Toward a Family of Maturity Models for the Simulation Interconnection Problem

Ernest H. Page
Abstraction and Associates
11505 Purple Beech Drive
Reston, VA 20191
703-282-3473
ernie@thesimguy.com

Richard Briggs
John A. Tufarolo
Virtual Technology Corporation
5510 Cherokee Avenue
Suite 350
Alexandria, VA 22312
703-658-7050
rbriggs@virtc.com
jtufarolo@virtc.com

Keywords:
Composability, Interoperability, Capability Maturity Models, Formal Specification

ABSTRACT: *Starting with a restricted view of composability, we propose a framework for the broader simulation interconnection problem and suggest roles for composability, interoperability and integratability within that framework. We briefly highlight how the framework could be used to: (1) further the ongoing efforts in the theory of simulation composability, and (2) serve as a basis for a family of Maturity Models for simulation.*

1. Introduction

In layman's terms, the general problem we seek to address is easy to describe. Within the U.S. Department of Defense (DoD), in particular, a great deal of effort over the past two decades has been devoted to enabling independently developed simulations to work together at runtime. For historical context refer to [1,2,3,4]. This activity has been engaged largely under the rubric of *interoperability*. Despite the belief that reuse through interoperability is critical to improving quality and reducing the costs of systems analysis, acquisition and training, the construction and use of *federations* of interoperating simulations is fraught with complexity. For large-scale federations, in particular, the process is costly and error-prone. And so the problem DoD confronts is this:

> *Given two simulation systems, A and B, what is a reasonable expectation regarding their ability to execute cooperatively in support of a given set of modeling objectives? What is a reasonable expectation regarding the costs and benefits of making A and B work together?*

Whether the "cooperative execution" of *A* and *B* is referred to as *interoperation* or something else—interconnection, integration, federation, confederation, composition—is largely unimportant. The important question is whether or not analytically useful results will be produced by *A* and *B*, or whether or not useful training will be provided by these two systems. Many, many factors, existing at many "levels" of the problem, dictate the answer to this question. For example, do the algorithms within *A* and *B* adopt a common set of assumptions? Do the software implementations of these algorithms permit the necessary data exchanges, e.g. does one system represent a value as an integer while the other represents the value as a Boolean? When the systems are installed for execution, have the proper configuration files been installed with them? Are the host tables configured so that the machines involved can address one another? Are the network cards functioning correctly? Are the multicast groups set up properly? Have the user interfaces been configured to trap errant user input at runtime? Is network access restricted to prevent someone from running an ftp job during execution and degrading network performance? Are back-up generators available in case of power failures?

Each of these questions—and many, many others—must be answered in order to determine whether or not two or more systems can be made to produce something useful on a given date and time, at a given (set of) location(s).

In this article, we suggest a partitioning for these concerns within what we refer to as the *general simulation interconnection problem*. In particular, we are interested in achieving two objectives: (1) ascribing an unambiguous and narrowly defined role for the emerging concept of composability, and (2) establishing a framework suitable for the development of a family of Maturity Models for the simulation interconnection problem.

The remainder of the article is organized as follows. Sections 2 and 3 review the current state of Capability Maturity Models and simulation composability, respectively. A framework for the simulation interconnection problem is suggested in Section 4 and assessed in Section 5. Conclusions are given in Section 6.

2. Capability Maturity Models

Capability Maturity Models (CMM) for software and systems engineering processes date to [5] and have seen a resurgence in recent years. A typical web search on “Capability Maturity Model” produces on the order of 200,000 hits[1], indicative of their prominence in the industry.

Capability Maturity Models (CMMs) are fundamentally mechanisms used to provide guidance to organizations defining processes. They describe *what* activities must be performed in order to meet certain criteria, while not prescribing *how* these activities should be accomplished [6]. *Capability* is typically portrayed as a series of finite, increasing levels. *Maturity* implies these capabilities must be “grown” over time. *Model* simply acknowledges that this is an abstract representation of something; representing its known/inferred properties, and can be used to further study its characteristics [7].

2.1 Prevalent Capability Maturity Models

CMMs are defined for several domains, including: software development, software acquisition, systems engineering, systems security engineering, and program management. These CMMs all share a common *process-centric* theme. That is, they focus primarily on organizational processes, and do not directly consider the end results. By contrast, a results-centric (or product-centric) approach focuses on whether desired results are achieved, regardless of the processes used to achieve those results.

The Carnegie Mellon Software Engineering Institute (SEI) Software Capability Maturity Model (SW-CMM) is often synonymously identified with all CMM work. Reinforcing that notion is the fact that the terms “Capability Maturity Model” and “CMM” are both registered by the SEI with the U.S. Patent and Trademark Office [8]. SEI currently recognizes six separate SEI CMM products [9,10,11]. Current SEI efforts focus on the development of the CMMI, which encompasses all extant CMMs.

2.2 Interoperability Assessments

Many DoD efforts have been established and evolved to address interoperability, including (summarized in [12]):

- *Joint Technical Architecture* - Defines standards governing implementation of system capabilities and interfaces.
- *Defense Information Infrastructure (DII) Common Operating Environment (COE)* - Establishes a commonly executable environment for systems.
- *Network Centric Enterprise Services (NCES)* - Follow-on to COE, that defines core and interest group services for the Global Information Grid (GIG).
- *Shared Data Environment (SHADE)* - Intended to reach agreement on comment data models for systems.
- *Joint Interoperability Test Command (JITC)* - Tests and certifies systems based on standards conformance and demonstrated application to application interoperability

[1] Result of an informal search on http://www.google.com; 26 January 2004.

- *Joint Battle Center* - Forum for conducting experiments regarding information systems interoperability, integration, technology insertion, and system performance (in a Joint environment)
- *Levels of System Interoperability (LISI)* - an approach to help define a formal construct for defining varying levels of information exchange

The LISI approach is based on a maturity model construct. We briefly highlight LISI below.

2.3 Levels of Information Systems Interoperability (LISI)

The Levels of Information Systems Interoperability (LISI) effort is an approach to help define a formal construct for defining varying levels of information exchange. This construct is in the form of a *LISI maturity model*.

The *LISI maturity model* defines a common DoD basis for requirements definition and for incremental system improvements. It defines stages for systems to mature, in order to improve their chances for interoperating with other systems.

The LISI maturity model addresses levels of interoperability between software operational (platform) environments. LISI defines five levels of interoperability:

1. *Isolated*: The systems are completely isolated, incapable of on line interoperability. Examples: manual gateway: diskettes, tape, hard copy exchange.
2. *Connected*: The systems are sufficiently connected to share files via email, ftp, or file sharing, but without assurance that an application capable of opening them will be available across platforms. Examples: FM voice, tactical data links, text file transfers, messaging systems, email.
3. *Functional*: Minimum common function with separate data and applications: heterogeneous product exchange for group collaboration; exchange of annotate imagery, maps with overlays.
4. *Domain*: Shared databases, sophisticated collaboration. Example: common operational picture
5. *Enterprise*: Distributed global information and applications; e.g. Interactive COP update, event-triggered global database updates. Examples: interactive applications, shared data and applications.

Unlike CMM-style maturity models, the levels within the LISI maturity model do not *necessarily* connote successive improvement. In a typical CMM, an organization at Level 4 is “better” than an organization at Level 2 in the sense that the Level 4 organization’s processes are more repeatable. Therefore, statistically speaking, one can place a higher expected value on the probability of success for the Level 4 organization to deliver on a contract than one can place on the Level 2 organization. In LISI, higher levels do not connote more interoperability. Two systems at Level 2 may be completely interoperable if, for example, ftp is all that is needed. However, like the traditional CMMs, it may be possible to ascribe increasing probabilities for successful interoperability as the LISI levels increase.

Tolk applies a LISI-style construct to the development of an interoperability framework for conceptual models, known as the Levels of Conceptual Interoperability Model (LCIM) [43]. In the following sections, we suggest a framework for the general simulation interconnection problem. The framework (currently) differentiates composability, interoperability and integratability. We anticipate that LISI/LCIM-style constructs are useful (needed) for each of these dimensions.

3. Composability

Composition, as a term applied to the creation of software artifacts, is rooted in the component-based software engineering (CBSE) paradigm [13]. Although it is arguable that the barriers to software reuse are anything other than cultural in nature, one of the primary goals of CBSE is to foster software reuse through the identification and development of software *components* as “building blocks” for complex software systems. Key aspects of CBSE include: (1) a component is deployed in executable form (e.g. binary), (2) a component is stateless, and (3) a component is named but not unique; instances of a component are indistinguishable. Significant investment in component-oriented software architectures has been undertaken in the commercial marketplace over the past decade resulting in technologies such as CORBA, Microsoft COM, and Enterprise JavaBeans.

Within the military simulation domain, *composability* has arisen as a cousin of the longstanding Department of Defense (DoD) objective of interoperability. Like many terms from the DoD lexicon, the notion of composability is vaguely and disparately applied. In most contexts, the term composition is used to describe the rather generic act of "creating a system from components." Since interoperability also involves systems and components, this broad use of the term composability is difficult to differentiate from interoperability. Further, the nature of "components" is rarely specified, so composability can be used to refer to any act of creating a system by combining "stuff"—where "stuff" is undefined. This notion of composability arguably subsumes all acts of creation.

3.1 A brief history

The earliest uses of the term composability within the military simulation context date to the Composable Behavioral Technologies (CBT) project during the mid 1990s (see [14]). The purpose of CBT was to give ModSAF users a convenient way to develop new entity behaviors without appealing to the underlying SAF source code. Shortly after the initiation of CBT, composability appeared as a system objective within the JSIMS Mission Needs Statement. A taxonomy and use case for composability in the context of JSIMS appears in [15]. The impact of composability as a system objective on the JSIMS design is described in [16]. Composability is also identified as a key system objective for OneSAF [17], and the OneSAF Product Line Architecture Framework (PLAF) has been designed to accommodate high degrees of user tailorability [18].

In 1998, the DARPA Advanced Simulation Technology Thrust (ASTT)—which was chartered to develop technology in support of JSIMS—funded two separate studies on simulation composability: (1) the Model Based Simulation Composition (MBSC) project, which developed a prototype composition environment for JSIMS [19,20,21,22]; and (2) a study by Page and Opper that investigated the composability problem from a computability and complexity theoretic perspective [23].

A focus-paper session at the 2000 Winter Simulation Conference addressed methodologies for composable simulation [24,25]. Recently, the work of Petty, Weisel and Mielke [26,27,28,29] provides a broad survey of the uses of the term composability, extends the work of Page and Opper, and examines the composite validation problem within the context of automata theory.

The Defense Modeling and Simulation Office (DMSO) initiated a collection of studies as part of the Composable Mission Space Environments (CMSE) initiative in FY03 [30]. The comprehensive report by Davis and Anderson provides a broad survey of the topic of composability and suggest a wide-ranging investment strategy for the DoD in this area [31].

The topic of "web-based simulation" emerged in the mid-1990s and introduced the concepts of composing simulations via web protocols (see [32,33,34]). The Extensible Modeling and Simulation Framework (XMSF) is an effort by the Naval Postgraduate School, George Mason University, SAIC and Old Dominion University to develop a ubiquitous web-based simulation environment [35].

3.2 Current definitions

A recent survey of the topic of composability by Petty and Weisel observes that uses of the term composability vary widely [26]. The authors suggest the following as a "common" definition:

> *Composability – the capability to select and assemble simulation components in various combinations into valid simulation systems to satisfy specific user requirements.*

The authors elaborate their definition as follows:

> The defining characteristic of composability is that different simulation systems can be composed at configuration time in a variety of ways, each suited to some distinct purpose, and the different possible compositions will be usefully valid simulation systems. Composability is more that just the ability to put simulations together from parts; it is the ability to combine and recombine, to configure and reconfigure, sets of parts from those available into different simulation systems to meet different needs.

In their survey, Davis and Anderson adopt the Petty and Weisel definition with some additional specification regarding the nature of components [31]. Davis and Anderson eschew the use of the term *valid* in favor of the term *meaningful*.

3.3 Theory and analysis

Motivated by the implication of high degrees of automated support for composability expressed within the JSIMS and OneSAF program requirements, Page and Opper consider composition from a computability and computational complexity theoretic perspective [23]. The authors observe that prior work in analyzing simulation model specifications suggests that many of the problems attendant with simulation model development, verification and validation are *fundamentally hard*, and that automation can only provide so much relief [23; p. 554]. For example, problems such as the following cannot be solved in the general case: (1) determining if a model is finite (i.e. will run to completion); (2) determining if a model specification is complete; and (3) determining if a model specification is minimal. Other analyses has shown that problems such as the following have no *efficient* solution: (1) determining whether any given state will occur during an execution of a model; (2) determining the existence of, or possibility for, simultaneous events; and (3) determining whether a model implementation satisfies a model specification.

Page and Opper observe informally that the class of *model specifications* must include the class of *model compositions*, and therefore any result in the computational complexity of model specifications applies directly to model compositions. In support of this observation, they provide a formal analysis of a simple, generic methodology for composable simulation. They observe that building simulation models by composition implies not only *identifying* (via search) relevant candidates from (possibly massive) component repositories, but also answering the following: (1) does a combination of components exist that satisfies the modeling objectives, and (2) if so, can the "best" (or a "good enough") solution be identified in a reasonable time. If not, how closely can the objectives be met?

Determining whether a collection of components satisfies a modeling objective might be accomplished in any number of ways, including:

- Determination made strictly on the basis of the descriptions of component capabilities (i.e. *metadata*).
- Determination made by modeling or approximating component interactions.
- Determination made by constructing the composed model and observing the result(s).

Page and Opper observe that a determination made on the basis of metadata is the least computationally intensive solution—assuming such a determination were possible. They suggest a simple formal model of composition based on set theory and describe a generic decision problem for composability as follows:

COMPOSABILITY

INSTANCE:	A set *O* of objectives and a collection *C* of components.
QUESTION:	Is there a valid composition that meets the objectives stated in *O*?

The authors conjecture that the decision problem COMPOSABILITY is NP-complete. In the development of a proof of this conjecture, the authors observe, however, that certain objectives may be undecidable on their face, e.g. the simulation terminates for a given set of circumstances. To accommodate this, Page and Opper suggest two variants of the COMPOSABILITY decision problem: (1) BOUNDED COMPOSABILITY – each objective in *O* is decidable; and (2) UNBOUNDED COMPOSABILITY – some objective in *O* is undecidable. Further, the authors observe that it may be possible for two components, *A* and *B*, to satisfy some objective *O* and that their ability to satisfy *O* could not be predicted based on *any* metadata for *A* and *B*. Page and Opper suggest that this characteristic is like the property of *emergence* in complex adaptive systems and suggests two more variants of the COMPOSABILITY decision problem: (1) EMERGENT COMPOSABILITY – composition cannot be evaluated based on metadata; and (2) NONEMERGENT COMPOSABILITY – the composition can be evaluated based on metadata.

The cross-product of the variants yields four decision problems:

- UNBOUNDED EMERGENT COMPOSABILITY
- BOUNDED EMERGENT COMPOSABILITY
- UNBOUNDED NONEMERGENT COMPOSABILITY
- BOUNDED NONEMERGENT COMPOSABILITY

From a complexity perspective, BOUNDED NONEMERGENT COMPOSABILITY (BNC) is the simplest. Page and Opper provide a proof that BNC is NP-complete. This proof suggests that use cases for composability that imply automated support for determining valid combinations of components (e.g. Steps 2 and 5 from the JSIMS Composability Use Case [15]) cannot have an efficient solution in the general case.

Petty, Weisel and Mielke [27] extend the work of Page and Opper and suggest another variant of the problem, ANTI-EMERGENT COMPOSABILITY (AC). In AC, two models *A* and *B* may each satisfy some objective *O*, but their combination does not. Petty, Weisel and Mielke suggest a general form of the component selection problem that subsumes the variants and prove that it is NP-complete.

4. The General Simulation Interconnection Framework

With the intention of defining a family of Maturity Models suited to assist the government in answering the questions posed in Section 1, we suggest a framework for the general simulation interconnection problem as illustrated in Figure 4.1.

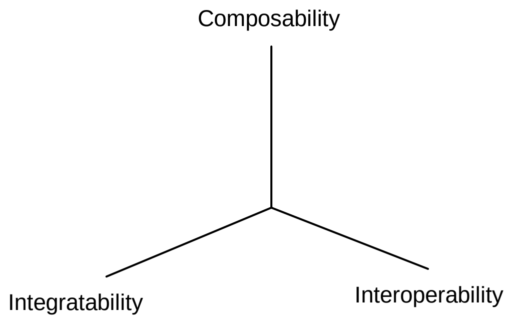

Fig. 4.1. Aspects of the General Simulation Interconnection Problem

We suggest (at least) three dimensions for the problem:

- *Composability* – realm of the model (e.g. two models are composable if their objectives and assumptions are properly aligned).
- *Interoperability* – realm of the software implementation of the model (e.g. are the data types consistent, have the little endian/big endian issues been addressed, etc.)
- *Integratability* – realm of the site the simulation is running at (e.g. have the host tables been set up; are the NIC cards working properly).

To successfully achieve the cooperative execution of two or more models, each of these dimensions of the interconnection problem must be "solved". In establishing the framework, and the associated family of Maturity Models, we would like to characterize these dimensions such that they are independent. That is, two models can be made to be interoperable, without their being composable. Similarly, two composable models may not necessarily be interoperable.

We note that this use of the term interoperability reflects what is typically referred to as *syntactic* or *technical* interoperability. While this use of the term composability seems closely equivalent to *semantic* or *substantive* interoperability [26,36].

4.1 Toward a Formalized Restricted View of Composability

As noted in Section 3, the term composability is used within the military simulation domain to imply a variety of notions, ranging from interoperability, to end-user tailorability, to any act of creation. We believe that for the term to be useful, it must be unambiguously differentiated from these other concepts.

The definition suggested by Petty and Weisel loosely differentiates the notions of composability and interoperability as follows [26]:

> Essentially, interoperability is the ability to exchange data or services at run-time, whereas composability is the ability to assemble components prior to run-time … It can be seen that interoperability is necessary but not sufficient to provide composability. Composability (engineering and modeling) does require interoperability (technical and substantive). Federates that are not interoperable can not be composed, so interoperability is necessary for composability. However, interoperability is not sufficient to provide composability, i.e. federates may be interoperable but not composable. Recall that an essential aspect of composability is the ability not just to combine federates but to combine and recombine federates into different simulation systems. Federates that are interoperable in one specific configuration or with one specific object model, and cannot be combined and recombined in other ways, are

not composable … The matter of substantial effort is crucial to the distinction between interoperability and composability.

These distinctions seem somewhat problematic, because it isn't immediately clear how a run-time characteristic (interoperability) could be a necessary condition to enable a pre-run-time characteristic (composability). Further, the term "substantial effort" requires quantification.

Petty and Weisel distinguish composability and integratability as follows [26]:

> Integration is the process of configuring and modifying a set of components to make them interoperable and possibly composable. Essentially any federate can be integrated into any federation with enough effort, but composability implies that the changes can be made with little effort.

This distinction also seems somewhat problematic since the level of "effort" is also used to distinguish between composability and interoperability

As an alternative, we suggest that composability be viewed as a property of a set of models. Specifically it should be expressed as some function on the congruity of the objectives and assumptions underlying each model in the set. In this sense we agree with the definition of Petty and Weisel. That is, composability is a property that may be assessed prior to run time. However, in our view composability is independent of interoperability. Interoperability is a property of the software implementation of a set of models (or other systems). The objectives and assumptions underlying two models, *A*, and *B*, may be wholly congruent and thus the models composable, but their software implementations may utilize different programming languages, data types, marshalling protocols and so forth such that the two implementations are not interoperable.

Informally, two models are *composable* if they share compatible objectives and assumptions. Quantifying and reasoning about the "compatibility" of objectives and assumptions should be the domain of research in algebras and calculi for composition.

4.2 Incorporating composability within existing formalisms

This definition for composability could be easily formalized within, for example, the Discrete Event System Specification (DEVS) (see [37]). The "classic" DEVS defines a basic discrete event system specification as:

$$DEVS = \{X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta\}$$

Where

X is the set of inputs
Y is the set of outputs
S is the set of *sequential* states
$\delta_{ext}: Q \times X \rightarrow S$ is the *external state transition function*
$\delta_{int},: S \rightarrow S$ is the *internal state transition function*
$\lambda,: S \rightarrow Y$ is the output function
ta: $\mathrm{S} \rightarrow \Re^{+} \cup \infty$ is the *time advance function*
$Q = \{(s,e\} \mid s \in S, 0 = e = ta(s)\}$ is the set of *total states*.

To accommodate reasoning on the composability of models, we might augment the 7-tuple to include:

O : a set of objectives,
A : a set of modeling assumption

The DEVS formalism includes support for the specification of multicomponent systems under the rubric of *coupled* models. The specification of coupled models augments the DEVS structure with sets of input and output ports. Couplings are assessed with respect to the congruity of the data flowing along those ports and the time advance functions of the individual models.

Similarly, this definition could be employed within the formalism developed by Petty and Wiesel based on computable functions [29].

4.3 Composability and simulation time

The representation of time is, arguably, the singularly distinctive characteristic of simulations when compared to other types of software. Techniques for the cooperative execution of discrete event and continuous simulations date to the 1960s (see [38]).

Such applications are referred to as *combined*, *mixed* or *hybrid* simulations. In the continuous simulation domain, the area of multiparadigm modeling deals with a number of open problems with the combining of continuous models of varied formalisms, such as finite element models and computational fluid dynamics models (see [39]). Carson, Bruce and Baxter discuss the reconciliation of real-time and logical time within a High Level Architecture context [40].

The DoD M&S Glossary stipulates that time is an artifact of a simulation and not a model. If we adhere to that view, then reconciling disparate time flow mechanisms falls within the domain of interoperability in our framework. And so the treatment of time would not be part of the determination of the composability of two models. If we wanted to include the treatment of time within the composability domain, we could simply state that every model must have an assumption regarding the (preferred) implementation of time.

5. Assessing the Framework

Arguments about terminology can be useful, but only to a certain point. The fundamental capability the military simulation community seeks is the ability to make independently-developed systems execute in a cooperative fashion to generate useful analytical or training results. On this point, the community is more-or-less in agreement. The names that we give to the components of this activity may be varied, but these differences are likely more a matter of aesthetics than correctness. In the following paragraphs, we argue in favor of the utility of the framework we have suggested, but recognize there are many alternate, equally valid frameworks.

If we adopt the perspective of Petty and Weisel [26], where the defining characteristic of composability is related to the ease and repetitiveness of the act of composition—that is, a system is composable only if you can put it together easily and for more than one purpose—then we should focus our attention on standards for model specification and implementation. The way we describe and construct models should be in a representation that is familiar to all those who practice composition. Otherwise, language differences will certainly impede the realization of composable simulation. We could focus heavily on the modern software practices that enable enterprise architectures in many domains – code introspection and discovery; peer-to-peer networking, and so forth. However, Page and Opper [23] and Petty, Weisel and Mielke [27] illustrate that the problems attendant with model validation have no efficient automated solution and therefore, by this definition, composable simulation will be very difficult to achieve.

If we adopt the perspective of OneSAF where composable means user tailorable, we must grapple with the question of "what is programming"? If the widgets we provide the user and the rules for the composition of those widgets are sufficiently robust such that a user can express any computation in terms of compositions of widgets, then the user is, in fact, programming. The user is simply programming in some higher-order language. Clearly such a language would be tailored to a problem domain (e.g. military simulation), and perhaps a given level of abstraction within that problem domain (e.g. entity-level). Defining the family of abstractions for military modeling has been identified as a Grand Challenge problem [41].

If we adopt the perspective suggested here that composability is a function of two models and should be expressed as some function of the underlying objectives and assumptions of those two models, then we have created an explicit structure that acknowledges that not all models should be interconnected. We can define standard interfaces, and standards for data exchange. We can ensure that networks and multicast groups are properly configured. But if two models have sufficiently conflicting sets of assumptions, those models cannot, *and should not*, work together. In our framework, those models would be classified as not composable. We can define maturity levels for interoperability and integratability. We can, and perhaps should, completely eliminate every technical barrier to the interconnection of two models—in the same sense that Brooks argues for the elimination of the *accidental* complexities of software [42]. But we can do nothing about their fundamental composability—which we will stipulate is an *essential* complexity of simulation modeling. In our framework, it is simply an assessment that must be made; a solution to some function on objectives and assumptions sets that might yield a fuzzy answer, but in all probability yields a binary result—the models are, or are not, composable.

6. Conclusions

The DoD should pursue such a paradigm of composability and the underlying mathematics that enables it. As we march down the road of interoperability in the name of fiscal responsibility we should incorporate explicit tests for composability. We must recognize that there is no Universal abstraction. There is no way to define a model of a system—except perhaps a null model—that is composable with every other model of that system. Models are inherently fragile, their fragility an artifact of their relationship to a set of objectives and assumptions. Models are unlike most other types of software in that they cannot be asserted with a definiteness of purpose. For example, "sort a list of integers in nondecreasing order" is quite different than "calculate the line-of-sight between two objects". The former problem statement includes, at most, a small number of assumptions which can be quickly rooted out. Further, the ability of an algorithm to successfully accomplish its stated task can be determined by a simple examination of the input and output. This is not the case for the latter problem statement. This problem statement—which implies the construction of a model—carries with it an infinite number of assumptions. And the correctness of the model cannot be determined based solely on an examination of the model's input and output. The assumptions must be reconciled against the objectives in order to characterize the correctness of the model.

Forthcoming articles will present further developments in the theory of composition, and discuss the details of the Maturity Models for interoperability and integratability. The degree to which composability supports the concept of a Maturity Model remains an open question.

References

[1] Alluisi, E.A. "The Development of Technology for Collective Training: SIMNET, a Case History," The Human Factors Society, Reprinted in *A Revolution in Simulation: Distributed Interaction in the '90s and Beyond,* L.D. Voss.

[2] Page, E.H., Canova, B.S. and Tufarolo, J.A. "A Case Study of Verification, Validation and Accreditation for Advanced Distributed Simulation", *ACM Transactions on Modeling and Computer Simulation*, **7**(3), pp. 393-424, July 1997.

[3] Kuhl, F., Weatherly, R and Dahmann, J. *Creating Computer Simulation Systems – An Introduction to the High Level Architecture*, Prentice Hall, Upper Saddle River, NJ, 1999.

[4] Fujimoto, R.M. *Parallel and Distributed Simulation Systems*, John Wiley and Sons, New York, NY, 2000

[5] Humphrey, W.S. *Managing the Software Process*, SEI Series in Software Engineering, Addison-Wesley Publishing Company, 1989.

[6] Systems Security Engineering Capability Maturity Model (SSE-CMM) Project. Systems Security Engineering Capability Maturity Model (SSE-CMM) Model Description Document Version 3.0. 15 June, 2003

[7] The American Heritage® Dictionary of the English Language, Fourth Edition, 2000.

[8] U.S. Patent and Trademark Office, 6 October 1998.

[9] Capability Maturity Models Webpage. http://www.sei.cmu.edu/cmm/cmms/cmms.html

[10] Curtis, Bill, Hefley, William E., and Miller, Sandy A. People Capability Maturity Model (P-CMM) Version 2.0, Carnegie Mellon Software Engineering Institute, July 2001, CMU/SEI-2000-MM-01.

[11] Paulk, Mark C., How ISO 9001 compares with the CMM. http://www.sei.cmu.edu/cmm/papers/9001-cmm.pdf. IEEE Software, January 1995, pp. 74-83.

[12] C4ISR Architecture Working Group, Levels of Information Systems Interoperability (LISI) - Final Document. March 1998.

[13] Szyperski, Clemens. Component Software – Beyond Object-Oriented Programming – Second Edition. Addison-Wesley and ACM Press, 2002, ISBN 0-201-74572-0.

[14] Courtemanche, A.J., von der Lippe, S.R. and McCormack, J. "Developing User-Composable Behaviors," In: *Proceedings of the Fall 1997 Simulation Interoperability Workshop*, 97F-SIW-068, 8-12 September 1997, Orlando, FL.

[15] JSIMS Composability Task Force, Final Report, 30 September 1997.

[16] Pratt, D., Ragusa, C. and von der Lippe, S. "Composability as an Architecture Driver", In: *Proceedings of the 1999 I/ITSEC Conference,* Orlando, FL, 29 November – 2 December 1999.

[17] U.S. Army Simulation, Training and Instrumentation Command. OneSAF Operational Requirements Document. Version 1.0, Orlando, FL, June 2000.

[18] Courtemanche, A.J. and Whittman, R.L. "OneSAF: A Product Line Approach for a Next Generation CGF", In: *Proceedings of the 11th Conference on Computer-Generated Forces and Behavior Representation*, pp. 349-361, Orlando, FL, 7-9 May 2002.

[19] Aronson, J. and Wade, D.M. "Benefits and Pitfalls of Composable Simulation", In: *Proceedings of the Spring 2000 Simulation Interoperability Workshop*, 00S-SIW-155, 26-31 March 2000, Orlando, FL.

[20] Aronson, J. and Wade, D. "Model Based Simulation Composition", In: *Proceedings of the Fall 1998 Simulation Interoperability Workshop*, 98F-SIW-055, 14-18 September 1998, Orlando, FL.

[21] Davis, D. and Aronson, J. "Component Selection Techniques to Support Composable Simulation", In: *Proceedings of the Spring 1999 Simulation Interoperability Workshop*, 99S-SIW-043, 14-19 March 1999, Orlando, FL.

[22] Wade, D.M. and Aronson, J. "Model Based Simulation Composition: A Unified Scenario Model to Support Composable Simulation", In: *Proceedings of the Spring 1999 Simulation Interoperability Workshop*, 99S-SIW-044, 14-19 March 1999, Orlando, FL.

[23] Page, E.H. and Opper, J.M. "Observations on the Complexity of Composable Simulation", In: *Proceedings of the 1999 Winter Simulation Conference,* pp. 553-560, Phoenix, AZ, 5-8 December 1999.

[24] Kasputis, S. and Ng, H.C. "Composable Simulations", In: : *Proceedings of the 2000 Winter Simulation Conference,* pp. 1577-1584, Orlando, FL, 10-13 December 2000.

[25] Davis, P.C., Fishwick, P.A., Overstreet, C.M. and Pegden, C.D. "Model Composability as a Research Investment: responses to the Featured Paper", In: : *Proceedings of the 2000 Winter Simulation Conference,* pp. 1585-1591, Orlando, FL, 10-13 December 2000.

[26] Petty, M.D. and Weisel, E.W. "A Composability Lexicon", In: *Proceedings of the Spring 2003 Simulation Interoperability Workshop*, 03S-SIW-023, 30 March – 4 April 2003, Kissimmee, FL.

[27] Petty, M.D., Weisel, E.W. and Mielke, R.R. "Computational Complexity of Selecting Components for Composition", In: *Proceedings of the Fall 2003 Simulation Interoperability Workshop*, 03F-SIW-072, 14-19 September 2003, Orlando, FL.

[28] Weisel, E.W., Petty, M.D. and Mielke, R.R. "Validity of Models and Classes of Models in Semantic Composability", In: *Proceedings of the Fall 2003 Simulation Interoperability Workshop*, 03F-SIW-073, 14-19 September 2003, Orlando, FL.

[29] Petty, M.D. and Weisel, E.W. "A Formal Basis for a Theory of Semantic Composability", In: *Proceedings of the Spring 2003 Simulation Interoperability Workshop*, 03S-SIW-054, 30 March – 4 April 2003, Kissimmee, FL.

[30] Defense Modeling and Simulation Office (DMSO). Composable Mission Space Environments homepage. https://www.dmso.mil/public/warfighter/cmse/

[31] Davis, P.K. and Anderson, R.H. *Improving the Composability of DoD Models and Simulations*, RAND, National Defense Research Institute, Santa Monica, CA, 2003.

[32] Page, E.H., Buss, A., Fishwick, P.A., Healy, K.J., Nance, R.E. and Paul, R.J. "Web-Based Simulation: Revolution or Evolution?" *ACM Transactions on Modeling and Computer Simulation,* **10**(1), pp. 3-17, January 2000.

[33] Fishwick, P., Hill, D. and Smith, R. (Eds.) *Proceedings of the 1998 SCS International Conference on Web-Based Modeling and Simulation*, 11-14 January, San Diego, CA,.

[34] Bruzzone, A.G., Uhrmacher, A. and Page, E.H. (Eds.) *Proceedings of the 1999 International Conference on Web-Based Modeling and Simulation*, SCS Simulation Series **31**(3), 17-20 January, San Francisco, CA.

[35] Extensible Modeling and Simulation Framework Homepage. http://www.movesinstitute.org/xmsf/xmsf.html

[36] Dahmann, J.S. "High Level Architecture Interoperability Challenges", Presentation to the NATO Modeling and Simulation Conference, Norfolk, VA, 25-29 October 1999.

[37] Zeigler, B.P, Praehofer, H. and Kim, T.G. *Theory of Modeling and Simulation*, Second Edition, Academic Press, New York, NY, 2000.

[38] Nance, R.E. "The Time and State Relationships in Simulation Modeling", *Communications of the ACM*, 24(4): 173-179, April 1981.

[39] Mosterman, P.J. and Vangheluwe, H. (Eds.) *ACM Transactions on Modeling and Computer Simulation,* **12**(4), Special Issue on Computer Automated Multi-Paradigm Modeling, October 2002.

[40] Carson, J., Bruce, D. and Baxter, J. "Towards Time Management Interoperability", In: *Proceedings of the Fall 2003 Simulation Interoperability Workshop*, 03F-SIW-046, 14-19 September 2003, Orlando, FL.

[41] Fujimoto, R.M., Lunceford, W.H., Page, E.H. and Uhrmacher, A.M., (Eds). *Proceedings of the Dagstuhl Seminar on Grand Challenges for Modeling and Simulation*, Seminar 02351, Dagstuhl, Germany, 26-30 August 2002. Available at: http://www.informatik.uni-rostock.de/~lin/GC/report/index.html
[42] Brooks, F.P., "No Silver Bullet: Essence and Accidents of Software Engineering." *IEEE Computer*, **20**(4):10-19, April 1987.
[43] Tolk, A. and Muguira, J.A. "The Levels of Conceptual Interoperability Model", In: *Proceedings of the Fall 2003 Simulation Interoperability Workshop*, 03F-SIW-007, 14-19 September 2003, Orlando, FL.

Author Biographies

ERNEST H. PAGE is President of Abstraction and Associates (2001-present) and has been active within the military modeling and simulation community for the past ten years. Since receiving the Ph.D. in Computer Science from Virginia Tech in 1994, he has served on the technical staff of The MITRE Corporation (1995-2001) and has held the positions of Technical Advisor to the U.S. Army Model and Simulation Office (2000-2003) and Lead Scientist for the RDECOM MATREX (2003). He has authored over 40 articles in simulation and currently serves on the editorial boards of the ACM *Transactions on Modeling and Computer Simulation*, SCS *Simulation*, and the *Journal of Defense Modeling and Simulation*.

RICHARD BRIGGS is Vice president of Technology at Virtual Technology Corporation and has been active in the distributed simulation community since 1994. He has built DIS and HLA based simulation systems and was involved in the HLA development activities since 1995.

JOHN A. TUFAROLO is a Systems Engineer at Virtual Technology Corporation. He is currently the Integration Lead for the RDECOM MATREX and has been involved in many of the large DoD simulation systems over the last decade.