Proceedings of the 2010 Winter Simulation Conference
B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds.

APPLYING A MODEL DRIVEN APPROACH TO COMPONENT BASED MODELING AND SIMULATION

Deniz Cetinkaya
Alexander Verbraeck
Mamadou D. Seck

Systems Engineering Group, Faculty of Technology, Policy and Management
Delft University of Technology
Jaffalaan, 5, 2628BX, Delft, THE NETHERLANDS

ABSTRACT

Hierarchical component based modeling and simulation holds great promise, especially in terms of modeling efficiency and model reuse. However, in practice, the approach has not yet lived to its potential. After a diagnosis of this state of affairs, a solution inspired from model driven engineering is proposed. The basic architecture of the framework is explained, based on meta-models, models, and their respective relations. Finally a usage workflow is provided, describing how the framework can be used by different actors within a simulation lifecycle.

1 INTRODUCTION

Simulation models are becoming more and more complex and large. When each simulation model is designed from scratch, the lack of reusability makes simulation a time consuming and expensive task. Reuse has the potential of reducing the cost of specification, coding, validation and verification.

Reuse in simulation modeling refers to the development of new models using pre-existing modeling elements like parts of a simulation code, functions, simulation components, and even similar simulation models. (Pidd 2002) emphasizes four different types of model reuse: code scavenging (reusing existing code), function reuse (reusing predefined functions that provide specific functionalities), component reuse (reusing encapsulated simulation modules that provide a well-defined interface) and full model reuse (reusing a pre-existing model). In the real life, reuse is more likely to occur in reusing existing code and functions. Although component reuse and model reuse has been a goal for a long time, it has not been achieved effectively.

Introducing a component based approach into the simulation field can help managing larger models, since it promises reuse of interoperable components and hierarchical modeling.

Looking at the existing simulation methodologies, at the conceptual modeling level, a top-down approach is adopted. The modeler first partitions the system into the relevant subsystems and defines the relationships between the latter, without delving yet into their inner details. For example, to represent an airport system, one would identify such subsystems as gates, security check points, information desk, check-in desks and so forth. Hierarchical decomposition is a commonly used strategy for managing complexity (Simon 1962). At the simulation model construction level, a bottom-up composition approach is commonly adopted. Basic primitives available in the modeling language are aggregated to provide the desired functionality of the identified subsystems.

Conceptual modeling is recognized as an important step in a simulation study which generally benefits from diagramming techniques. A range of methods have been used for representing simulation conceptual models, such as process flow diagrams, event graphs, UML, simulation activity diagrams, etc. (Robinson 2006). However, there is no commonly accepted conceptual modeling technique for simulation. Thus, the decomposition of a complex system can be done in various ways, depending on the paradigm adopted by the chosen diagraming technique. Besides, conceptual modeling techniques being at best semi-formal, they are often not reused explicitly in the further steps of the simulation process, as formal transformation methods are not available to guarantee model continuity. This

means that, based on exactly the same conceptual model, different simulation modelers will most likely create different simulation models. This puts an excessively high share of simulation project success responsibility in the hands of the code writer. This situation would have been mitigated if stakeholders were involved in the design of the conceptual models, and if the latter were reused explicitly in the further stages of the process.

Although some pre-packaged commercial tools such as Arena, MATLAB-Simulink, Plant Simulation or Enterprise Dynamics offer solutions for easing the simulation model construction process, progress can still be made at this stage. A clear metamodel and a rigorous formalism are lacking in many cases. The general tradeoff in model construction tooling seems to be that of *flexibility vs. rigor*. On the one hand, some tools are very flexible, close to the intuitive way modelers think about their system of interest, and with little constraints, leading to informal models with high probability of errors. On the other hand, other tools are very strict, expressed in intricate mathematical formalism, with steep learning curves, and resulting in overly complex code limiting the possibilities for reuse. In any case the model components are usually not platform-independent and not compatible with other components developed in different environments.

In short, improving reuse in the existing modeling methodologies requires some development in the state of the art of conceptual modeling and simulation model construction. Figure 1 illustrates the top-down and bottom-up approaches in conceptual modeling and simulation model construction, respectively. To foster reuse, an integrated modeling framework is needed.

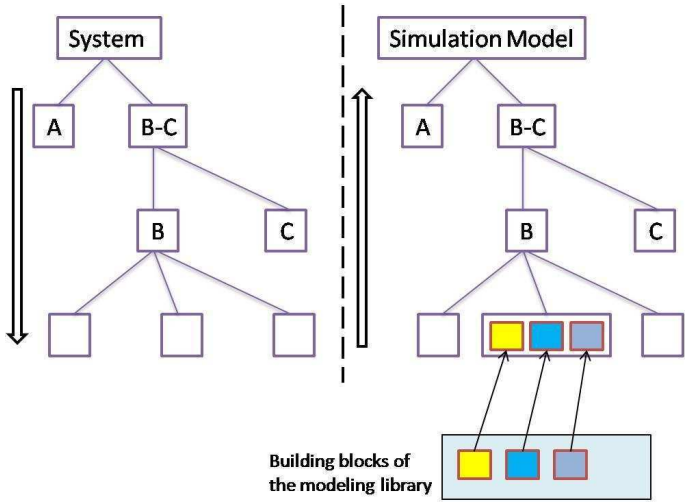

Figure 1: Hierarchical modeling approach.

In order to provide reusability and interoperability, it is useful to have generic model definitions that are independent from the implementation details. A higher level representation on top of the rigid simulation model implementation is expected to make the simulation model development process faster. We think that metamodeling and the model-driven development (MDD) approaches can help reduce the gap between system definition and simulation model construction steps through automated model transformations. Applying a model-driven approach into component based modeling and simulation can solve reusability and interoperability problems. In a similar effort, (Bakshi, Prasanna, and Ledeczi 2001) presents a model based extensible framework to facilitate embedded system design and optimization, namely MILAN(Model based Integrated simuLAtioN framework). MILAN is a collaborative project between the University of Southern California and the Vanderbilt University. MILAN provides a formal paradigm for specification of structural and behavioral aspects of embedded systems, an integrated model-based approach, and a unified software environment for system design and simulation. The rest of the paper is organized as follows. In section 2, MDE is discussed in the context of the modeling and simulation lifecycle, with an example. In section 4, a usage workflow for the proposed framework is presented. Finally, section 4 concludes the paper with a discussion of future research.

2 APPLYING A MODEL DRIVEN APPROACH TO MODELING AND SIMULATION

Model Driven Engineering (MDE) facilitates models, metamodels and model transformations. In MDE the central concept in the process of language development is the language metamodel. A metamodel is a definition of a language in the form of a model, i.e. a metamodel is itself a model (Kleppe, Warmer, and Bast 2003). In other words, the metamodel has also to be defined in a language and needs a metamodel. Thus, the meta-metamodel is introduced, allowing to specify metamodels. To avoid an infinite regress in the number of meta levels, meta-metamodels are often self reflexive and therefore the metamodel of the meta-metamodel is the meta-metamodel itself. Most approaches implementing MDE define a three level metamodeling stack for model, metamodel, and meta-metamodel. MDE fosters interoperability, adaptability and productivity because the definition of a metamodel allows for a common understanding of the elements described.

MDE is an open and integrative method and it is currently only described informally. The MDE approach addresses the steps required to take a model from conceptual design through to final implementation. The MDE principles may be implemented by different approaches like MDA (Model Driven Architecture) with MetaObjectFacility (MOF) (MDA 2001), MIC (Model Integrated Computing) with MetaGME (MIC 1997), Eclipse Modeling Framework (EMF) with ECore (EMF 2004) and many others. MOF, MetaGME and ECore serve a certain MDE implementation as meta-metamodel.

In this work, a model driven approach is applied to obtain the continuity necessary to bridge the gap between the conceptual modeling and simulation modeling processes. Firstly, a unified modeling and simulation lifecycle is suggested. In this lifecycle, a problem owner delineates a certain part of reality as a system of interest, then a modeler makes an abstraction of the system and prepares the conceptual model for the simulation study. The simulation modelers gather data, translate, and augment the conceptual models to obtain an executable simulation model in some appropriate formalism. Finally, simulation experts carry out experiments with the simulation model to analyze the system. Figure 2 illustrates the basic elements and outcomes of the lifecycle.

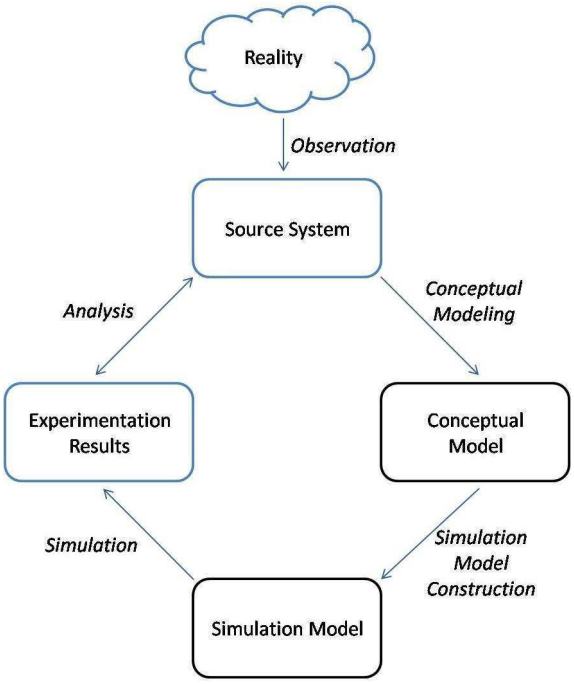

Figure 2: Basic elements and outcomes of the modeling and simulation lifecycle.

According to the suggested lifecycle, a **C**omponent Based **H**ierarchic**A**l **I**ntegrated Modeling and Simulatio**N** (CHAIN) framework for modeling and simulation is proposed. We divide simulation model construction into two layers, specification is represented separately from implementation. The framework has five layers and presents an integrated solution from problem definition to final evaluation. In each layer, different user roles can be defined such as problem owner, modeler, simulation specialist

and computer scientist. The primary processes of the framework is shown in Figure 3. Supplementary processes can be added like validation, verification, documentation and maintenance in different workflows.

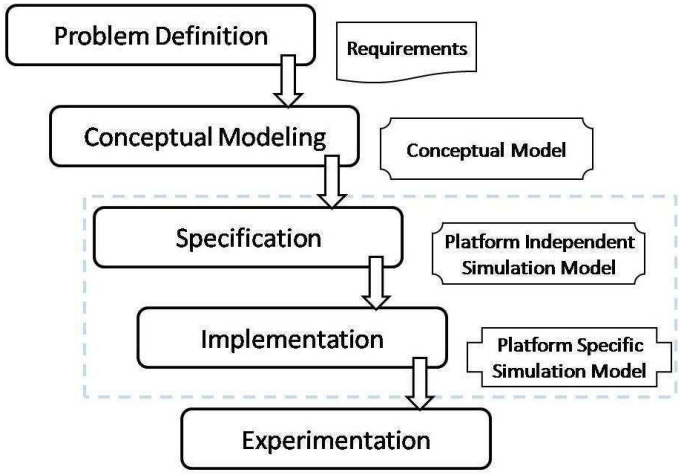

Figure 3: CHAIN: A framework for hierarchical modeling and simulation.

2.1 Problem Definition

At the Problem Definition layer, problem owner defines the purpose of the simulation study and his/her requirements. This essentially consists in setting the boundaries of the problem and choosing the value system (Key Performance Indicators) according to which the performance of the model will be assessed.

2.2 Conceptual Modeling

At the Conceptual Modeling layer, the modeler makes a high level abstraction of the real system according to a given worldview and prepares the Conceptual Model (CM). CM should be described using both textual and graphical presentations to support model transformations. CM serves as a bridge between problem owners and simulation specialists. Conceptual Modeling layer can be supported with modeling templates for different domains.

2.3 Specification

At the Specification layer, the simulation specialist defines a platform independent specification of the simulation model. Like the CM, the PISM (Platform Independent Simulation Model) could be described using either textual, graphical presentations, or both. The PSIM needs a formal specification such as DEVS, Petri Nets, State Charts, Partial Differential Equations, Bond Graphs, Finite State Automata, ...etc. It should allow modeling system functionality without taking into account any specific platform where the model could be later implemented.

2.4 Implementation

At the Implementation layer, the computer scientist develops the executable Platform Specific Simulation Model (PSSM). Specification and Implementation layers can be supported with component libraries where each component can be matched to its specific implementation for different platforms.

2.5 Experimentation

At Experimentation layer, the modeler and/or simulation specialist runs the executable model according to some scenarios and analyzes the experimentation results. Besides, validation experimentations can

be performed to validate if the input/output behavior of the simulation model matches to the purpose of the simulation study.

At the CM and PISM layers, well defined metamodels are needed. The Metamodeling levels of the CHAIN framework are presented in Figure 4. CMs are defined by a conceptual modeling language expressed with a Conceptual Model metamodel (CM metamodel). PISMs are defined as an instance of Simulation Model metamodel (SM metamodel). Both CM metamodel and SM metamodel are instances of a higher level meta-metamodel which is self reflexive, thus automated model transformations can be supported from CM to PISM and from PISM to PSSM. Specifications described in the PISMs are adapted to specific platforms by means of PSSMs from which the code is automatically generated.

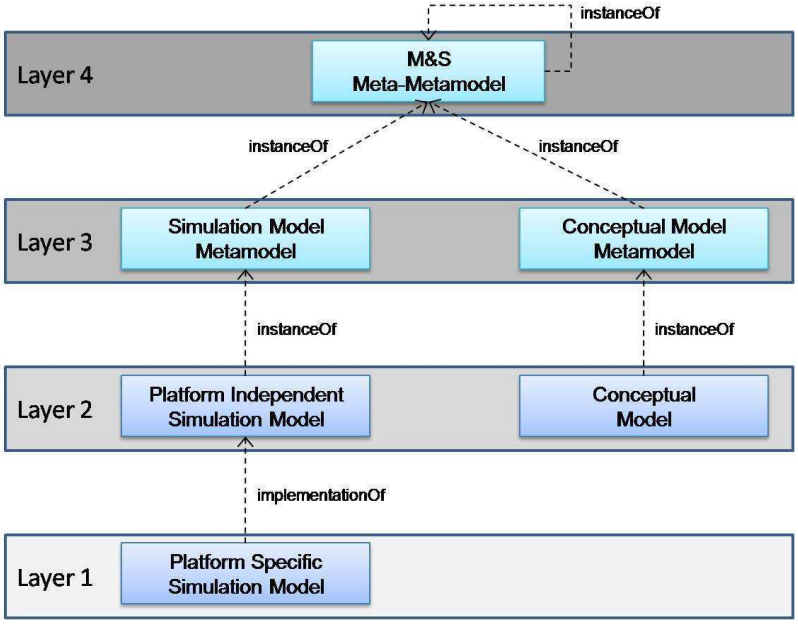

Figure 4: Metamodeling stacks of the framework.

The main problem in hierarchical modeling is assembly and integration in different components. A library based approach is recommended, where well documented libraries can be defined at each layer. These libraries contain pre-built, validated, parameterized and flexible modeling components. Figure 5 shows Conceptual Modeling, Specification and Implementation levels of the framework from the implementation point of view. A simple container terminal conceptual model is shown in the figure, since we work on container terminals as a test case in our research. A container terminal is a facility where cargo containers are transshipped between different transport vehicles, for further transportation. The transshipment may be between ships and land vehicles, for example trains or trucks, or it may be between only land vehicles. A model of a container terminal and its operating system can be used to predict performance indicators such as productivity, maximum capacity and profitability.

The conceptual modeling and specification layers are related through a mapping relation. It is to say that any element in the conceptual modeling library can be mapped to an element in the PISM library. For instance, based on previous efforts, one can link a given concept, say an AGV, with a specification that delivers the desired functionality. This is only possible because the Simulation Model Metamodel and the Conceptual Model Metamodel are derived from the same M&S Meta-Metamodel (See Figure 4). This feature offers continuity between the conceptualization and specification phases. A specification, or at least a part of it, can be constructed from a conceptual model. In the same way, specification and implementation are related. We call this link a matching relation. For each formal specification, a number of platform specific implementations can be constructed. Indeed, a DEVS specification can be implemented using various tools and platforms.

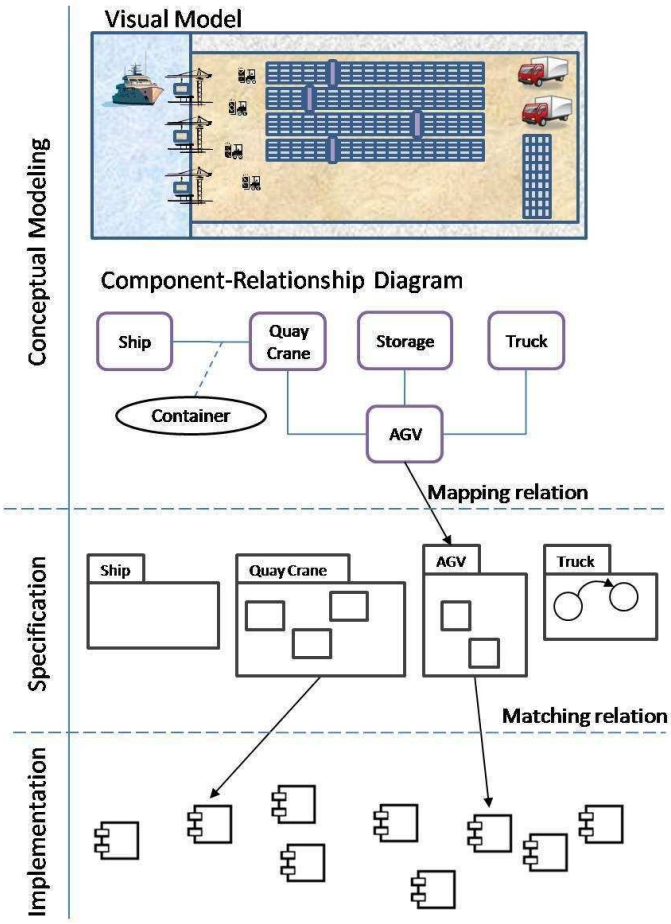

Figure 5: Component based modeling from the implementation point of view.

3 A SAMPLE WORKFLOW FOR THE PROPOSED FRAMEWORK

In this section we suggest a sample workflow for the proposed framework. In this workflow after defining the purpose of the simulation study, the modeler defines the conceptual model according to a worldview by using conceptual model templates and high level conceptual modeling libraries. During his/her modeling process, he/she searches for the existing components and selects the most appropriate ones if applicable. A mapping between the CM elements and the component specification is done. If the specification has implementation a matching between the component specification and implementation is done. The Simulation modeler defines and develops the missing components and adds them to the library. When the model is complete, simulation experts run the simulation and analyze the results. We call this workflow S^2Ma^2RT (Search, Select, Map, Match, Run and Test) and it is shown in Figure 6.

4 CONCLUSION AND FUTURE WORK

This work proposed a framework for component based modeling and simulation. Multi level abstraction and hierarchical modeling are in principle very powerful. We have shown some limitations in the conceptual modeling phase, in the model construction phase, and at their interface. As a solution, we propose CHAIN, an integrated model-driven framework. The framework is based metamodeling layers with well defined relations between them in order to guarantee an optimal transition between the steps of the process. The continuity thus obtained fosters stake-holder inclusion and model reuse. Finally, a usage workflow, S^2Ma^2RT is presented. The proposed framework has been tested on simple cases

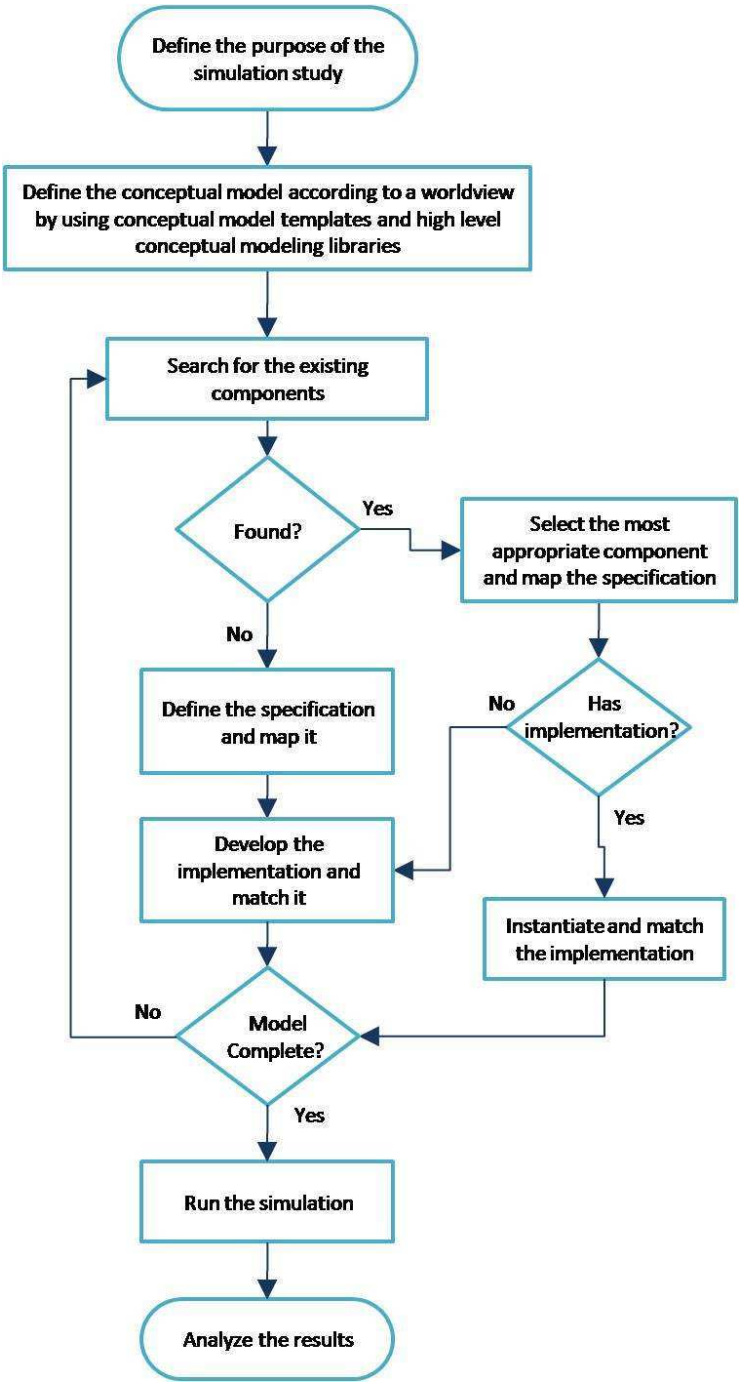

Figure 6: A sample workflow for the proposed framework.

(Cetinkaya, Verbraeck, and Seck 2010) and will be further tested on larger cases, including namely the simulation based design of a container terminal. Future research will include the definition of a domain specific conceptual modeling metamodel for simulation.

ACKNOWLEDGMENTS

We would like to thank Dr.ir. Job Honig from the Systems Engineering Group of the Faculty of Technology, Policy and Management at Delft University of Technology for his valuable comments.

REFERENCES

Bakshi, A., V. K. Prasanna, and A. Ledeczi. 2001. Milan: A model based integrated simulation framework for design of embedded systems. In *Workshop on Languages, Compilers, and Tools for Embedded Systems (LCTES)*, 82–93. Snowbird, UT.

Cetinkaya, D., A. Verbraeck, and M. D. Seck. 2010. A metamodel and a devs implementation for component based hierarchical simulation modeling. In *ANSS '10: Proceedings of the 2010 Spring Simulation Conference*: SCS.

EMF 2004. Eclipse modeling framework. Available via <http://www.eclipse.org/modeling/emf> [accessed March 26, 2010]. The Eclipse Foundation.

Kleppe, A., S. Warmer, and W. Bast. 2003. *Mda explained - the model driven architecture: Practice and promise*. Addison-Wesley, Boston.

MDA 2001. Model driven architecture. Available via <http://www.omg.org/mda> [accessed March 26, 2010]. Object Management Group (OMG).

MIC 1997. Model-integrated computing. Available via <http://www.isis.vanderbilt.edu> [accessed March 26, 2010]. Institute for Software Integrated Systems, Vanderbilt University.

Pidd, M. 2002. Reusing simulation components: simulation software and model reuse: a polemic. In *Proceedings of the 34th conference on Winter Simulation*, ed. J. L. S. E. Yücesan, C.-H. Chen and J. M. Charnes, 772–775: Institute of Electrical and Electronics Engineers, Inc.

Robinson, S. 2006. Conceptual modeling for simulation: issues and research requirements. In *WSC '06: Proceedings of the 38th conference on Winter Simulation*, ed. J. L. B. G. L. D. M. N. L. F. Perrone, F. P. Wieland and R. M. Fujimoto, 792–800: Institute of Electrical and Electronics Engineers, Inc.

Simon, H. A. 1962. The architecture of complexity. In *Proceedings of the American Philosophical Society*, 467–482.

AUTHOR BIOGRAPHIES

DENIZ CETINKAYA is a Ph.D. student at Delft University of Technology. She is in the Systems Engineering Group of the Faculty of Technology, Policy and Management. She received her M.Sc. in Computer Engineering from the Middle East Technical University, Turkey in 2005. She received her B.Sc. with honors in Computer Engineering from the Hacettepe University, Turkey in 2002. Her research focuses on component based modeling and simulation. Her e-mail address is <d.cetinkaya@tudelft.nl>.

ALEXANDER VERBRAECK is a full professor in Systems and Simulation in the Systems Engineering Group of the Faculty of Technology, Policy and Management of Delft University of Technology, and a part-time full professor in supply chain management at the R.H. Smith School of Business of the University of Maryland. He is a specialist in discrete event simulation for real-time control of complex transportation systems and for modeling business systems. His e-mail address is <a.verbraeck@tudelft.nl>.

MAMADOU D. SECK is an assistant professor in the Systems Engineering Group of the Faculty of Technology, Policy, and Management of Delft University of Technology. He received his Ph.D. degree from the Paul Cezanne University of Marseille and his M.Sc. and M.Eng. degrees from Polytech Marseille, France. His research interests include modeling and simulation formalisms, dynamic data driven simulation, human behavior representation and social simulation, and agent directed simulation. His e-mail address is <m.d.seck@tudelft.nl>.