TOWARDS A COMPONENT BASED CONCEPTUAL MODELING LANGUAGE FOR DISCRETE EVENT SIMULATION

Deniz Cetinkaya
Alexander Verbraeck
Mamadou Seck
Systems Engineering Group, Faculty of Technology, Policy and Management
Delft University of Technology
Jaffalaan, 5, 2628BX, Delft, THE NETHERLANDS
email: d.cetinkaya@tudelft.nl, a.verbraeck@tudelft.nl, m.d.seck@tudelft.nl

KEYWORDS

Conceptual modeling, conceptual modeling language, discrete event simulation, hierarchical modeling

ABSTRACT

Recent studies state the importance of conceptual modeling in simulation life cycles. Proper development of a conceptual model is critical for expressing the context, elements, relationships, limitations and purpose of the simulation study. Surprisingly there are many simulation projects that have no explicit conceptual model, a poorly or only partially developed conceptual model, or incomplete documentation of the simulation conceptual model. The reason for the deficiency in conceptual modeling stage is that there does not exist a well defined simulation conceptual modeling method. In this paper, a brief overview of the conceptual modeling techniques used in simulation field is provided and the need for a unified simulation conceptual modeling method is stated. Then, a conceptual modeling approach for discrete event simulation is proposed and compared to other modeling techniques.

INTRODUCTION

In general terms, each simulation study has a problem definition, conceptualization (conceptual modeling), model building (simulation model construction), and experimentation stages. Conceptual modeling is probably the most difficult aspect of a simulation study and recent studies state the importance of conceptual modeling in simulation life cycles (Pace 2000, Yilmaz and Oren 2006, Robinson 2006; 2008). During the simulation conceptual modeling stage, a modeler makes an abstraction of the system and prepares the conceptual model for the simulation study. A simulation conceptual model is a simplified representation of the real system without reference to the implementation details. It generally describes the elements, relationships, boundaries and objectives of a simulation study.

Conceptual modeling not only requires that the modeler develop an appropriate model, but that all parties involved in a simulation study understand and agree to that model. As such, it is important that the conceptual model is represented and communicated in a manner that is understandable to all. A range of modeling methods have been used for representing simulation conceptual models, such as process flow diagrams, event graphs, activity diagrams, IDEF diagrams, Petri nets, etc. (Robinson 2006). Many of the techniques present an abstract way of thinking which is not natural and so it is difficult to properly model the real system in the required level of detail. For example, a flow diagram provide an overview of the system and do not have much detail. Petri nets are well defined and they represent a directed graph of nodes and arcs. However, there is not an elegant way of representing hierarchies graphically.

Moreover, conceptual models are often not reused explicitly in the further steps of the simulation process, as formal model transformation methods are not available to guarantee model continuity (Olive 2007). 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.

Therefore, we can conclude that there is a big semantic gap between the conceptual modeling stage and the simulation model construction stage. Therefore, we would like to pay attention to the deficiency in conceptual modeling stage and the lack of a commonly accepted standardized conceptual modeling method and language in Modeling and Simulation (M&S). In short, the existing modeling methodologies require some development in the state of the art of conceptual modeling and simulation model construction stages.

In this paper, firstly a brief overview of the conceptual modeling techniques used in simulation field is provided. Then, two useful modeling approaches, namely hierarchical modeling and component based modeling are discussed. After that, a conceptual modeling approach for discrete event simulation is proposed. Finally, conclusions are drawn and future work is outlined.

CONCEPTUAL MODELING METHODS USED IN M&S

Simulation conceptual modeling generally benefits from general purpose diagramming techniques, which are not adequate for meeting the needs of simulation projects. Despite the fact that conceptual modeling is an important step in a simulation study, there is not a common simulation conceptual modeling language. Thus, in many cases conceptualization deeply depends on the skill and experience of individual modelers. This section provides a brief overview of the conceptual modeling methods used in M&S.

In order to provide a better understanding, after giving a brief introduction we will give a sample model of a single server queue for each method. Simulation of a single server queuing system is a common example of discrete event simulation such as an information desk at an airport or a hotel, a pharmacy, a barber shop, or a ticket office. This example was chosen because of its simplicity enables an easier comparison of the methods.

For example, consider a service facility with a single server for which we would like to estimate the average delay in the queue for arriving customers. We define the following state variables: status of the server (idle or busy), number of customers waiting to be served (if any), the arrival time of each customer waiting in the queue. We define three types of events: arrival, service and departure. Delay in the queue means the length of time from the arrival of a customer at the information desk queue until the instant he/she begins to be served.

Event Graphs

Event graphs provide a representation for discrete event simulation (Schruben 1983). An event graph partitions the model into events and relationships between events. The events are represented by vertices (nodes) in the graph and relationships between events are represented as directed edges (arcs) between event vertices. Figure fig:event$_graphshowsaneventgraphforthesampleproblem.$

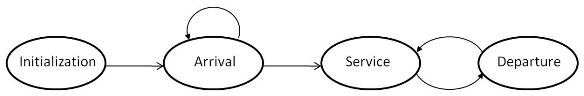


Figure 1: An event graph for a single server queue (Seila et al. 2003)

Activity Cycle Diagrams

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. UML activity diagrams can be used to describe step-by-step workflows of components in a system. An activity diagram mostly consists of, activities (rounded rectangles), decisions(diamonds) and flows(arrows). Bars represent the start (split) or end (join) of concurrent activities. A black circle represents the start (initial state) of the workflow and an encircled black circle represents the end (final state). Flows(arrows) run from the start towards the end and represent the order in which activities happen. Activity diagrams can be regarded as a form of flowchart. Figure 2 shows an activity cycle diagram for the sample problem.

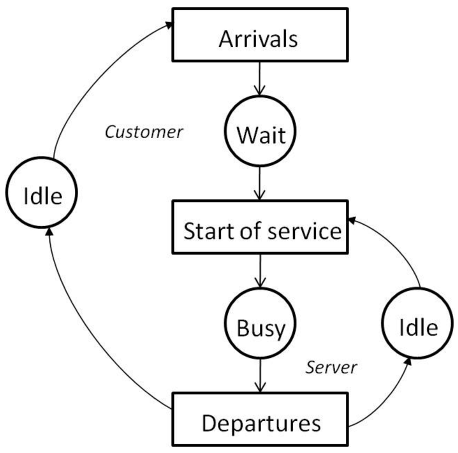


Figure 2: An activity diagram for a single server queue (Seila et al. 2003)

IDEF Diagrams

IDEF (Integration DEFinition) is a family of modeling languages in the field of systems and software engineering. They cover a wide range of modeling methods, yet the most-widely recognized and used one is IDEF0. IDEF0 (Integration Definition for Function Modeling) is a function modeling methodology for describing organizations or systems. It is a model that consists functions, data and objects. Functions are represented by boxes. Data or objects that interrelate those functions are represented by arrows). Figure 3 shows a simple IDEF0 diagram for the sample problem.

Petri Nets

Petri nets are bipartite graphs and provide a mathematically rigorous modeling framework. They serve as a ready simulation model, as well as a conceptual model.

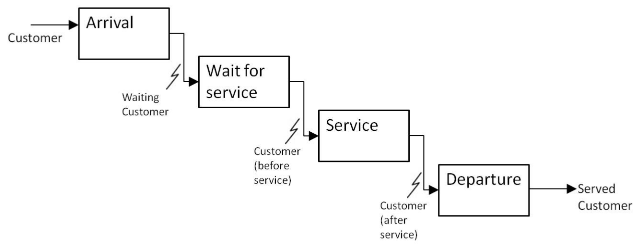


Figure 3: An IDEF0 diagram for a single server queue

However, their analysis is intractable for large models. Petri nets consist of places, transitions, and directed arcs. Arcs run from a place to a transition or a transition to a place, never between places or between transitions.The places from which an arc runs to a transition are called the input places of the transition; the places to which arcs run from a transition are called the output places of the transition. Places may contain a number of tokens. A transition of a Petri net model is fired whenever there is a token at the start of all its input arcs. Figure 4 shows a petri net model for the sample problem.

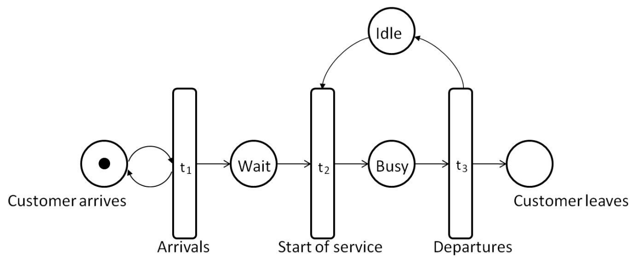


Figure 4: A petri net model for a single server queue

There are various types of Petri nets, such as timed Petri nets, stochastic Petri nets, and colored Petri nets. The use of stochastic Petri nets has become particularly important in the modeling of discrete event systems. Timed Petri nets are the particular types of Petri nets that associate the time and time delays.

HIERARCHICAL COMPONENT BASED MODELING

As modelers build more complex and complicated models for large systems, it becomes hard to design, develop, manage and maintain the simulation models. The monolithic approach for developing models becomes too cumbersome in large simulation projects. Besides, when each simulation model is designed from scratch, the lack of reuse makes simulation a time consuming and expensive task (Oses et al. 2004).

Applying different software engineering approaches into the simulation field can help managing larger models, such as thinking at various levels of abstraction or component based development. In this section two modeling approaches, which have been applied in the simulation field and provided valuable contributions, are discussed. These are hierarchical modeling and component based modeling.

Hierarchical modeling (also known as multi-level modeling) provides a way to represent a system in a hierarchical structure to deal with large scale or complex models in a thorough manner (Simon 1962). Hierarchical modeling allows modeling with more manageable sub-parts at different levels of detail. The ability to move among the different levels of a model hierarchy greatly increases the manageability and understandability of large models (Daum and Sargent 1999). Hierarchical modeling can provide for a more natural way of modeling and help to focus on different degrees of detail when using a model.

Hierarchical models are generally developed in two different ways, that are top-down and bottom-up strategies. In both cases, hierarchical models mostly represent a tree-like structure. In the top-down approach, a system is broken down into subsystems and this is called as decomposition. During the top-down modeling process, modelers specify the main parts and relationships of the system without inner details first and then they fill in the lower levels. In the bottom-up approach, subsystems are coupled together to form a larger system and this is called as composition. During the bottom-up modeling process, modelers first think of the lowest level, i.e. smallest parts or building blocks of the system and then they use these previously constructed building blocks to compose larger models and systems. Simulation models can be developed by employing either a top-down decomposition approach or a bottom-up composition approach.

In the component based approach software systems are built by assembling components already developed and prepared for integration. Component based modeling and simulation is an interesting research area that many researchers studied in the last decade (Buss 2000, Himmelspach and Uhrmacher 2004, Sarjoughian and Elamvazhuthi 2009, Verbraeck and Valentin 2008). Component based simulation relies on having pre-built, validated simulation model components that can be coupled to form a composed model that represents a system. A simulation model component is expected to be a self-contained, interoperable, reusable and replaceable unit, providing useful services or functionality to its environment through properly defined interfaces (Verbraeck and Dahanayake 2002). Component based approach promises to have many benefits over a monolithic approach such as reuse of interoperable components and rapid development (Verbraeck and Valentin 2008).

The development process for component based systems consists of two major stages: component development and component composition (Oses et al. 2004). These stages are usually carried out by different parties, like domain experts and software engineers. When a component library is available, a developer can build a system in a bottom-up fashion, by combining components into larger components, where an assembly of the highest level components is considered to be the system. In component based approaches, overall software quality increases due to components are thoroughly tested first and reviewed during reuse (Sommerville 2007).

The component based approach has originally a bottom-up way of assembling components, which means that it can be applied together with the hierarchical modeling approach. Simulation model components can be assembled in many ways into a hierarchy. New components can be built from scratch in each layer or reused if they already exist in pre-defined and verified component libraries, so it is not necessary to always create larger components from smaller components.

Applying a unified hierarchical component based modeling approach looks like an encouraging way in the simulation field. During the the conceptual modeling stage, by applying a top-down hierarchical modeling approach, a modeler can first partition the system into the relevant subsystems and define the relationships between them, 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. At the simulation model construction stage, by applying a bottom-up component based approach, basic available primitives and building blocks can be composed to provide the desired functionality of the identified subsystems and the simulation model.

However, there is a big semantic gap between the conceptual modeling and the simulation model construction stages. For example, to be able to reuse the existing components, one should know that what is already available. This means that, there must be a way to express how the components relate to the subtrees in the conceptual model. Besides, good classification and documentation is essential for the successful reuse of simulation model components. We believe that in order to bridge this gap, we need a common simulation conceptual modeling language and a model transformation method between the conceptual model and the simulation model. A higher level representation on top of the rigid simulation model implementation is expected to make the simulation model development process faster.

A CONCEPTUAL MODELING APPROACH FOR DISCRETE EVENT SIMULATION

In this section, a hierarchical component based conceptual modeling approach is suggested. The proposed conceptual modeling method will basically define a system with its components, relations, and objectives based on the following definition of a system. A system is a set of interrelated components working together toward some common objective or purpose (Kossiakoff and Sweet 2003, Blanchard and Fabrycky 2006).

We define two types of nodes, that are components and entities. Each component can have four different types of variables: input variables, output variables, local variables and parameters. Components can have various properties such as descriptors and rules. Descriptors define the meta information such as name, version, author, bugs, keywords, etc. They can be used in cataloging and searching components. Rules are constraints that can be defined about components. They can be used to express the boundaries of the system. Besides, we define a component type for each component which is used for classification purposes. We only allow type inheritance in our method and use component type information to classify the components. Type inheritance only provides a limited support for component structure.

Every component has an objective, which is defined by its behavior. At the conceptual modeling level, we provide a way to define the pseudo algorithm for the behavior of a component. This will be used to support model transformation and not obligatory.

Entities are specialized components, having both variables and properties. The only difference between a component and an entity is that entities do not have an internally defined behavior. For example, entity components can be used to represent system resources. A graphical representation for components and entities is shown in Figure 5.

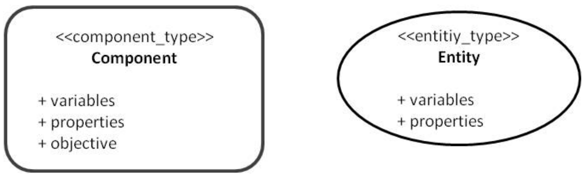


Figure 5: Visual representation for components and entities

In order to define the state of the system, we use the definition of Law and Kelton (1991): the state of a system is the collection of variables necessary to describe a system at a particular time. Hence, we use the local variables to refer to the state and state change is possible when the local variables are updated. Output of a component is available when the output variables are updated.

Relations define how components and entities relate to each other. Six basic relations are suggested in our method and a textual representation for them is listed in Table 1.

Due to hierarchical modeling is applied, composition and decomposition capability is especially handled. Besides, composition and aggregation are differentiated clearly. The hierarchies are represented with 'HAS A'

Relation
Fixed Composition CompA **HAS** CompB
Temporary Composition CompA **GOES** CompB
Logical link CompA **ISLINKED** CompB
Physical link CompA **ISJOINED** CompB
'Send-To' relation CompA **SENDS** EntityC **TO** CompB
'Send-To-Via' relation CompA **SENDS** EntityC **TO** CompB **VIA** CompD

Table 1: Textual representation of basic relations

relation and called as fixed composition. Since a unified approach is performed, composition refers to both composition and decomposition capability. Aggregation refers to a temporary whole-part relationship during the execution of the simulation model. This type of relation is called as temporary composition and represented as with 'GOES TO' relation.
Logical links and physical links are distinguished as well. Association relations or any other logical relationships can be expressed with logical links. 'Send-To' and 'Send-To-Via' relations are provided for transferring data between components. When necessary and appropriate cardinality information can be defined for the relations, such as: 1..*, * or 0..*, n, 0..1, 1, ...etc. A graphical representation for the suggested relations is illustrated in Table 2.
In order to define the objective of the components we suggest four main behavioral modeling primitives, which are if condition, while loop, switch case and assignment. Then we define an expression as a combination of these primitives. A possible textual notation is given below:

- $IF < condition > THEN < expression > ELSE < expression >$
- $SWITCH\{CASE < case >< expression >\}$

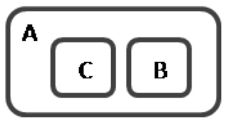
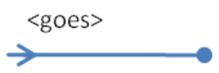
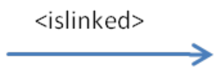
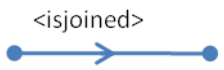
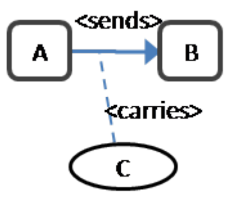
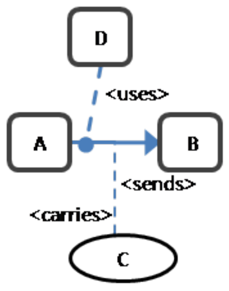
Relation	**Representation**
Fixed Composition	
Temporary Composition	
Logical link	
Physical link	
'Send-To' relation	
'Send-To-Via' relation	 D <uses> A B <sends> <carries> C

Table 2: Basic relations of the proposed conceptual modeling method

- $WHILE < condition > DO < expression >$
- $Assignment :< variable >= value$
- $Expression : \{IF\_Cond, While\_Loop, Switch, Assignment\}$

A sample component diagram of a single server queue is demonstrated in Figure 6. The model defines the following steps:

- *Customer* arrives(GOES) to the *Waiting Queue* of the *Service Desk*
- *Waiting Queue* ISJOINED to the *Service Process*
- *Waiting Queue* is a queue component, thus when available *Customer* is sent to *Service Process*
- When *Service Process* is finished, *Customer* leaves
- *Waiting Queue* calculates the delay time for each *Customer*

- *Service Process* calculates the service time for each *Customer*

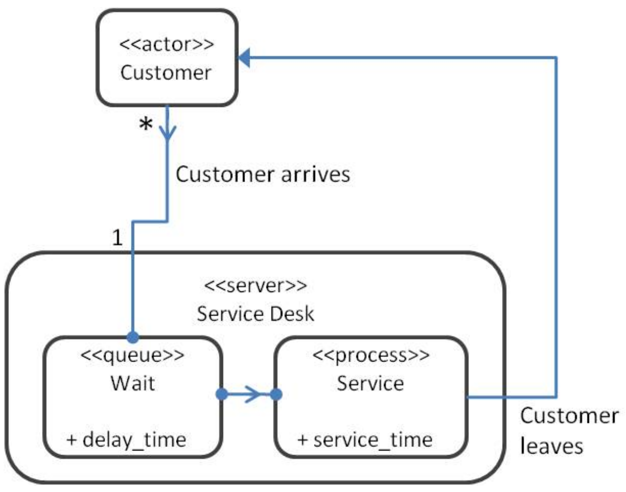


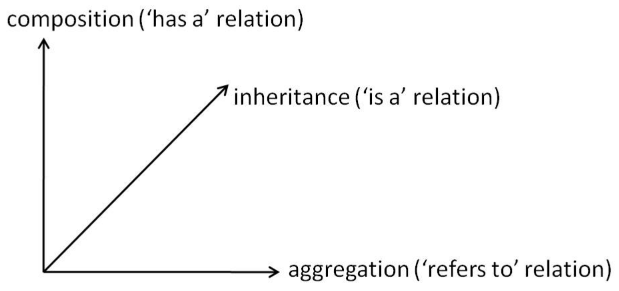
Figure 6: A sample conceptual model with the proposed method

COMPARISON WITH THE EXISTING TECHNIQUES

In order to compare the proposed method in its current status with the other techniques, a number of requirements for an effective conceptual modeling language is represented below:

1. It should represent the system structure (elements and relations) clearly.
2. It should represent the purpose of the simulation study (objectives and boundaries).
3. It should be abstract from technical or organizational details (Ribbert et al. 2004).
4. It should support classification and inheritance.
5. It should support hierarchical modeling to develop manageable and understandable models.
6. It should be formal enough to avoid misinterpretations. Besides, it should be theoretically possible to map the conceptual model to a formal specification to support model transformations (Ribbert et al. 2004).
7. It should be easy to learn and use (Ribbert et al. 2004).

Most of the conceptual modeling languages and modeling techniques provide the first four requirements. However, the main problem in simulation conceptual modeling is hierarchical modeling and model composability (Kasputis and Ng 2000). Recent studies state that model composability is troublesome in simulation and the existing methodologies require additional effort to facilitate it (Röhl and Uhrmacher 2006, Yilmaz and Oren 2006). Indeed, combining multi-level abstraction and composition with inheritance and aggregation is not easy, neither in theory nor in practice. Figure 7 shows the three dimensions of hierarchical simulation conceptual modeling. Object oriented modeling methods provide few mechanisms to describe components. Simply adopting the object oriented concepts is not adequate for expressing the hierarchies in simulation models. Thus, when a modeler wants to add different layers into his/her models, object oriented conceptual modeling techniques become insufficient.



composition (‘has a’ relation)
inheritance (‘is a’ relation)
aggregation (‘refers to’ relation)

Figure 7: Three dimensions in hierarchical simulation conceptual modeling

Although many modeling techniques have well defined syntax or semantics, a clear metamodel and a rigorous formalism are lacking in many cases. Besides, they do not provide formal model transformation methods to guarantee model continuity. Only Petri net models serve as a ready simulation model. However, they are far from being practical and easy to use. In many cases, simulation modelers need to be experienced and trained.
The proposed method represents the system structure with components and relations. Components have objectives and rules that define the purpose of the simulation study. Hence, it satisfies requirement 1 and 2. The method satisfies requirement 3 partially, since it allows defining pseudo algorithms for objectives. It satisfies requirement 4 partially as well, due to it only allows type inheritance. Hierarchical modeling is supported via the component structure and we claim that the method is easy to use. Requirement 6 is a future work at the moment, a metamodel and a model transformation method will be defined for the proposed method. After that, a more detailed comparison will be performed.

CONCLUSION AND FUTURE WORK

Although, an effective and consistent conceptual model is critical for expressing the purpose of the simulation study, many simulation projects have no deliberate conceptual modeling stage. Moreover, formal model transformation methods are not available to help the simu-

lation model developers while moving from the conceptual model to simulation model. As a result, simulation models generally do not have a higher level representation on top of the rigid simulation model implementation and so they are not understandable to others. We think that the deficiency in simulation conceptual modeling is caused by the lack of a well-defined conceptual modeling method and language in M&S. However, this subject has not been adequately studied yet in simulation field. This work aims at improving the conceptual modeling stage and increasing the reuse of simulation model components in modeling and simulation. We suggest a conceptual modeling approach for discrete event simulation and lead to new insights about conceptual modeling. Reuse of simulation model components will help the modelers to construct their simulation models faster, better and more reliable. Additionally, a common conceptual modeling method will provide a better understanding for conceptual models.
As a future work, we will define the suggested simulation conceptual modeling method formally and propose a metamodel for the simulation conceptual modeling language. After that, a unified modeling and simulation methodology that ensures model continuity will be proposed by the use of the gained insights about conceptual modeling.

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>`.

REFERENCES

Blanchard B.S. and Fabrycky W.J., 2006. *Systems engineering and analysis.* Pearson Prentice Hall, NJ, 4th ed.

Buss A., 2000. *Component-based simulation modeling.* In J. Joines; R. Barton; K. Kang; and P. Fishwick (Eds.), *Proceedings of the 32nd Winter Simulation Conference (WSC '00).* IEEE Computer Society, San Diego, CA, USA. ISBN 0-7803-6582-8, 964–971.

Daum T. and Sargent R.G., 1999. *Scaling, hierarchical modeling, and reuse in an object-oriented modeling and simulation system.* In *Proceedings of the 31st Winter Simulation Conference (WSC '99).* ACM, Phoenix, Arizona, USA. ISBN 0-7803-5780-9, 1470–1477. doi:http://doi.acm.org/10.1145/324898.325304.

Himmelspach J. and Uhrmacher A., 2004. *A component-based simulation layer for JAMES.* In *Proceedings of the 18th Workshop on Parallel and Distributed Simulation (PADS '04).* IEEE, Piscataway NJ, 115–122.

Kasputis S. and Ng H.C., 2000. *Composable simulations.* In J.A. Joines; R.R. Barton; K. Kang; and P.A. Fishwick (Eds.), *Proceedings of the 32nd Winter Simulation Conference (WSC '00).*

Kossiakoff A. and Sweet W.N., 2003. *Systems enginnering: principles and practice.* Wiley Series.

Law A.M. and Kelton W.D., 1991. *Simulation modeling and analysis.* McGraw-Hili, Inc., 2nd ed.

Olive A., 2007. *Conceptual modeling of information systems.* Springer.

Oses N.; Pidd M.; and Brooks R.J., 2004. *Critical issues in the development of component-based discrete simulation. Simulation Modelling Practice and Theory*, Volume 12, no. 7-8, 495–514.

Pace D.K., 2000. *Ideas about simulation conceptual model development. Johns Hopkins APL Technical Digest*, Volume 21, no. 3, 327–336.

Ribbert M.; Niehaves B.; Dreiling A.; and Holten R., 2004. *An Epistemological foundation of Conceptual Modeling.* In *In Proceedings of the 12th European Conference on Information Systems.* Turku, Finland, 1557–1568.

Robinson S., 2006. *Conceptual modeling for simulation: issues and research requirements.* In L.F. Perrone; B. Lawson; J. Liu; and F.P. Wieland (Eds.), *Proceedings of the 38th Winter Simulation Conference (WSC '06).* WSC, Monterey, California, USA. ISBN 1-4244-0501-7, 792–800.

Robinson S., 2008. *Conceptual modelling for simulation Part I: definition and requirements. Journal of the Operational Research Society*, 59, 278–290.

Röhl M. and Uhrmacher A.M., 2006. *Composing simulations from XML-specified model components.* In L.F. Perrone; F.P. Wieland; J. Liu; B.G. Lawson; D.M. Nicol; and R.M. Fujimoto (Eds.), *Proceedings of the 38th Winter Simulation Conference (WSC '06).* IEEE, 1083–1090.

Sarjoughian H.S. and Elamvazhuthi V., 2009. *CoSMoS: a visual environment for component-based modeling, experimental design, and simulation.* In O. Dalle; G.A. Wainer; L.F. Perrone; and G. Stea (Eds.), *Proceedings of the 2nd International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools '09).* ICST, Rome, Italy.

Schruben L., 1983. *Simulation modeling with event graphs. Communications of the ACM*, 26, no. 11, 957–963.

Seila A.; Ceric V.; and Tadikamalla P., 2003. *Applied Simulation Modeling.* Brooks/Cole Publishing, Thomson Learning Inc.

Simon H.A., 1962. *The architecture of complexity.* In *Proceedings of the American Philosophical Society.* 467482.

Sommerville I., 2007. *Software Engineering.* Addison-Wesley, 8th ed.

Verbraeck A. and Dahanayake A.N.W., 2002. *Building blocks for effective telematics application development and evaluation.* Delft University of Technology.

Verbraeck A. and Valentin E., 2008. *Design guidelines for simulation building blocks.* In S.J. Mason; R.R. Hill; L. Mönch; O. Rose; T. Jefferson; and J.W. Fowler (Eds.), *Proceedings of the 40th Winter Simulation Conference (WSC '08).* WSC, InterContinental Hotel, Miami, Florida, USA. ISBN 978-1-4244-2708-6, 923–932.

Yilmaz L. and Oren T.I., 2006. *Prospective issues in simulation model composability: basic concepts to advance theory, methodology, and technology. The MSIAC's M&S Journal Online*, Volume 2, 1–7.