

Observations on the Complexity of Composable Simulation

Ernest H. Page
Jeffrey M. Opper

The MITRE Corporation
1820 Dolley Madison Blvd.
McLean, VA 22102

Abstract

We consider the issue of composability as a design principle for simulation. While component-based modeling is believed to potentially reduce the complexities of the modeling task, we describe some of the complexities introduced through composability that might tend to offset the benefits of component-based modeling on a large scale.

1 Introduction

A recent topic of interest in the general software community involves the notion of *component-based* software development. Representing the latest attempt to describe effective mechanisms for reusability, component technologies such as CORBA, COM/DCOM/OLE, ActiveX and JavaBeans, to name but a few, have received significant attention. Within the simulation community, component-based modeling is becoming increasingly prevalent. Within the defense simulation community, for example, a significant investment has been made to engender system-level reuse through *interoperability*. Standards like the High Level Architecture (HLA) have emerged. Through the HLA, the runtime interoperation of simulations (and other types of systems) is supported. The HLA permits simulation models to be used in multiple contexts and, by extension, is fostering the development of a component economy in the military simulation marketplace [2, 5, 12].

Component-based economies are also being conceived and implemented within the context of singular simulation systems. Defense simulations such as the Modular Semi-Automated Forces simulation (ModSAF) provide users with a palette of object and behavior *primitives* which may be combined—in a restricted sense—to form complex objects and behaviors. In many ways the capabilities of these systems mirror those of modern special-purpose simulation environments, like CACI's COMNET, which allow models to be constructed from iconic components. If the domain is sufficiently well-defined, and the rules for composition narrow, a user seldom needs to generate code when developing a model.

Outside the defense community, a component-based modeling economy is part of the *web-based simulation* vision. A web populated with digital objects—typically, models of physical counterparts—is described where modeling objectives are provided to search engines that, in turn, identify the appropriate digital object(s)

from which to construct an experimental model [8]. Common interfaces and protocols, like those defined in the HLA, permit the objects to interoperate at runtime [25]. The physical locations of the objects involved in the computation are not relevant to the modeler. In this envisioned future, simulation becomes ubiquitous. Model conceptualization, construction, execution and analysis is distributed, collaborative, and interactive. Levels of automated support for the modeling process significantly increase, and the pace of modeling is rapid [27, 28].

The software community has struggled with the concept of reuse for many years. Components offer a useful mechanism to support reuse. But a number of questions are raised by them as well. How do you describe components? How do you search for them? How do you put them together, and how do you know what you've got when you're done? What kinds of restrictions are required on the composition process? Intuitively, unconstrained composition is not achievable. In much the same manner that a programming language needs a syntax and semantics, so must a compositional architecture. But how much freedom is possible? What price must be paid for compositional freedom? Architecture Description Languages (ADLs) represent formal methods for describing—and, to a lesser extent, reasoning about—component-based systems [1, 14, 15]. Component calculi are also appearing [10, 17, 21, 34] but the answers to many of these questions remain unclear.

This paper reports on part of a study funded by the Defense Advanced Research Projects Agency (DARPA) to investigate composability as a design principle for simulation. The remainder of the paper is organized as follows. Section 2 briefly reviews some related analyses in computational complexity for simulation. We describe some of the complications introduced by composability as a design principle for simulation in Section 3. In Section 4 we present a simple formal model for composable simulation and use that model to characterize the state space of the composability problem. We also generate an NP-completeness proof regarding the complexity of identifying suitable compositions from component repositories. Some conclusions and a discussion of future research are given in Section 5.

2 Simulation and Complexity

Within the simulation community, few formal treatments of computational complexity exist. The act of model construction is regarded to be a fundamentally hard problem. Efficient structures and algorithms for model implementations—event list management, pseudorandom number generation, process transformation, and so forth—have been around for years. Some significant results in this area have emerged, however, from work in simulation model specification.

The earliest work in this area is due to Overstreet [22]. Overstreet defines a formalism for simulation model specification in which the description of model behavior has several useful and desirable properties:

1. The formalism is independent of traditional simulation world views.
2. A specification can be analyzed to identify natural components that measure complexity and identify

potential problems with the specification.

3. A specification can be transformed to produce additional representations that conform to traditional simulation world views.
4. Some aspects of a model can be left unspecified without hindering the analyses and transformations identified above.
5. A model is defined in terms that do not prescribe any particular implementation techniques, such as the time flow mechanism.

The goal of this formalism, the Condition Specification (CS), is to provide a world-view-independent model representation that is expressive enough to represent any model and concise enough to facilitate automated diagnosis of the model representation.

The CS has been used to explore a range of diagnostic analysis techniques for simulation model specifications [18, 19, 20, 23, 24, 26]. Most of the techniques are based on graph representations of the CS. At a more fundamental level, Overstreet demonstrates that any CS has an equivalent Turing Machine specification (and vice versa), thus showing that the following problems are undecidable:

- Whether any model specification in the CS is finite (i.e. an implementation will run to termination)
- Whether any two model actions are order independent
- Whether any model specification is complete
- Whether any model specification is minimal
- Whether any two model specifications are functionally equivalent

Deriving from work in Event Graphs, Yücesan and Jacobson [11, 31, 32, 33] generate similar conclusions to those of Overstreet. Focusing on tractability analysis rather than decidability analysis, Yücesan and Jacobson show the following problems are NP-hard:

- Whether or not any given state will occur during an execution of a model
- Determining the existence of, or ruling out the possibility for, simultaneous events
- Determining whether a model implementation satisfies a model specification
- Determining whether an execution can reach a state in which the termination condition has not been reached and no events remain on the event list

Similar results are also demonstrated for Petri net representations of simulation models [30]. Collectively, these analyses demonstrate 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.

3 Some Complexities of Composability

Intuitively, component-oriented design offers a reduction in the complexity of system construction by enabling the designer to reuse appropriate components without having to re-invent them. However, in the context

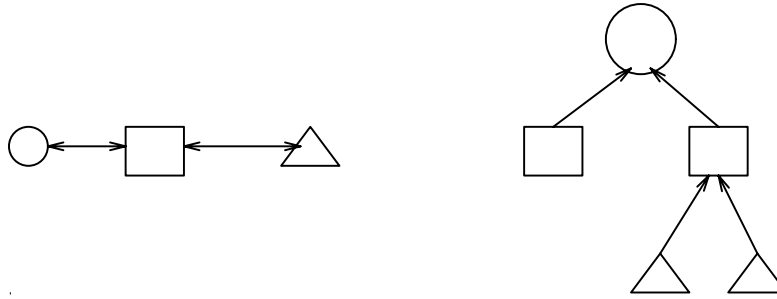


Figure 1: Horizontal and Vertical Dimensions of Simulation Composability.

of simulation, component-oriented design seems to introduce risks—or at least the possibility of risk—in a variety of ways. As illustrated notionally in Figure 3, the composable simulation problem seems to have both *horizontal* and *vertical* dimensions:

- In the horizontal dimension we consider the act of coupling components to facilitate their interoperation. This type of *peer-to-peer* integration, for example, is used within the Aggregate Level Simulation Protocol Joint Training Confederation (JTC) to provide a Theater-wide training simulation [16]. However, justifying a level of modeling abstraction with respect to a set of modeling objectives is a fundamentally challenging problem for *any* simulation model. The presence of multiple models and multiple levels of abstraction increases the difficulty. This problem has been referred to as the *multiresolution modeling* problem. Experience with the JTC and similar simulation systems indicates that multiresolution problems require significant effort to identify and ameliorate. Recent theoretical work in this area also indicates that multiresolution modeling is fundamentally hard to do correctly [6, 7, 29].

Thus, in the horizontal dimension, composability facilitates multiresolution modeling which is fundamentally hard.

- The vertical dimension of composability involves the act of coupling two components for the sake of aggregation. Commonly, aggregation/de-aggregation forms the basis for resolving resolution differences in interoperating defense simulations. However, it is easily seen that abstraction through aggregation may not provide the best (or even a valid) solution. For example, Kepler’s laws of planetary motion are not well-described in terms of quantum mechanics. One would have great difficulty aggregating atomic-level models into planetary-scale models and, having done so, such models would be poor representatives of the physical system.

Thus, in the vertical dimension, composability may encourage abstraction through aggregation/de-aggregation, which may in turn hinder the application of other, more suitable, methods for abstraction.

In addition to the risks attendant with the horizontal and vertical dimensions of composability, there seems to be an obvious scalability limit to the general reusability problem:¹ as the number of candidates for reuse (composition) becomes large, the benefits of reuse (composition) become negated by the costs of storage, organization and retrieval of candidates. The web seems to be a potential example of this phenomenon. The more information the web contains, the harder it becomes to find what you need, and the less useful the web becomes. If the web-based simulation vision is realized, and if it becomes profitable to publish models on

¹We view composition as a special case of the composability problem.

the web, the proliferation of models on the web in the future might mirror the proliferation of information on the web today.

But building simulation models by composition implies not only *identifying* (via search) relevant candidates from (possibly massive) component repositories but also answering the following:

- Does a combination components exist that satisfies the modeling objectives?
- 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 that a collection of components satisfies a requirement (or modeling objective) might be accomplished in any number of ways, including:

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

We assert that “ideally” we would like to make the determination of suitability using the first approach since it is computationally the simplest. However, even in this determination-by-metadata case we are confronted with a potentially computationally intensive problem. For *any* feature for which a variety of components are candidates, the complexity of analyzing the solution space could be $O(2^n)$ if all possible combinations are feasible.

In the following section, we introduce a simple formal model of the composable simulation problem and we begin to consider the range of computational complexity associated with composability.

4 A Formal Model for Composable Simulation

We begin with some notation. Let $O = \{o_1, o_2, \dots, o_n\}$ be a set of system objectives. Informally, we say that a system, S , *satisfies* or *meets* O if and only if each of the objectives in O are met. A variety of formalizations for satisfaction are possible here. We construct one such formalization in the subsequent text. Let $C = \{c_1, c_2, \dots, c_m\}$ be a set of components located in a repository, R . A system, S , consists of one or more of these components. That is, for purposes of this analysis, $S \subseteq C$.² If $|S| > 1$ then we say that S is a *composed* system. We say that the *composition* of two components, c_i and c_j is *valid* if the mechanisms necessary to make these two components interact are available. This may mean that c_i and c_j are present in binary compatible format, or that rules for their joint compilation are specified, or that the components can communicate via a common interface. We will allow the notion of “valid composition” to be context dependent. We now introduce our central decision problem.

²Alternately, we could view a system as a collection of components and connectors. For the current development, however, there is no need to represent connectors.

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 ?

CONJECTURE: COMPOSABILITY is NP-complete.

DISCUSSION: To demonstrate that a problem, Π , is NP-complete we must accomplish two things: (1) show that Π is in NP; (2) demonstrate that a known NP-complete problem may be polynomially transformed to Π [9].

Showing that a decision problem is in NP is typically done by noting that a nondeterministic algorithm may guess a solution and then verify it in polynomial time. In the case of COMPOSABILITY it is not clear that a polynomial time algorithm can be defined to determine if a set of objectives are met. The objectives may contain, for example, a requirement that the model halt for a given set of circumstances. This problem is, of course, undecidable. Therefore, given the current framing of our problem, we suspect that it is not in NP.

We are also confronted with the complexities of nonlinear, or emergent, behavior. To facilitate construction of the proof, we are motivated to identify (at least) 4 variations on the composability problem. \square

We formalize the notions above as follows. For a set of objectives, O , if it is possible to verify in polynomial time that each $o_i \in O$ is satisfied, then we say the system objectives are *bounded*. If one or more of the objectives cannot be verified in polynomial time, we say the objectives are *unbounded*. An example of a bounded objective would be that a model exhibits medium fidelity where such a determination may be made by examining the algorithms in the model or a tag that denotes the validated model's assigned fidelity. Generally, bounded objectives are those can be verified by scanning the model. An example of an unbounded objective would be that a model termination condition will be met during model execution, or that a model is functionally equivalent to another model. As described in Section 2 such determinations have been shown to be undecidable.

Let \diamond denote the composition relation. We denote the composition of component a and component b as: $a \diamond b$. Generally, compositions may display a type of *emergent* behavior, that is, the capability of a composition may not be the simple sum, product, or union of the capabilities of the individual components within the composition. We state this formally in terms of the system objectives. Let \models denote the satisfies operator. If $a \models o$ we say that "*a satisfies o.*" If $a \not\models o$ and $b \not\models o$ but $(a \diamond b) \models o$ then we say the composition is *emergent*. If $a \not\models o$ and $b \not\models o$ and $(a \diamond b) \not\models o$ then we say the composition is *nonemergent*. We are now ready to define four variations of the general composability problem.

UNBOUNDED EMERGENT COMPOSABILITY

INSTANCE: A set O of unbounded objectives and a collection C of components demonstrating emergent behavior.

QUESTION: Is there a valid composition that meets the objectives stated in O ?

BOUNDED EMERGENT COMPOSABILITY

INSTANCE: A set O of bounded objectives and a collection C of components demonstrating emergent behavior.

QUESTION: Is there a valid composition that meets the objectives stated in O ?

UNBOUNDED NONEMERGENT COMPOSABILITY

INSTANCE: A set O of unbounded objectives and a collection C of components demonstrating nonemergent behavior.

QUESTION: Is there a valid composition that meets the objectives stated in O ?

BOUNDED NONEMERGENT COMPOSABILITY

INSTANCE: A set O of bounded objectives and a collection C of components demonstrating nonemergent behavior.

QUESTION: Is there a valid composition that meets the objectives stated in O ?

We consider the case of BOUNDED NONEMERGENT COMPOSABILITY. We show that it is NP-complete through a reduction to the SATISFIABILITY problem.

4.1 A review of the satisfiability problem

The satisfiability problem is a decision problem from Boolean logic. The following characterization of the problem is adopted from [9]. Let $U = \{u_1, u_2, \dots, u_m\}$ be a set of Boolean *variables*. A *truth assignment* for U is a function $t : U \rightarrow \{T, F\}$. If $t(u) = T$ we say that u is “true” under t ; if $t(u) = F$ we say that u is “false.” If u is a variable in U , then u and \bar{u} are *literals* over U . The literal u is true under t if and only if the variable u is true under t ; the literal \bar{u} is true if and only if the variable u is false.

A *clause* over U is a set of literals of U , such that as $\{u_1, \bar{u}_3, u_8\}$. It represents the disjunction of those literals and is satisfied by a truth assignment if and only if at least one of its members is true under that assignment. A collection C of clauses over U is *satisfiable* if and only if there exists some truth assignment for U that simultaneously satisfies all the clauses in C .

SATISFIABILITY

INSTANCE: A set U of variables and a collection C of clauses over U .

QUESTION: Is there a satisfying truth assignment for C ?

THEOREM 1 (*Cook’s Theorem*): SATISFIABILITY is NP-complete.

PROOF: see [4]. □

4.2 On the complexity of bounded non-emergent composability

THEOREM 2: BOUNDED NONEMERGENT COMPOSABILITY is NP-complete.

PROOF: It is easy to see that BNC \in NP since a nondeterministic algorithm may simply guess a composition and, since the objectives are bounded, may verify in polynomial time that the composition meets the objectives.

We transform SAT to BNC. Let the sets U (of Boolean variables) and C (of clauses) represent an arbitrary instance of SAT. We must construct a set O of objectives and a set B of components such that there exists $S \subseteq B$ that meets the objectives in O if and only if there is a satisfying truth assignment for C .

First, we observe that attainment of system objectives is a conjunctive property; all objectives must be met by S . Further, we observe that there may exist numerous ways to meet a given objective, o_i . We construct $P_i = \{p_{i,1}, p_{i,2}, \dots, p_{i,j}\}$, the set of properties that satisfy objective o_i . Without loss of generality, we assume that satisfaction of any property $p_{i,k} \in P_i$ satisfies o_i .³

³This assumption simplifies the form of the proof by allowing the composition problem to map “cleanly” to SAT which requires a Boolean expression in conjunctive normal form (CNF). Lewis and Papadimitriou [13, p. 403] describe a polynomial-time transformation for converting an arbitrary Boolean expression into CNF. By constructing our example directly in CNF, we have a clean mapping of clauses in SAT to objectives in BNC.

For convenience we denote the absence of property p as \bar{p} . For example, if p represents the ability of a component to run in real time, \bar{p} denotes that the component does not run in real time. Let $\hat{P} = \bigcup_{1 \leq i \leq |O|} P_i$.

The selection of a component, c , implies the introduction of some subset, $K_c \subseteq \hat{P}$ of properties. Since we assume that composition is nonemergent, $a \diamond b$ introduces a set of properties equal to the set union of the properties introduced by a and b individually. That is, if $K_a = X$ and $K_b = Y$ then $K_{a \diamond b} = X \cup Y$. We say that a composition $a \diamond b$ is *well-formed* if and only if $K_{a \diamond b}$ contains no contradictions. That is, $\neg \exists p \exists \bar{p} p \in K_{a \diamond b} \wedge \bar{p} \in K_{a \diamond b}$.

The reduction from SAT to BNC is straightforward. For each $u \in U$ define an equivalent $p \in \hat{P}$ and for each $c \in C$ form a set P . Let each P describe an objective $o \in O$. Without loss of generality, we will consider the case in which each property is related to one and only one component.⁴ Construct B as follows: for each $p \in \hat{P}$ define two components, one that introduces p and one that introduces \bar{p} .

Case 1: Suppose that there exists a satisfying assignment for U . Then there exists an assignment, q , to the literals in U that satisfies each of the clauses in C . It follows by construction that each $o \in O$ has some property p that can be satisfied. For each property, p , that is satisfied, add the component, c , that introduces p to the composition, S . The addition of c leaves S well-formed since both p and \bar{p} cannot be present unless both u and \bar{u} are present in q . Since we assumed the assignment for U would be a satisfying assignment, both u and \bar{u} cannot appear in q .

Case 2: Suppose that there exists a subset S of components that meets the objectives in O . Let P denote the set of properties satisfied by the components of S . Since S meets O , for each $o \in O$ there exists $p \in P$ that satisfies o . Since S is well-formed the set P does not contain contradictions. Therefore P represents a valid assignment for the variables in U . By construction, each clause of U is satisfied.

It follows that BNC is NP-complete. □

5 Conclusions and Future Directions

Components provide a useful “granularity” from which to address the notion of software reuse and significant attention has been paid to component-oriented technologies within the commercial software sector. Not surprisingly, perhaps, the degree of vendor-generated hyperbole surrounding components is high. One does not have to look hard to find a product announcement that hails a certain approach to component-based software development as the conquering silver bullet.

Of course, in reality components are not much more than a metaphor and organizing principle for software design—like objects, actors, agents, and so on. Arguably, they solve none of the fundamentally hard problems associated with the design, development and maintenance of software-based systems. For system developers, components may be useful, but are unlikely to be a panacea.

But what about components in the simulation domain? The simulation community has demonstrated a longstanding focus on providing support for the modeling task. From the early development of conceptual frameworks, modeling theories and methodologies, special-purpose simulation specification and programming languages, holistic simulation-support environments, and into the visions of web-based simulation, the

⁴In the general case a component may introduce numerous properties. However, we can convert any such case into a case where each component is related to a single property as follows. Let $C = \{c_1, c_2, \dots, c_j\}$ be the set of components that introduce property I . Define j properties replacing I by introducing the requirement for a particular component. That is, $I_i = I \wedge c_i, \forall 1 \leq i \leq j$.

modeler—and providing automated support for the modeling task—has been at the heart of many advances in simulation over the past forty years. How does composability effect the modeling task? Does composition make a modeler’s job easier or harder? Does composition facilitate automation or confound it?

Through the DARPA Advanced Simulation Technology Thrust (ASTT) we are considering composability as a design principle for simulation. In this paper we describe some of the risks attendant with building models by composition. We suggest that risks exist for both the horizontal and vertical composition of models. We describe a simple formal model of composition and use that model to identify a range of composability problems. Intuitively, identifying suitable compositions from a component repository is an intractable problem. We provide a formal NP-completeness proof that supports this intuition. The ramification of this result is, of course, that for repositories of sufficient size, brute-force search techniques will not be adequate. Approximate techniques such as heuristic search or branch and bound will be required. Selection of a suitable composition will involve a multivariable optimization problem—objectives may stipulate fidelity requirements, reliability requirements, efficiency requirements, and so forth. The optimization problem is further complicated by the presence of emergent behavior, which may not be effectively represented in component metadata, and by a stochastic response surface, which dictates the need to observe multiple replications of a composition before drawing conclusions about it. It seems clear, then, that composability can induce complexity into the modeling task as well as alleviate it.

Plans for further investigation involve continued exploration of the composability space. We have addressed the $O(2^n)$ scenario. Are there composition “use cases” with different complexities? For example, it seems reasonable that a use case for composition might be based on the product rule from statistics. Such cases would be polynomially solvable. Can use cases based on the combination rule or permutation rule be defined as well? In addition to our analysis of the composability problem with respect to computational complexity, we are also planning to relate composability as a system objective to other system objectives, such as maintainability, reliability, and correctness, using an Objectives-Principles-Attributes style approach [3].

6 Acknowledgements

This work is supported by the Defense Advanced Research Projects Agency (DARPA) under contract number DAAB07-99-C-C201 as part of the Advanced Simulation Technology Thrust (ASTT). Our thanks to former ASTT Program Manager Dell Lunceford and current Program Manager Larry Willis for their insights and support of this research. The views and conclusions herein are those of the authors and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency, The MITRE Corporation or the U.S. Government.

References

- [1] Allen, R.J. and Garlan, D. (1997). "A Formal Basis for Architectural Connection," *ACM Transactions on Software Engineering and Methodology*, **6**(3), pp. 213-249, July.
- [2] Allen, R.J., Garlan, D. and Ivers, J. (1998). "Formal Modeling and Analysis of the HLA Component Integration Standard," *SIGSOFT '98*, pp. 70-79, November.
- [3] Arthur, J.D. and Nance, R.E. (1987). "Developing an Automated Procedure for Evaluating Software Development Methodologies and Associated Products," Technical Report SRC-87-007, Systems Research Center, Virginia Tech, Blacksburg, VA, April.
- [4] Cook, S.A. (1971). "The Complexity of Theorem-Proving Procedures," In: *Proceedings of the Third Annual ACM Symposium on the Theory of Computing*, pp. 151-158.
- [5] Dahmann, J.S., Kuhl, F. and Weatherly, R. (1998). "Standards for Simulation: As Simple as Possible But Not Simpler – The High Level Architecture for Simulation," *Simulation*, **71**(6), pp 378-387, December.
- [6] Davis, P.K. and Bigelow, J. (1998a). "Introduction to Multiresolution Modeling (MRM) with an Example Involving Precision Fires," In: *Proceedings of SPIE: Enabling Technology for Simulation Science II*, pp. 14-27, Orlando, FL, 14-16 April 1998.
- [7] Davis, P.K. and Bigelow, J.H. (1998b) "Experiments in Multiresolution Modeling," available from the DARPA ASTT web site: <http://www.astt.com/>
- [8] Fishwick, P.A. (1998). "Issues with Web-Publishable Digital Objects," In: *Proceedings of SPIE: Enabling Technologies for Simulation Science II*, pp. 136-142, Orlando, FL, 14-16 April.
- [9] Garey, M.R. and Johnson, D.S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, NY.
- [10] Hinton, H.M. (1997). "Under-Specification, Composition and Emergent Properties," In: *Proceedings of the 1997 New Security Paradigms Workshop*, pp. 83-93, Langdale, Cumbria, UK.
- [11] Jacobson, S.H. and Yücesan, E. (1995). "Intractability Results in Discrete-Event Simulation," *Recherche Opérationnelle*, **29**(3), pp. 353-369.
- [12] Kuhl, F., Dahmann, J.S. and Weatherly, R.M. (1999). *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*, Prentice Hall.
- [13] Lewis, H.R. and Papadimitriou, C.H. (1981). *Elements of the Theory of Computation*, Prentice-Hall, Englewood Cliffs, NJ.
- [14] Luckham, D.C., Augustin, L.M., Kenney, J.J., Veera, J., Bryan, D. and Mann, W. (1995). "Specification and Analysis of System Architecture using Rapide," *IEEE Transactions on Software Engineering*, April.
- [15] Medvidovic, N. (1996). "A Classification and Comparison Framework Software Architecture Description Languages," Technical Report UCI-ICS-97-02, Dept. of Information and Computer Science, University of California, Irvine, Irvine, CA, February.
- [16] Miller, G and Zabek, A. (1996). "The Joint Training Confederation and the Aggregate Level Simulation Protocol," *Phalanx*, **29**, pp. 24-27.
- [17] Milner, R., Parrow, J. and Walker, D. (1989). "A Calculus of Mobile Processes (Part 1 and 2)," LFCS Laboratory for Foundation of Computer Sciences, University of Edinburgh, June.
- [18] Nance, R.E. and Overstreet C.M. (1987). "Diagnostic Assistance Using Digraph Representations of Discrete Event Simulation Model Specifications," *Transactions of the Society for Computer Simulation*, **4**(1), pp. 33-57, January.

- [19] Nance, R.E. and Overstreet, C.M. (1987). "Exploring the Forms of Model Diagnosis in a Simulation Support Environment," *Proceedings of the 1987 Winter Simulation Conference*, pp. 590-596, Atlanta, GA, December 14-16.
- [20] Nance, R.E., Overstreet, C.M. and Page, E.H. (1996). "Redundancy in Model Representation: A Blessing or a Curse?," In: *Proceedings of the 1996 Winter Simulation Conference*, pp. 952-958, Coronado, CA, 8-11 December.
- [21] Nierstrasz, O. (1991). "Toward an Object Calculus," In: *Proceedings of ECOOP '91 Workshop on Object-Based Concurrent Computing*, pp. 1-20, July.
- [22] Overstreet, C.M. (1982). "Model Specification and Analysis for Discrete Event Simulation," PhD Dissertation, Department of Computer Science, Virginia Tech, Blacksburg, VA, December.
- [23] Overstreet, C.M. and Nance, R.E. (1985). "A Specification Language to Assist in Analysis of Discrete Event Simulation Models," *Communications of the ACM*, **28**(2), pp. 190-201, February.
- [24] Overstreet, C.M. and Nance, R.E. (1986). "World View Based Discrete Event Model Simplification," *Modelling and Simulation Methodology in the Artificial Intelligence Era*, M.S. Elzas, T.I. Ören and B.P. Zeigler (Eds.), North-Holland, pp. 165-179.
- [25] Page, E.H. (1998) "The Rise of Web-Based Simulation: Implications for the High Level Architecture," In: *Proceedings of the 1998 Winter Simulation Conference*, pp. 1663-1668, Washington, DC, 13-16 December 1998.
- [26] Page, E.H. and Nance, R.E. (1999). Incorporating Support for Model Execution within the Condition Specification," to appear in *Transactions of the Society for Computer Simulation*.
- [27] Page, E.H., Buss, A., Fishwick, P.A., Healy, K.J., Nance, R.E. and Paul, R.J. 1998. "The Modeling Methodological Impacts of Web-Based Simulation," In: *Proceedings of the 1998 SCS International Conference on Web-Based Modeling and Simulation*, pp. 123-128, San Diego, CA, 11-14 January.
- [28] Page, E.H., Buss, A., Fishwick, P.A., Healy, K.J., Nance, R.E. and Paul, R.J. 1999. "Web-Based Simulation: Revolution or Evolution?" submitted to: *ACM Transactions on Modeling and Computer Simulation*, February.
- [29] Reynolds, P.F., Natrajan, A. and Srinivasan, S. (1997). "Consistency Maintenance in Multiresolution Simulations," *ACM Transactions on Modeling and Computer Simulation*, **7**(3), pp. 368-392, July.
- [30] Törn, A.A. (1981). "Simulation Graphs: A General Tool for Modeling Simulation Designs," *Simulation*, **37**(6), pp. 187-194, December.
- [31] Yücesan, E. (1989). "Simulation Graphs for the Design and Analysis of Discrete Event Simulation Models," PhD Dissertation, Cornell University, Ithaca, NY.
- [32] Yücesan, E. and Jacobsen, S.H. (1992). "Building Correct Simulation Models is Difficult," In: *Proceedings of the 1992 Winter Simulation Conference*, pp. 783-790, Arlington, VA, December 13-16.
- [33] Yücesan, E. and Jacobson, S.H. (1996). "Structural and Behavioral Equivalence of Simulation Models," *ACM Transactions on Modeling and Computer Simulation*, **6**(1), pp. 53-75, January.
- [34] Vion-Dury, J.-Y., Bellisard, L. and Marangozov, V. (1997). "A Component Calculus for Modeling the Olan Configuration Language," Institut National De Recherche en Informatique et en Automatique, Report No. 3231, August.