for the simulation engineer to be aware of the harmonization and alignment principle that addresses the triangle of represented concepts, internal decision logic, and external evaluation logic. It is obvious that if we want to evaluate something, it needs to be modeled. But how much detail is enough? The question can only be answered in collaboration with the sponsor, but every detail needs to play a role in the internal decision rules and must be considered in the external evaluation for success, the measures of effectiveness – how well are needed function performed – and measures of performance – how much do they contribute to the success of the mission. If the internal decision rules are not aligned with what is evaluated as a success externally, we are wasting resources. If the focus lies on different attributes of the concepts, discontinuities will be observed. It is the task of the simulation engineer to avoid this for all scenario elements.

3.2.1 Land Components

Land based operations, such as conducted by the army, are characterized by the distribution and range of the weapon systems, sensors, and communication means, or aggregations thereof. Typical weapon systems comprise

- Infantry is made up of soldiers, sometimes modeled as squads, with handheld firearms, like rifle, machine guns, or even anti-tank missiles. They may be protected by body armor, but, in general, are soft targets that should avoid direct fire without protection.
- Infantry transportation is provided by off-road capable vehicles, like Jeeps or High Mobility Multipurpose Wheeled Vehicle. They are normally not armored and have only light weapons, like mounted machine guns.
- Armored Fighting Vehicles, sometimes referred to as Infantry Fighting Vehicles or Mechanized Infantry Combat Vehicles, are light tanks designed to carry infantry into battle and provide fire support for them. They carry several soldiers that may be able to engage in battles while being in the tank and have light to medium weapon systems for direct fire.
- Main Battle Tanks carry the main part of direct fire battles. They have strong armor and heavy weapons.
- Mortars are high-angle-of-fire weapons that fire ammunition in a high angle so that it falls onto the enemy. Mortars come in several sizes, from small mortars that can be used and carried by infantries to bigger mortars that are part of the artillery.
- Main artillery systems are howitzers and rocket launchers. Howitzers can be towed or self-propelled. As a rule, Howitzers fire ammunition while rocket launchers, often MLRS, launch self-propelled rockets, but there are exceptions for modern howitzers.
- Army aviation focuses most often on helicopters, often referred to as rotary wing air craft, but also uses fixed wing air craft. These are used mainly for transportation and air based fire support.

These systems are aggregated into combat or maneuver units, fire support units, combat engineers, air defense units, and aviation units. Headquarters, communications and networks, and logistics and supply are additional challenges to cope with.

3.2.2 Air Components

Air components are as complex as land operations, but due to the high speed of their operations, the focus lies more often on the individual events connected with aircraft in the sky, so called sorties, than the overall number of entities available. Many models of air operations are therefore more activity driven than there land-based cousins focusing on the entities and objects instead. However, their typical weapon systems include fighters, bombers, and many special task platforms.

- Fighters are highly maneuverable, but often short range aircraft. They engage hostile fighters and escort own bombers to protect them from air attacks.
- Bombers are less maneuverable, but usually have a long range. They are mainly designed to attack ground targets or sea targets dropping bombs or launching shorter range missiles.
- Transportation aircraft in various sizes provide the means for air lift operations.
- Drones are unmanned air vehicles that are controlled remotely for surveillance as well as combat operations.
- Command and Control and Intelligence aircraft provide all kind of means for command, control, communications, and sensing to air and ground forces. Long range surveillance and Airborne Warning systems belong to this group as well.

Many modern aircraft, in particular strategic bombers and intelligence platforms, are stealth platforms that are nearly invisible to radar observers. Due to the technical nature of air warfare, the available capabilities on the ground – ensuring a quick turn-around maximizing the number of sorties – are also important and often make up a significant part of the model.

If space-based entities, such as satellites, are modeled, they often fall under the lead of the air forces, but with increasing importance, more and more simulation systems are introduced with focus on this new element of warfare.

Another topic that traditionally was covered under air operations but deserves its own group of models by now is the ballistic missile defense. In particular nuclear components and intercontinental missile defense are topics covered in models focusing on these topics.

3.2.3 Naval Components

Navy warships are complex and expensive, and are rarely built in large numbers. For models, this provides a special challenge, as even if two ships belong to the same category, they may still have clearly distinguishable capabilities reflecting the technical state of the art when they were built.

Naval forces conduct surface operations, underwater operations, and littoral operations. They can provide massive fire power by naval artillery, including missiles, as well as air power by naval air forces. They engage in sea mine warfare against surface and underwater vessels, actively as well as passively.

There are many vessel types - giving the caveat already mentioned – such as

- Aircraft carriers that are deployable air bases on the sea.
- Battle cruisers and battle ships provide the artillery firepower and missile launching capability of naval force.
- Frigates and corvettes are used to protect battle ships and aircraft carriers, in particular against opposing submarines. Special submarine hunters focus exclusively on battles against enemy submarines.
- Destroyers and cruisers fulfill a similar role as frigates and corvettes, but their main weapon system is the torpedo.
- Tenders provide logistic and maintenance for the navy and the systems.
- Submarines are used for underwater warfare.

Coast guard operations usually fall under another jurisdiction than navy operations. As their tasks – in particular in peace times – are very different from navy tasks, they have to extend navy models to cover tasks like drug interdiction, alien migration interdiction, fishery violation, and search and rescue operations.

3.3 Supporting Guidelines

The North Atlantic Treaty Organization (NATO) Code of Best Practice for Command and Control (C2) Assessment (Alberts et al. 2002) and the Technical Cooperation Program (TTCP) Guide to Experimentation (Labbé et al. 2006) are both guidelines helping the simulation engineer as well as the operations research expert to conduct better simulation-based experiments and analysis.

3.3.1 NATO Code of Best Practice for C2 Assessment

The NATO Code of Best Practice for Command and Control Assessment (COBP) was produced to facilitate high quality assessment in the area of C2. The COBP offers broad guidance on the assessment of C2 for the purposes of supporting a wide variety of decision makers and C2 researchers. The COBP presents a variety of operations and operational research methods related to combat modeling that can be applied and orchestrated in support of analysis and evaluation of C2 related research questions. As such, it is a best practice guide on how to apply various means of operations research within a combat modeling related study. Furthermore, the COBP is the product of international collaboration that has drawn together the operational and analytical experience of leading military and civilian defense experts from across the NATO nations. It represents the common understanding on how to conduct good C2 research within a coalition. In summary, the COBP enhances the understanding of best practice and outlines a structured process for the conduct of operational analysis for C2. It shows how to structure a study that utilizes combat modeling and distributed simulation. It can be downloaded without cost.

3.3.2 TTCP Guide to Experimentation GUIDEx

TTCP is an international organization that collaborates in defense scientific and technical information exchange, program harmonization and alignment, and shared research activities for the five nations: Australia, Canada, New Zealand, United Kingdom, and the USA. The TTCP Guide for Understanding and Implementing Defense Experimentation (GUIDEx) provides critical guidance to support successful defense experimentation. It has been produced by defense experimentation expert representatives from the defense science and technology (S&T) organizations of these nations. Like the NATO COBP, it is distributed without cost.

The main objective of the GUIDEx is the application of scientific principles to conducting defense experiments. It emphasizes the need for frequent communication with stakeholders and utilizing integrated teams to conduct the work under observance of ethical, environmental, political, multinational, and security issues.

4 COMBAT MODELING

Modeling is the task-driven, purposeful simplification and abstraction of a perception of reality that is shaped by cognitive, physical, and legal constraints. The following subsections will cope with the combat modeling challenges that have to be addressed in every defense related model. In addition to (Tolk 2012), these section utilizes (Deitz and Edwards 2009) and (Strickland 2011).

4.1 Modeling the Environment

The environment is often assumed to be implicitly given, but it deserves as much attention as every other simulated entity. The reason is that the common battle space or battle sphere actually is situated, i.e., it builds the common foundation for how elements move, sense, act, and communicate. If the resulting virtual battle space differs significantly in two combat models that are linked or federated, the results will likely differ significantly as well. If the resulting simulation systems are composed, the result will be an unfair fight: a systemic bias that may be rooted in the different representation of the environment.

The environment comprises everything required by the simulated entities. If these are land entities, the modeling of terrain, cover, surface, etc. is of particular interest. For air force entities, clouds and wind and different temperatures in different air layers are important. For a sea based entities, current and salinity can be as important as the sea state – i.e. height of the waves. The ocean itself has a climate that is for submarines as important as the climate of the atmosphere is for the aircraft. So-called space weather influences satellites and need to be modeled when these entities are needed. In summary, everything that is perceived to be important for the defense domain should be included and not be simplified or abstracted away. If a certain attribute of the environment is needed, it has to be captured and modeled somewhere.

To give an example, modeling the environment for land-based entities is more than just using the terrain elevation. Many other factors are often needed, such as terrain roughness, the degree of urbanization and/or forestation, the vegetation and soil type, rivers, roads, and bridges, natural and man-made obstacles and barriers, precipitation, weight bearing capacity and many more details. Questions like ”Does the season make a difference for the model, as trees and bushes may have leaves or not?” have to be asked and answered.

For modeling the terrain, as a rule cells are used that capture the various properties, such as height or cover, as attributes. To compute the influence of the terrain on the current status and activities of a simulated entity, the attributes of the cell the entity is in as well as the attributes affected by the activity have to be considered in the computation. There are only three regular shapes that can be used to cover a plane without leaving gaps (excluding Escher–like irregular shapes): triangles, parallelograms, and hexagons. Triangles have the advantage that three points define a plane so that an area approximated by triangles can easily be visualized without any gaps in the representation of elevation, as long as the elevation is stored in the corners of the triangles. Parallelograms - with rectangles being a subgroup - allow the approximation of the terrain via a chessboard like structure of cells, allowing in particular the use of Cartesian grids, such as they are used in military maps. Hexagons allow the use of the advantages of triangles (by storing the elevation in each of the six corners as well as in the center, thus using de facto six triangles to represent the elevation), plus they provide for computation friendly definitions of distances: when numbering the hexagons accordingly the distance of two hexagons can be derived by simple additions and subtractions, avoiding the computationally intensive multiplications and square-root operations needed in Cartesian grid systems.

For all these aspects, we also have to understand if we are adding detail in support of the simulation or of the visualization. In the ideal case, every detail should be considered for simulation as well as visualization, but this practice is not always followed in real-world applications. It is good practice to understand visualization as a special form of external evaluation and apply the *Harmonization and Alignment Principle:* only if an attribute is used for the internal decision process it should also be visualized. Otherwise, we can easily become guilty to work with ”smoke and mirrors” as our simulation visualization presents a different picture than the one used by the simulation itself.

This approach of using grid cells to capture the main characteristics is comparable to creating a ”game board” on which the simulated entities are acting. Moving, sensing, and acting is guided by rules, similar to those being created for war games, just that they are captured as algorithms and are parametrized for more accuracy. However, the mental picture of the game board with the simulated entities being the figures influences many of the following algorithms, and even standard development.

4.2 Modeling Movement

Modeling movement is strongly connected to the environment. Movement can be modeled implicitly or explicitly. Implicit mobility models outsource the computation of all mobility factors to produce mobility maps that are used to look-up the possible speed at runtime. Explicit mobility models use the mobility relevant properties of the simulated entity as well as of the relevant parts of the environment to compute the speed on the fly. The list of attributes can be rather impressive. The Army mobility model (AMM) utilizes the following vehicle attributes: vehicle weight, vehicle geometry (in particular ground contact geometry), vehicle power characteristics, dynamic reaction to obstacle impact, vehicle braking characteristics, front end strength, dynamic reaction to rough terrain, and the driver's tolerance to longitudinal shock. In addition, the following environmental characteristics are use to compute the speed: surface type, surface strength, surface roughness, slope, season, precipitation form (rain, snow), precipitation amount, obstacle geometry, obstacle spacing, vegetation size, vegetation density, and visibility characteristics.

Another important factor is the current task conducted by the simulated entity. If simply in transfer mode, cloud cover is no big issue for modern aircraft, but if visual contact is required to fulfill a mission, this can be a major slow-down factor. Similar observations are true for land forces as well: the same system in the same terrain will move differently when it simply moves into a new assembly area or when it is looking for enemies hiding in the terrain.

When modeling movement explicitly, point movement is often used to take these different aspects into account. Attributes of the system are used to compute a number of points that it can use to move in the current simulated time step. The attributes of the environment are also used to compute resistance points. How far a system can move is then defined by the point values. This simple approach allows to add tactical resistance values to terrain cells: if land mines are laid, it increases the value; if hostile systems can shot at you in a certain cell, it increases the value; if artillery shoots into a cell, it increases the value; etc. The total resistance value is then the sum of environment and tactical resistance. Optimization algorithms can now be used to compute the path of least resistance, providing simulated entities with the ability to move using artificial intelligence to behave tactically appropriately.

When using models that aggregate several weapon systems into a unit, or even several units into a higher unit, these point algorithms have to be modified. Usually, the unit schema are used that prescribe the distribution of systems – or units – within this schema. Typical arrangements on the tactical level are lines, columns, or wedges. It is common practice to select a reference system within this schema, often the leader of the formation, that is used to compute the movement. All other systems follow accordingly. The schema is often used to present a tactical standard schema that needs to be adopted to the current terrain constraints. For higher aggregates, shapes like circles or rectangulars are often used. To avoid model artificialities, the tactical schema or often adapted to the terrain.

Finally, the results of a combat situation may influence the speed and movement as well. In particular casualty numbers that usually slow the speed down. The casualty rate is another factor. The way these factors influence the speed may be highly dependent on other states of the unit: a veteran unit may slow the operation down to get a better idea of what is going on while a new unit may panic and rush. In an open terrain, the unit may run for cover, etc. These are decisions the simulation engineer needs to make with the user of the model. In any case, all these factors and their effect need to be captured and documented for validation.

4.3 Modeling Sensing

The easiest way to model sensing is to give simulated entities full access to all the information available in the model: the ground truth. In reality, weapon systems and units do not see and know everything. Their decision is based on a perception of their situation that is incomplete and inaccurate. The more information is provided via communication with other units and the better the results are that are observed by its own sensors, the better is the perception of the unit.

There are many types of sensors: acoustic sensors, like microphones or hydrophones, that listen for sounds in the environment. Chemical sensors that identify chemical and biological substances. Electromagnetic sensors observing changes in the electrical and magnetic field.Thermal sensors, such as infrared sensors, utilize changes in heat. Optical sensors observe the visible spectrum of the electromagnetic spectrum. For all these sensor types, the target–background–ratio is pivotal: if the signal of interest is overshadowed by the same or very similar signals from the background, it can hardly be detected: a weak sound in a noisy environment cannot be heard, a chemical agent that smells just like the environment does cannot be detected. The reason for using camouflage is to blend optically into the environment, etc. Therefore, in order for a sensor to detect a target, three requirements have generally to be fulfilled:

- The sensor has to be able to detect a certain property or a combination of properties (like an infrared spectrum)
- The target exposes at least one of the observable properties (like giving out heat in the detectable infrared spectrum).
- The background does not expose the same observable property or at least is significantly different (the environment is colder than the target).

Not fulfilling one requirement prevents detection of the target. This requires, however, that important attributes observable for weapon systems or units must not only be modeled for the targets, but also for the environment. If we don't know how hot the environment is, we cannot determine if an infrared sensor is effective, etc.

The steps of creating a perception are normally observing the assigned area, detecting that something is present, tracking the movement of this object, classifying the type of the detected object, recognizing whose side the object is on, and identifying the details. In combat, this usually leads to target acquisition.

Line–of–sight algorithms play a special role, as they define if two systems can see each other or whether an obstacle is in the way. To save computing time, they are often used in advance to produce visibility maps that provide the information which environmental cells can be observed from the current one.

Radar and sonar models are more complex than line–of-sight applications and take many additional factors into account, like transmitted power, the gain factor of the antenna, cross–section of the target radar, noise and temperature of the radar system, and more. High resolution models also take the earth curvature into account and compute reflection characteristics of the observed waveform in the observed environment.

There are many options utilized for modeling sensing, from simple cookie–cutter function to computationally expensive high–resolution models of ray tracing and wave distribution. For the simulation engineer it is therefore important to capture and document all these aspects to avoid unfair combinations of applied sensor models that create a systemic bias of this composition, e.g., if one model includes a sensor that can penetrate an environmental obstacle while the other model simple uses line–of–sight based perceptions. It is also important to understand which attributes are used to create a perception and what values for them have what effect on the modeled sensors.

4.4 Modeling Effects

Although there are many effects on the battlefield, the main effect looked for is attrition of the opponent. Most combat models on the entity level are looking at the probability of hitting the target, i.e., how accurate is the shot, and the probability that the hit kills the target, i.e., how efficient is the ammunition used against the armor of the target. To compute the effects, models distinguish between direct fire weapons, that require a line–of–site between shooter and target, and indirect fire weapons.

The standard formula used for direct fire weapons is:

$$P_k = P_{k|h} * P_h \tag{1}$$

P_h is the probability to hit the target with the current shot. The conditional likelihood to destroy a target when it is hit is $P_{k|h}$. The resulting probability to kill a target with a given shot is computed to P_k. If you shot more than one shot at the target, the overall probability to destroy combines with n shots to $P_k^n = 1-(1-P_k)^n$. These n shots can result from one shooter shooting n times in a short period of time or from n shooters shooting at the same target. However, for salvos, like machine gun fire, another formula is used.

Indirect fire weapons compute the effect by the lethal area of one shell A_l compared to the overall target area A_T. One shell destroys a target in the target area with the likelihood of $P_k = A_l/A_T$, A salve of *n* indirect fire shells targeted at the same area computes the likelihood to destroy a target in the target area to $P_k^n = 1-(1-P_k)^n$.

Not every shot destroys the target completely. In many combat models, the following damage classes are defined: fire power kill, movement kill, communication kill, and catastrophic kill. The catastrophic kill is a total loss of the target, the other categories are self explaining. The probability computations for these events are equivalent to those described above.

Some models alternatively use a game-based point system to compute if a system is destroyed or not. Every system receives a certain point level in the beginning, and every duel reduces the points while maintenance can increase the points. If the points fall under a certain threshold, the system receives the related damage.

For aggregated combat models, the so-called Lanchester equations are still used to compute attrition of forces. Frederick W, Lanchester formulated them in 1916 to show the usefulness of force accumulation in modern warfare. He looked at units as force collections that mainly decrease the number of opponents within duels while simultaneously being decimated by them as well, both based on attrition coefficients depending on the duel situation. This view results in differential equations describing the battle and the number of forces to be expected on both sides over time.

In direct fire, the amount of destroyed targets on the blue site dB depends only on the number of red shooters at the given time $R(t)$ times the red Lanchester coefficient l_r. The same is true on the opposing site as well. To solve the differential equation for $(B(t)$ and $R(t)$ at any given time, we need to know the initial force numbers B_0 and R_0. The result is the so–called square law for direct fire attrition:

$$l_b[B_0^2 - B^2(t)] = l_r[R_0^2 - R^2(t)] \tag{2}$$

In indirect fire, the number of destroyed targets is proportional to the amount of targets in the target area as well as the amount of shooters shooting into this area. Therefore, the amount of red losses depends on the number of blue shooters, the number of red targets, and the attrition coefficient: $dR = l_b B(t)R(t)$. The blue losses are computed equivalently. Resolving these differential equations results in the linear law for indirect fire attrition:

$$l_b[B_0 - B(t)] = l_r[R_0 - R(t)] \tag{3}$$

The military operations research community derived many additional Lanchastrian equation to support the analyses of attrition. Coefficients were derived analytically as well as empirically. Although often criticized for the many assumptions and constraints, these equations are still in use. So far, no alternative with a similar solid mathematical foundation has been agreed upon.

4.5 Modeling Communications and Decision Making

We already learned about the importance of communication in the creation of a perception, which can be highly improved if information from trusted sources regarding the current situation are received by the unit or weapon system that needs a better situational awareness. Generally, communication between systems and units is pivotal to exchange information and orders between superior and subordinates. Information is also often exchanged between neighbored units. The command and control structure between units is the main guide when setting up theses communication channels. In particular for distributed planning, the communication of operational orders became increasingly important. While many CAX system still outsource the decision making and planning to the training audience, constructive simulations become more and more sophisticated in modeling command, control, and communication of the related pieces of information.

When modeling the communications explicitly, line–of–sight models coupled with range models are still an often used option. If two communication device can share information, like radios working on the same frequency, and they are within range and connected via line–of–sight, they can communicate. More detailed simulation models capture for each information exchange requirement the necessary communication means, the required or usable channels, the required bandwidth, and capacity and time constraints. If more than one option exists, optimization algorithms can be used to compute the best use of all communications means.

Some aggregate models assume perfect connectivity, but allow for time delays. Other models use the connection probability to compute if a message makes it through or not. More and more models explicitly model network communication models, such as the Optimized Network Engineering Tools (OPNET) model group. Newer concepts, like airborne networks, or digital radio based tactical Internet options, require new models that are more and more shared with industry, as they are used for cellphone coverage, etc., as well.

5 DISTRIBUTED SIMULATION

The last sections gave an idea about the multitude of options to model the environment and the entities, and how they move, look, shoot, and communicate in their virtual battle space. It is already a challenge to ensure consistency in a single model, but this challenge increases when several independent simulation systems shall be federated to support a common training event or some analytic activity. The two subsections of this section will first address some general challenges of distributed simulation and then have a short look at supporting interoperability standards.

5.1 Challenges of Distributed Simulation

This subsection focuses on what tasks a simulation engineer will face when executing distributed simulations where independently developed systems are performed on autonomous networked computers supported by information exchange models and protocols that govern the exchange of information between these simulation systems. The tasks of a simulation engineer in this context in general can be summarized as follows:

- Selecting the best simulation systems in support of the task,
- Composing the simulation systems into a federation,
- Exposing the information needed by other simulation systems conform with the selected interoperability protocol,
- Integrating the information provided by other simulation systems via the interoperability protocol into the respective receiving simulation systems,
- Avoiding inconsistencies, anomalies, and unfair fight situations,
- Addressing additional issues regarding multiple interoperability protocols that are used within the federation,
- Ensuring that all simulation systems and information exchange models are initialized consistently,
- Ensuring that all information needed can be exchanged via the supported information exchange models and interoperability protocols during execution.

We will first look at the various roles that simulation systems and the runtime infrastructures have to play before we look into commonalities and differences of interoperability and composability.

5.1.1 Simulation Systems and Runtime Infrastructures

The main reason for building a federation is the coupling of functionality of contributing systems to provide a new capability. To allow this, the common entities, events, and state changes represented in participating simulation systems must be represented in both systems consistently and synchronized, that means the challenges of temporal and mapping inconsistencies must be addressed, as discussed later in this chapter. To this end, the infrastructure that supports the interoperability protocol and the information exchange model must support three requirements: (1) All information exchange elements must be delivered to the correct simulation systems (effectiveness); (2) Only the required information exchange elements must be delivered to the simulation systems (efficiency); and (3) The delivery must happen at the right time (correctness).

Synchronizing time and avoiding time anomalies is one of the most challenging tasks. It is not surprising that many solutions focus on real–time solutions that do not require a complex time-algorithm ensuring consistencies of temporal cause–effect chains in multiple time representations.

Just adapting an interface to a protocol is generally not sufficient to prepare an interoperable solution. No matter how we create the federation, the individual simulation systems must be able to fulfill a set of tasks as well. This needs to be supported by the design of the simulation system from the beginning:

- All information that needs to be provided from the system to the federation needs to be retrieved and mapped to the protocol used.
- All information provided by the federation to the simulation system needs to be read from the protocol and mapped to internal representations.
- The simulation system must consider in its algorithms which information it can change and it needs to update, and which information is owned by another system and only represented for awareness.
- The simulation system must be able to set its time in accordance with the supported protocol, actively and passively.

5.1.2 Interoperability and Composability

The M&S community understands interoperability quite well as the ability to exchange information and to use the data exchanged in the receiving system. *Interoperability* can be engineered into a system or a service after definition and implementation. Alternative data representations can be mediated into each other as long as the constraints are understood. Only when data have to be disaggregated (which requires that the information that got lost in the aggregation process be reinserted) the engineer has the problem from where to extract this needed information, but often heuristics can be applied that lead to satisfactory results.

Composability is different from interoperability. Composability is the consistent representation of truth in all participating systems. It extends the ideas of interoperability by adding the pragmatic level to cover what happens within the receiving system based on the received information. In contrast to interoperability, composability cannot be engineered into a system after the fact. Composability requires often significant changes to the simulation to ensure that a research question is either answered equivalently in all participating simulation systems, or it is not answered at all. Inconsistent versions of truth are not allowed.

5.2 Interoperability Standards

Two standards have been developed for simulation interoperability that will be described here. Many alternative options are possible, like using not standardized, but internationally successfully applied solutions as described in (Powell and Noseworthy 2012), or using more general solutions like semantic web methods, but describing these options goes beyond the context of this tutorial.

5.2.1 IEEE 1278: Distributed Interactive Simulation (DIS)

The IEEE 1278 Standard for Distributed Interactive Simulation (DIS) evolved from the SIMNET project of DARPA. There are five volumes: IEEE 1278.1 – Application Protocols; IEEE 1278.1A – Supplement to Application Protocols: Enumeration and Bit-encoded Values; IEEE 1278.2 – Communication Services and Profiles; IEEE 1278.3 – Exercise Management & Feedback (EMF): Recommended Practice; and IEEE 1278.4 – Verification Validation & Accreditation. Of particular interest for this tutorial are the enumerations that standardize the so–called Protocol Data Units (PDU) used to exchange information.

DIS was mainly developed to support simulators. They are connected via a network supporting the application protocol, such as an Ethernet token ring. The PDUs are broad–casted from the sending simulator to all other simulator. If they can use the information, they do so, otherwise they ignore the data package. The PDUs are standardized to the bit level. Each PDU comprises of a header and the pay load. The header allows the receiving simulator to decide if this data is of interest. They payload comprises the information describing details on the originating and receiving entity and the type of event. There are 50 types defined, such as fire, detonation, and collision events, but also transmitter, designator, and signal events. Some PDUs allow to create new objects or delete objects no longer needed.

The general characteristics of DIS are the absence of any central management; all simulations remain autonomous and are just interconnected by information exchange via PDUs; each simulator has an autonomous perception of the situation; cause-effect responsibilities are distributed for the PDUs to minimize data traffic. There is no time management or data distribution management. The PDUs are transmitted in a ring or on a bus and each simulator uses PDUs that are directed at one of his entities.

5.2.2 IEEE 1516: High Level Architecture (HLA)

The IEEE 1516 Standard for Modeling and Simulation High Level Architecture is defined by three core volumes, all updated in 2010: IEEE 1516 – Framework and Rules; IEEE 1516.1 – Federate Interface Specification; and IEEE 1516.2 – Object Model Template (OMT) Specification. In addition, the IEEE 1730-2010 - Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP), augmented by the IEEE 1730.1-2013 IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process Multi-Architecture Overlay (DMAO), are of interest, as they define rules and guidelines on the development of standardized simulation federations.

HLA was developed to unify various distributed simulation approaches within the US DoD and has been adopted by NATO as well. The objective of defining the HLA was to define a general purpose architecture for distributed computer simulation systems. It defines a federation made up out of federates, which are the simulation systems, and the connection middleware that allows the information exchange between the simulation systems. To this end, three components are defined by the technical parts of the standards:

- The HLA Rules describes the general principles defining how the federation and the participating federates work together, i.e., how responsibilities for updates are shared, who does what when, etc.
- The Interface Specification between the connection middleware which is called Runtime Infrastructure (RTI) and a federate, which provides an application interface in both direction: what services provided by the RTI the simulation system can call, and what services the RTI will call in order to request something from the simulation system.
- The Object Model Template (OMT) that defines the structure of the information exchange between the federates via the RTI.

In order to make sure that (1) all information required is provided to the right federate, (2) only the information required is provided to the right federate, and (3) the information is provided at the correct time, six management areas are provided for effectiveness, efficiency, and timeliness: federation, declaration, object, data distribution, time, and ownership management.

The Object Model Template (OMT) defines what information can be exchanged. In principle, there are two categories of information that can be exchanged, which are persistent objects and transient interactions. The main difference is that interactions are distributed just once while objects are created, they can be updated, they can change ownership, and they can be destroyed. All interactions and objects including parameters and attributes and other definitions build the Federation Object Model (FOM). The information exchange within the federation is done in orchestration of RTI services with the OMT definitions. The information provided in the OMT defines what information can be exchanged between the participating federates, the services provided by the RTI defines define how the information can be exchanged.

6 CONCLUDING REMARKS

This tutorial could hardly scratch on the surface of all topics. The simulation engineer supporting this domain has to an expert in many domains and support bridging many gaps between important experts. He needs to understand the fundamentals of combat and the related missions and tasks, he has to know the basics about the weapon systems and the tactics and procedures, and he has to understand how to model all aspects accordingly. Once modeled, the simulation system must be implemented allowing to be used in distributed operations and exercises. Therefore, he needs to understand the computational and conceptual challenges of distributed computing, applied to the defense domain. As such, combat modeling and distributed simulation remain one of the most challenging application domains within the M&S discipline and provide many valuable lessons learned for other domains interested to apply M&S in their field on a comparable scale, such as health care and medical simulation are currently aiming at.

REFERENCES

Alberts, D. S. et al. 2002. *NATO Code of Best Practice for Command and Control Assessment*. Washington, DC: CCRP Press.

Cayirci, E., and D. Marincic. 2009. *Computer Assisted Exercises and Training: A Reference Guide*. Hoboken, NJ: John Wiley & Sons.

Deitz, P. H., and E. W. Edwards. 2009. *Fundamentals of Ground Combat System Ballistic Vulnerability/Lethality*. American Institute of Aeronautics and Astronautics.

IEEE 2010. *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – IEEE Std 1516-2010, 1516.1-2010, 1516.2-2010*. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

IEEE 2012. *IEEE Standard for Distributed Interactive Simulation – IEEE 1278.1-2012*. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Labbé, P. et al. 2006. *Guide for Understanding and Implementing Defense Experimentation GUIDEx*. Norfolk, VA: The Technical Cooperation Program, NATO ACT.

Ören, T. 2011. "The many Facets of Simulation through a Collection of about 100 Definitions". *SCS M&S Magazine* 2 (2): 82–92.

Page, E. H., and R. Smith. 1998. "Introduction to Military Training Simulation: a Guide for Discrete Event Simulationists". In *Proceedings of the 1998 Winter Simulation Conference*, edited by J. C. D.J. Medeiros, E.F. Watson and M. Manivannan, 53–60. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Powell, E. T., and J. R. Noseworthy. 2012. "The Test and Training Enabling Architecture (TENA)". In *Engineering Principles of Combat Modeling and Distributed Simulation*, edited by A. Tolk, Chapter 20, 449–477. Hoboken, NJ: John Wiley & Sons.

Sargent, R. G. 2013. "Verification and Validation of Simulation Models". *Journal of simulation* 7 (1): 12–24.

Strickland, J. 2011. *Mathematical Modeling of Warfare and Combat Phenomenon*. Raleigh, NC: Lulu Enterprises Inc.

Tolk, A. 2012. *Engineering Principles of Combat Modeling and Distributed Simulation*. Hoboken, NJ: John Wiley & Sons, Inc.

U.S. D.O.D. Joint Staff 2010. *Department of Defense Dictionary of Military and Associated Terms*. Washington, DC: Department of Defense.

AUTHOR BIOGRAPHY

ANDREAS TOLK is Computer Science Principal in the Simulation Engineering Department of The MITRE Corp., Hampton VA USA. He is a Fellow of the Society for Modeling and Simulation (SCS). His email is atolk@mitre.org.