A CASE STUDY OF MODEL CONTEXT FOR SIMULATION COMPOSABILITY AND REUSABILITY

Michael Spiegel
Paul F. Reynolds, Jr.
David C. Brogan

Modeling and Simulation Technology Research Initiative
151 Engineer's Way, P.O. Box 400740
Department of Computer Science, University of Virginia
Charlottesville, VA 22904-4740, U.S.A.

ABSTRACT

How much effort will be required to compose or reuse simulations? What factors need to be considered? It is generally known that composability and reusability are daunting challenges for both simulations and more broadly software design as a whole. We have conducted a small case study in order to clarify the role that model context plays in simulation composability and reusability. For a simple problem: compute the position and velocity of a falling body, we found that a reasonable formulation of a solution included a surprising number of implicit constraints. Equally surprising, in a challenge posed to a small group of capable individuals, no one of them was able to identify more than three-quarters of the ultimate set of validation constraints. We document the challenge, interpret its results, and discuss the utility our study will have in future investigations into simulation composition and reuse.

1 INTRODUCTION

Simulation composability and reusability are highly desirable goals in simulation design. A component that can be reused multiple times or used in combination with other components can save a great deal of time, money, and human effort (Davis and Anderson 2003). Composability and reusability are difficult to achieve, however, because they require that components work under a range of possible contexts and that they can be validated under a range of possible business requirements.

Models are generally viewed in terms of an encompassing context (Yilmatz 2004). We define context in terms of Zeigler et al's experimental frames. Zeigler, Praehofer, and Kim define an experimental frame as a specification of the conditions under which a system is observed or experimented (2000). For a given system S, let P denote the set of all possible experimental frames of S. We define the *context* of a model as the set of all experimental frames under which the model is valid. This context C is a subset of P. Further, let C' be the relative complement of C in P. C' the set of all experimental frames under which a model is invalid. Define the set of all rules that decide whether an experimental frame belongs in C or C' as the *validation constraints* of a model. The validation constraints will be referred to in abbreviated form in this paper as simply the *constraints of a model*.

The simulation community needs a systematic approach for defining the valid context of a model (Kasputis and Ng 2000, Page and Opper 1999, Petty and Weisel 2003b). In the interest of developing an approach, we have undertaken a case study of identifying validation constraints for a relatively simple model. Our goal was to gain insight into the difficulties associated with capturing relevant constraints, measured in terms of effort required and likelihood of success.

Our early perception of the model we studied was that it was relatively simple. It consists of a perfectly smooth sphere that falls through an incompressible Newtonian fluid. The only two forces that act upon this sphere are gravity and drag forces. One could easily hypothesize that identification of validation constraints would be straightforward and not an overwhelming task for a reasonably talented individual. If the challenges associated with identifying constraints for the falling body problem are any measure, the task is anything but simple. We observed that talented individuals are likely to miss critical constraints and that the number of unstated constraints can be substantial. Clearly, systematic approaches to capturing model context will need to be comprehensive.

In order to provide the reader with an unbiased opportunity to experience firsthand the challenges we observed, we refrain from summarizing our results further until later. Our recommended approach to the remainder of this paper is to read through the model description, engage in the challenge, and then compare personal findings with those we report. The experience will be enlightening.

Our case study serves as a guidepost for the creation of best practices in enumerating validation constraints. It is clear that model composability and reusability require the specification of validation constraints. And while the final best practices for determining the constraints may deviate from our chosen methodology, the end results must remain the same.

In the next section we review the literature of context in component-based software design. Section 3.1 describes our process of determining constraints. Section 3.2 is a complete description of the falling body model used in the case study. Section 3.3 present a taxonomy of the constraints found in the model. Section 3.4 evaluates the usefulness of identifying validation constraints, and section 3.5 examines the constraints under a requirement of semantic composability. Section 4 discusses the implications of the study on composition and reuse, and the final section offers suggestions on where to proceed in the future. The complete list of constraints for the falling body model can be found in the Appendix.

2   RELATED WORK

A survey of simulation composability and component-based software design (CBSD) found that both domains must pay close attention to semantics (Bartholet et al. 2004). Component-based software design has been largely successful in limited domains where semantic composability is successfully enforced by the underlying syntactic composability. As CBSD scales beyond simple components such as mathematical libraries and GUI tool sets, the problem of semantic composability will plague both the CBSD and simulation communities.

Garlan, Allan, and Ockerbloom conducted a case study on the difficulties of CBSD in a large-scale system (1995). They identified several categories of "architectural mismatches" while trying to build an integrated development environment (IDE) out of existing subsystems. An architectural mismatch is an incorrect assumption on the part of a component about the environment in which it will operate. The authors of that study note that the creators of the original components were neither "lazy, stupid, nor malicious." Yet the existence of these undocumented assumptions led to massive challenges in the construction of their integrated development environment.

We have conduced an analogous case study to identify the types of undocumented context assumptions that appear in simulation composition and reuse. The taxonomy of architectural mismatches in the Garlan, Allan, and Ockerbloom study consists of four main categories that relate to the underlying abstract architecture of a component-based software design. A taxonomy of simulation context semantics should somehow be organized around characteristics that are specific to simulations (Carnahan, Brogan, and Reynolds 2005).

Another field that has already recognized context assumptions as a major source of software errors is safety-critical computing. The high cost of failure in the safety-critical community has motivated the study of determining from where these errors come. The most common sources of safety-critical errors are thought to be "(1) discrepancies between the documented requirements specifications and the requirements needed for correct functioning of the system, and (2) misunderstandings of the software's interface with the rest of the system." (Lutz 1993) A failure to accurately enumerate these functional requirements and interface requirements can lead to a system failure that will lead to loss of life or monetary expense.

The ability to accurately communicate business requirements among groups of people is still an open problem (Hayhurst and Holloway 2001). Hanks, Knight, and Strunk have explored the use of linguistic analysis on the communication channels of requirements design (2001). They suggest using a reference structure that recursively stores the definitions of all domain-specific terms. This structure, known as a domain map, serves as a systemic and complete repository of the semantics for a particular knowledge domain. The fundamental motivation of the domain map is "the principle of making the implicit explicit." A similar approach to composability and reusability specifications in the modeling and simulation community would prove useful.

3   IDENTIFYING VALIDATION CONSTRAINTS

3.1   Falling Body Challenge

The primary objective of our investigation was to gain insight into the complexity of capturing validation constraints. In the process of creating a list of constraints for the falling body model, we observed that different groups of individuals had a tendency to compose different lists of orthogonal constraints. That is, when a particular team created a list, they tended not to enumerate a set of shared implied constraints that a different team identified.

In light of the unidentified constraints that arose in composition we chose a competitive approach to enumerating validation constraints. A group of nine graduate students, undergraduate students, and faculty participated in an amicable contest, with a small prize offered to the winner, to identify the greatest number of validation constraints. Each contestant was given a description of the model exactly as it appears in Section 3.2. They were instructed to enumerate all reference frames in which they might reuse or compose this model but model validation would fail. The contestants were told to assume syntactic composability (Petty and Weisel 2003a). An example of a constraint we expected the contestants to identify was "the mass of the falling body remains constant over time."

Participants were instructed to ignore any constraints related to the implementation of the model. In the falling body simulation, there are multiple implementation choices of numerical methods and numerical precision. In computing position as a function of time, what sort of numerical integration method should be employed? What effect would it have on correctness of results? We have chosen not to discuss implementation assumptions in order to focus attention on the model instead of the simulation. When joining composable simulations it will be necessary to validate both the combined model and then to validate the combined simulation.

After a period of two weeks submissions were tabulated and a master list was created and discussed among the contestants and others. We defer discussion of the results to after the presentation of the challenge in the next section.

3.2 Falling Body Model

The following model is an extended version of the model presented in Appendix A of the monograph by Davis and Anderson (2003). A sphere is falling through some medium and experiencing drag as it falls. Let $p(t)$ equal the position of the sphere at time t and $p(0) = p_0$ be the initial position. Let $v(t) = p'(t)$ equal the velocity of the sphere at time t, and let $v(0) = v_0$ be the initial velocity. Calculate $p(t)$ and $v(t)$ for all $t \geq 0$.

The sphere is perfectly smooth, it has diameter d and mass m. The medium has uniform density ρ_f and uniform kinematic viscosity ν. Assume that when the sphere impacts with the earth, $p(t_{earth}) = 0$, it will remain on the ground for all $t > t_{earth}$.

The following forces will act on the sphere (Chow 1979):

- Gravity: The sphere experiences constant acceleration, $g \approx 9.8$ m/s^2.
- Buoyancy: $m_f = (1/6)\pi d^3 \rho_f$ against gravity.
- Inertial drag: $(1/2) m_f v'(t)$
- Viscous drag: $(1/2)\rho_f \cdot v(t) \cdot |v(t)| \cdot \pi/4 \cdot d^2 \cdot c_d(v(t))$
- Wave drag: Wave drag is negligible at subsonic speeds.

We apply Newton's Second Law to determine acceleration, and then employ numerical methods of integration to calculate velocity and position.
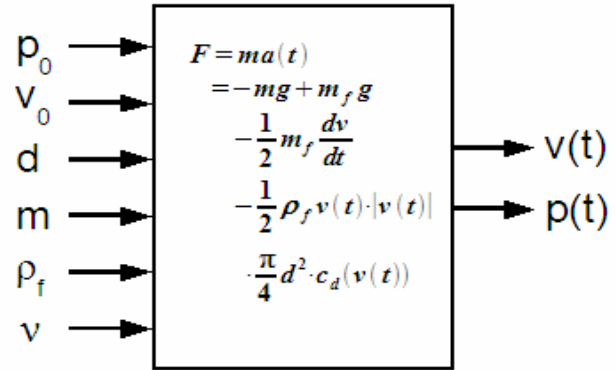

Figure 1: Falling Body Model

The term c_d is the drag coefficient and it is defined as a function of the Reynolds number, which is $Re = v(t) \cdot d / \nu$. The drag coefficient is determined experimentally as a function of the Reynolds number. Both the drag coefficient and the Reynolds number are dimensionless values. For a perfectly smooth sphere, c_d can be approximated piecewise with the following function,

$$c_d = \begin{cases} \frac{24}{Re} & 10^{-2} < Re \leq 1 \\ \frac{24}{(Re)^{0.646}} & 1 < Re \leq 400 \\ 0.5 & 400 < Re \leq 3 \cdot 10^5 \\ 0.000366 \cdot (Re)^{0.4275} & 3 \cdot 10^5 < Re \leq 2 \cdot 10^6 \\ 0.18 & 2 \cdot 10^6 < Re \leq 10^7 \end{cases}$$

The reader interested in pursuing the challenge without bias from the challenge results should pause at this point, continuing when identification of unstated constraints is completed. We discuss the results of the challenge next.

3.3 Challenge Results and Constraint Taxonomy

The competition produced a master list of twenty-nine validation constraints (see Appendix). From them we have derived a taxonomy of validation constraint types. Every effort has been made to remove redundant constraints and to represent identified validation constraints as concisely as possible. We make no claim to having found all validation constraints for the falling body model.

It is illuminating to consider that the top three contestants identified only 21, 19, and 16 constraints, respectively, out of the master list (see Table 1). No single participant was capable of identifying more than three-quarters of all currently identified constraints. Like the component designers of (Garlan, Allan, and Ockerbloom 1995) our challenge participants were neither "lazy, stupid, nor malicious." Each participant failed to identify several implicit constraints that other participants explicitly identified. The

Table 1: Falling Body Challenge Results

Participants	Validation Constraints

Participants	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Alice	X	X	X	X	X	X	X	X	X		X	X		X		X	X			X	X		X	X		X	X		X
Bob	X	X		X	X	X						X	X	X	X		X		X		X	X				X	X	X	X
Carol	X	X				X	X	X	X	X	X			X	X		X				X		X				X	X	X
Dave	X	X							X			X	X	X	X		X		X		X					X	X		X
Edna	X					X	X					X	X	X	X			X	X	X						X	X		
Fred	X	X	X			X	X					X	X	X	X		X									X	X		
Gracie	X		X									X	X	X	X					X						X		X	X
Herb		X				X						X								X	X		X						
Irene						X																					X	X	

inability of any single individual to enumerate more than three-quarters of the currently identified constraints emphasizes the claim that identifying validation constraints is not a trivial task.

The Appendix contains the master list of validation constraints. We have identified three major categories of constraints in the falling body model: Invariant Constraints, Dynamic Constraints, and Inter-Object Constraints.

Our taxonomy is based on the types of object attributes in the constraints (invariant versus dynamic) and on the object scope of the constraints (one versus many). An object attribute is a synonym for any property of the object. Attributes can be invariant or dynamic. Invariant attributes do not change over time, while dynamic attributes do change over time. Another term for a dynamic attribute is a behavioral attribute, which is defined as an aspect of the model that changes over time (Malak and Paredis 2004).

Invariant constraints must be guaranteed by the builder of a component. Once they are guaranteed the user of a component must verify these constraints only once, initially. If the constraint is verified at the initial time and it is guaranteed to be invariant, then it does not need to be verified at runtime.

Dynamic constraints must be reevaluated any time a change is made to the underlying behavioral attribute or attributes. In the case of the falling body model, the only dynamic attributes are the position and velocity of the falling body. Therefore all of the dynamic constraints must apply to a subset of these two attributes.

Inter-object constraints define a relationship between two or more types of objects, whereas invariant and dynamic constraints can only apply to one object type. Included in the inter-object constraints are all of the interaction semantics and all the constraints that must compare the attributes of two or more object types. Inter-object constraints highlight the importance of an interaction semantics. And interaction semantics are important in the

design of composable simulations as discussed further in Section 3.5.

The taxonomy of constraints divides the validation constraints along simulation-specific characteristics. All dynamic simulations employ some concept of changing state over time. The notion of change over time leads to the classification of invariant constraints and dynamic constraints. We make no claim about the completeness of our taxonomy of constraints. It is likely that investigations of other types of simulations will reveal new categories of validation constraints. Building a taxonomy of validation constraints helps in the identification of implicit unenumerated constraints.

3.4 Is the Effort Necessary?

It is worthwhile to investigate the usefulness of identifying all known validation constraints. Many of the constraints listed in the Appendix appear farfetched and one may argue that they would not be invoked under average conditions of use. Two examples of "farfetched" constraints include the Uncertainty Principle constraint and the General Relativity constraint, which place limits on extreme values of the sphere's invariant attributes. High energy physicists don't find the uncertainty principle farfetched, nor do engineers involved in the programming of GPS satellites fail to appreciate the importance of relativity constraints (Yam 2004). "Farfetched" in today's science is routine in tomorrow's.

What does a potential user of our falling body model face? We consider two examples in which selected constraints are selectively violated. These examples illustrate the type of analysis that must be undertaken to recreate the model under a new set of constraints. The two examples we consider are a dimpled golf ball dropped from a chosen altitude and a cannon ball fired out of a cannon.

Golf balls are dimpled so that they will travel farther when hit. If we wish to model the path of the golf ball, we must relax the Smooth Property constraint for the sphere.

Adding dimples to a sphere will change the values of the drag coefficient, c_d. The laminar-to-turbulent boundary of the drag coefficient is pushed backwards from 4×10^5 to 4×10^4 (Shaughnessy, Katz, and Schaffer 2005). The golf ball will now travel a longer distance. Breaking the Smooth Property constraint requires recomputing a new drag coefficient function for the particular roughness of the sphere.

In World War I, it was discovered that firing a cannon at a higher angle than the believed maximum range resulted in an increase in the range of the shell. This is due to the lower atmospheric pressure at higher altitudes (Chow 1979). We must make several syntactic and semantic changes to the falling body model in order to incorporate this phenomenon. First, to study the ballistics of the cannon the coordinate system of the model is expanded to two dimensions. The force equation of Figure 1 must be separated into a F_x component and a F_y component. The cannon will be fired with an initial velocity of 800 m/s. At this supersonic speed the drag coefficient is a function of Mach number and Reynolds number. We must either specify the new drag coefficient function or decide upon a suitable approximation for wave drag. Finally the atmospheric density must be specified as a function of the cannon ball's altitude. All the other constraints of our reference frames do not change.

Failure to appreciate the importance of various constraints when selecting a model can lead to unacceptable results. While it may be that our original formulation for the falling body is a suitable approximation for a golf ball or cannon ball in flight, that decision should be made knowledgeably by a designer. Such a knowledgeable decision can only be made if the constraints associated with a model are identified and understood.

3.5 Relaxing the Closed System Constraint

Any model with the Closed System constraint ("the model does not interact with any external objects") does not interact with external components. In place of the Closed System constraint we must introduce a precise semantics of interactions. The interaction semantics must specify all interaction behaviors between objects inside and outside the current system.

In the falling body problem interaction semantics are defined by specifying interactions between every pair of objects in the system. In a more general composable simulation, interaction semantics should be specified by using a type system for all objects in the system. Interactions must be specified for every pairing of object types. This requires foreknowledge of basic object types in a library of composable components.

As a practical matter one can leave unspecified any null interactions between two object types. In order to ensure precise semantics the meta-constraint must be made

that unspecified interactions default to null interactions. However this meta-constraint can lead to undesirable behavior if several model components are connected and novel cross-component interaction pairs are unintentionally left unspecified.

4 DISCUSSION

We conclude the following from our study: (1) identifying validation constraints is challenging, (2) simulation composition and reuse will require comprehensive identification of constraints, and (3) valid composition and reuse requires identifying methods for capturing critical constraints efficiently, and for maintaining them in the face of design changes.

In component-based software design several techniques have been developed to improve composability. Sullivan and Knight have shown that large-scale software components can be quickly and easily composed without architectural mismatches (1996). A design fault analysis tool was developed of about several million lines of code in the span of one person week of effort. The business logic of the application consists of about 10,000 lines of code; the remaining lines consist of library code.

A primary conclusion of the Sullivan and Knight study is that composability is easier when the components must conform to an overarching set of requirements. Or, in their own words, "if components are to be composable, they have to be designed for it." This same technique can be applied in the modeling and simulation community. A library of components may specify constraints on the overarching system it can modify. For example a library could specify that all of its components are not subject to the uncertainty principle, Brownian motion, special relativity, or general relativity.

Another useful tool in model context is the ability to quantify the error that is associated with deviating from the context. Malak and Paredis have developed the concept of a validity description to quantify this deviation (2004). A validity description formally defines an upper bound on a model's inaccuracy under a fixed set of conditions. This approach is successful when a small subset of constraints are allowed to relax. The validity description can be efficiently characterized in a small domain space. Under the entire model context, the modeler must choose which constraints to hold constant and which constraints are allowed to vary.

5 FUTURE WORK

We have conducted a small case study to determine the impact of validation constraints on the reusability of a model. The model itself was simple. The number of implicit constraints associated with it was surprisingly large. In an informal contest related to our study, no participant

identified more than 75% of the ultimate set of constraints identified. Borrowing from Garlan, Allan, and Ockerbloom our challenge participants were neither "lazy, stupid, nor malicious." (1995)

We believe the study reported here can be useful to the reader beyond the results above. The falling body model presents a fine example for testing any proposed reusability process. If the process cannot lead to the efficient extraction of the constraints listed in the Appendix, then it is of questionable value.

In the future we anticipate further study of our initial taxonomy of validation constraints. Will other types of simulations yield new categories of constraints? The taxonomy is useful only if it can serve as a general guidepost that suggests hidden constraints that have not been identified. Additionally the taxonomy for the simulation community may benefit from insights in the larger domain of software design. Generic software applications contain properties that are identified as invariant or time-dependent. Can the lessons from formal software analysis be applied to our objectives? We will be exploring these issues.

ACKNOWLEDGMENTS

APPENDIX: FALLING BODY CONSTRAINTS

1. Invariant Constraints

1.a Sphere Attributes

1. Sphere Property - The body is a sphere and it remains spherical.
2. Smooth Property - The body is smooth and it remains smooth.
3. Impermeable Property - The body is completely impermeable.
4. Initial Velocity - The body has an initial velocity of v_0 that has no horizontal component of motion.
5. Angular Velocity - The body has no initial angular velocity.
6. Constant Mass - The mass of the body remains constant over time. The body does not experience ablation or accretion.

7. Constant Diameter - The diameter of the body remains constant over time.
8. Distribution of Mass - The body has a centrally symmetric mass distribution that remains constant over time.
9. Uncertainty Principle - The diameter of the body is much greater than the Plank length.
10. Brownian Motion - The mass and diameter of the body are large enough such that Brownian motion of the fluid has negligible impact on the body.
11. General Relativity - The mass of the body is low enough to ignore the gravitational curvature of space-time.

1.b Fluid Attributes

12. Fluid Density - The fluid density is constant. The fluid is incompressible.
13. Fluid Pressure - The fluid pressure is constant.
14. Fluid Temperature - The fluid temperature is constant.
15. Kinematic Viscosity - The kinematic viscosity is constant. The medium is a Newtonian fluid.
16. Stationary Fluid - The fluid is stationary apart from being disturbed by the falling body.
17. Infinite Fluid - The volume of the fluid is large enough to completely envelope the sphere. The movement of the fluid is not restricted by a container such as a pipe or tube.

1.c Earth Attributes

18. Flat Terrain - The ground does not have terrain and remains flat for all $t > 0$.
19. Coriolis Effect - The Earth is not rotating. We ignore the Coriolis effect.

2. Dynamic Constraints

20. Mach Speed - The velocity of the body is sufficiently less than the speed of sound for that medium.
21. Special Relativity - The velocity of the body is sufficiently less than the speed of light for that medium.
22. Reynolds Number - The Reynolds number remains between 10^{-2} and 10^7 for all $t > 0$. The Reynolds number is a function of velocity.

3. Inter-Object Constraints

23. Sphere/Fluid Interaction - The body and the fluid interact only through buoyancy and drag. For example, the body cannot dissolve in the fluid, nor can the body transfer heat to the fluid.

24. Sphere/Earth Interaction - The body and the earth interact only through the gravitational force.
25. Fluid/Earth Interaction - The fluid and the earth do not interact.
26. Closed System - The Earth, sphere, and fluid do not interact with any other objects.
27. Simple Gravity - Gravity is a constant downward force of 9.8 m/s^2.
28. One-Sided Gravity - The mass of the body is much less than the mass of the Earth. The Earth is not affected by the gravitational pull of the body.
29. Inelastic Collision - The collision between the sphere and the ground is perfectly inelastic.

REFERENCES

Bartholet, R., D.C. Brogan, P.F. Reynolds, and J.C. Carnahan 2004. In search of the philosopher's stone: Simulation composability versus component-based software design. In *Proceedings of the Fall 2004 Simulation Interoperability Workshop*, Orlando, FL.

Carnahan, J.C., D.C. Brogan and P.F. Reynolds. Simulation-specific characteristics. Submitted for publication.

Chow, C. 1979. Introduction to computational fluid mechanics. Hoboken, New Jersey: John Wiley & Sons Inc.

Davis, P.K. and R.H. Anderson 2003. Improving the composability of Department of Defense models and simulations, Rand Corporation Report. http://www.rand.org/publications/MG/MG101/.

Garlan, D., R. Allen, and J. Ockerbloom 1995. Architectural mismatch or why it's hard to build systems out of existing parts. In *Proceedings of the Seventeenth International Conference on Software Engineering*. Seattle WA.

Hanks, K.S., K.C. Knight, and E.A. Strunk. 2001. Erroneous requirements: a linguistic basis for their occurrence and an approach to their reduction. In *Proceedings of the 26th Annual NASA Goddard Software Engineering Workshop*. Greenbelt, MD

Hayhurst, K.J. and C.M. Holloway 2001. Challenges in software aspects of aviation systems. In *Proceedings of the 26th Annual NASA Goddard Software Engineering Workshop*. Greenbelt, MD

Kasputis, S. and H.C. Ng 2000. Composable simulations. In *Proceedings of the 2000 Winter Simulation Conference*, ed. J.A. Joines, R.R. Barton, K. Kang, and P.A. Fishwick, 1577-1584.

Lutz, R. R. 1993. Analyzing software requirements errors in safety-critical, embedded systems. In *Proceedings of the IEEE International Symposium on Requirements Engineering*. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Malak, R. J., and C.J.J. Paradis 2004. Foundations of validating reusable behavioral models in engineering design problems. In *Proceedings of the 2004 Winter Simulation Conference*, ed. R.G. Ingalis, M.D. Rossetti, J.S. Smith, and B.A. Peters, 420-428. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Page, E.H. and J.M. Opper 1999. Observations on the complexity of composable simulation. In *Proceedings of the 1999 Winter Simulation Conference*, ed. P.A. Farrington, H.B. Nembhard, D.T. Sturrock, and G.W. Evans, 553-560. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Petty, M.D. and E.W. Weisel 2003a. A composability lexicon. In *Proceedings of the Spring 2003 Simulation Interoperability Workshop*, 181-187. Simulation Interoperability Standards Organization.

Petty, M.D. and E.W. Weisel 2003b. A formal basis for a theory of semantic composability. In *Proceedings of the Spring 2003 Simulation Interoperability Workshop*, 416-423. Simulation Interoperability Standards Organization.

Shaughnessy, E.J., I.M. Katz, and J.P. Schaffer. 2005. Introduction to fluid mechanics. New York: Oxford English Press.

Sullivan, K.J. and J.C. Knight 1996. Experience assessing an architectural approach to large-scale, systematic reuse. In *Proceedings of the 18th International Conference on Software Engineering (ICSE18)*, 220-229. Berlin, Germany, March 25-29.

Yam, P. September 2004. Everyday Einstein. *Scientific American*, 50-55.

Yilmatz, L. 2004. On the need for contextualized introspective models to improve reuse and composability of defense simulations. *Journal of Defense Modeling and Simulation* 1 (3): 141-151.

Zeigler, B.P., H. Praehofer, and T.G. Kim. 2000. Theory of Modeling and Simulation, 2nd Edition. Burlington, MA: Academic Press.

AUTHOR BIOGRAPHIES

MICHAEL SPIEGEL is a Ph.D. Candidate in Computer Science and a member of the Modeling and Simulation Technology Research Initiative (MaSTRI) at the University of Virginia. Michael earned his B.A. in Computer Science at Swarthmore College. He has previously held the position of research associate at StreamSage, Inc. studying natural language processing for multimedia search engines. His email addresses is mspiegel@cs.virginia.edu and his web address is www.cs.virginia.edu/~ms6ep.

PAUL F. REYNOLDS, JR. is a Professor of Computer Science and a member of MaSTRI at the University of Virginia. He has conducted research in Modeling and Simulation for over 25 years, an has published on a variety of M&S topics, including parallel and distributed simulation, multi-resolution modeling and coercible simulations. He has advised industrial and government agencies on matters relating to modeling and simulation. He is a plank holder in the DoD High Level Architecture. His email address is reynolds@virginia.edu.

DAVID C. BROGAN earned his Ph.D. from Georgia Tech and is currently an Assistant Professor of Computer Science and a member of MaSTRI at the University of Virginia. For more than a decade, he has studied simulation, control, and computer graphics for the purpose of creating immersive environments, training simulators, and engineering tools. His research interests extend to artificial intelligence, optimization, and physical simulation. His email address is brogan@virginia.edu.