A Description Structure for Simulation Model Components

Maritta Heisel
University Münster
Germany
heisel@uni-muenster.de

Johannes Lüthi
University of Applied Sciences FHS KufsteinTirol
Austria
johannes.luethi@fh-kufstein.ac.at

Adelinde Uhrmacher
University Rostock
Germany
lin@informatik.uni-rostock.de

Edwin Valentin
Delft University of Technology
The Netherlands
edwinv@tbm.tudelft.nl

Keywords: Model components; Semantics; Indexing description; Discrete and Continuous formalism

INTRODUCTION

Simulation model components become more and more commonly used in the development of simulation models [Diamond et al, 2002]. The benefits of reuse, easier model development, and the hiding of detailed and complex code of a simulation environment make it possible that simulation studies can be performed faster, that novices can develop simulation models and that problem owners get more support, because more experiments can be performed [Pater and Teunisse, 1997; Davis et al, 2000].

When more model developers become aware of these benefits and more simulation model components become available, they will try to search for simulation model components in repositories and databases before they undertake the complex and time-consuming development of these simulation model components themselves [Bruneton et al, 2002]. So far, simulation model developers have been reluctant to use simulation model components developed by others [Page and Opper, 1999; Valentin and Verbraeck, 2002]. One main reason for this fact is known as the "not-invented-here" syndrome. Simulation model components developed by someone else are not supposed to be correct and to constitute a valid representation of the system of the model developer. Therefore, model developers do not dare to use simulation model components developed by an unknown source.

The not-invented-here syndrome does not always apply. Examples exist where model developers successfully use externally developed simulation model components. When the model developer knows and trusts the developer of the simulation model component and receives a detailed description of the simulation model component, then model developers dare to do some tests to evaluate the usability of the simulation model component for their problem system. Examples of usable simulation model components are provided by commercial simulation environment vendors who have developed extensions to their software for specific domains, e.g. *Contact Center*, a discrete event resource representation of call centers by Rockwell Software for Arena (www.arenasimulations.com) and *Radar and Communications*, continuous by Mathworks for Simulink and Matlab (www.mathworks.com). These commercial parties provide complete sets of simulation model components, combined with extensive documentation and training material. This material and the status of the developers gave model developers the trust to use the simulation model components.

We are convinced that a good description of a simulation model component will support model developers to gain trust in simulation model components and help them to make a decision to use the component. Most description structures of components only explain the interfaces with other components, but in simulation models the reduction of the system is important. As a result model developers require understanding of the semantics of the interfaces and the functionalities of a simulation model component. Therefore, we need description structures for simulation model components that put the stress on the semantics of the component.

As a simulation model component does not describe a system per se but a system given a certain objective, we believe, as Overstreet et al [2002], that a key part of a model component description must involve capturing the objectives, assumptions and constraints under which the original models were developed in an explicit manner. Due to the different objectives, different model components can describe the same system, and one model might be a valid representation of the system with respect to one objective but not with respect to another objective. Multi-facetted modeling emphasizes the importance of developing families of models and recognizes the existence of multiplicities of objectives and models as a fact of life [Zeigler, 1984]. To select one of the models for reuse requires understanding

their meaning, part of which refers to their objectives, constraints, and underlying assumptions. These facts are difficult to capture as they are often only implicitly included, and modelers might not even be aware of many of the assumptions and constraints that apply.

In this article we propose a structure enunciating how to describe the characteristics of a simulation model component focusing on semantics of interfaces and functionalities. In detail we discuss a minimum set of items that we expect model developers to use to gain trust in the simulation model component. Moreover, the minimum description should enable model developers to have sufficient insight to start using the simulation model component to compose their simulation models.

Before we introduce the description structure and minimum items of a simulation model component, we will first explain what simulation model components are. We then discuss approaches for describing software components that were developed in the software engineering community. Based on these descriptions, we introduce our structure for describing a simulation model component. We illustrate the applicability of this structure by examples of simulation model components of an agent representing local authority in a city and a retailer of a supply chain. The paper concludes with a summary and directions for future research.

BACKGROUND ON SIMULATION MODEL COMPONENTS

Simulation models describe different types of systems: continuous, discrete and hybrid systems. They use different formalisms, e.g. block diagrams, hybrid automata, hybrid petrinets, timed petrinets, state charts, or DEVS, implemented in different simulation environments with different features. These simulation models have the disadvantages that they are hard to adjust, that it is time-consuming to develop them and that it is often very hard to reuse parts of earlier developed simulation models [Robinson, 1999; Keller et al. 1991]. Therefore currently a lot of attention is paid to the ability to use component technology as developed within electronics and software engineering. The benefit of these components should be that simulation models can be assembled from existing parts and that developing a simulation model requires only some parameterization. We call these parts “simulation model components”. A simulation model component is a component that is meant to be used as part of a simulation model representing a sub-system with a predefined reduction and generalization.

Simulation model components allow a model developer to compose a simulation model, ideally without any additional programming. The simulation model components are developed by a component developer who performs the simulation software coding and ensures that simulation model components can be plugged together to form a simulation model of a system in a specific domain, such as the call center (www.arenasimulations.com) and the radar systems (www.mathworks.com).

A simulation model aims at a valid, but reduced, representation of a system. Currently, model developers mainly use their own developed simulation model components, because they are not aware of or interested in model components developed by someone else. They fear the risk of a different level of reduction and a different way of referring to functionalities in a system.

Simulation model developers work with different simulation paradigms and therefore use simulation model components in different ways. In agent models, a component is seen as an agent that can be reused in another simulation model; in discrete event for resource allocation simulation models, a simulation model component is part of a set of components that enable simulation of specialized domains; and in continuous simulation, the simulation model components are a set of differential equations that are always operating together. Even though model developers use simulation model components differently, still they need comparable insight in the usability of a simulation model component as part of their simulation model.

A standardized description of a simulation model component should enable a model developer to find existing simulation model components and to judge whether a given simulation model component is suitable for their simulation model. Currently, developers of simulation model components either do not describe their components or only use an unstructured textual way of describing them. As a result, model developers have difficulties to identify the existence of usable simulation model components and to judge their applicability. This is one of the main reasons why model developers decide not to use simulation model components and do not benefit from the advantages that simulation model components could offer.

DESCRIPTIONS OF COMPONENTS IN SOFTWARE ENGINEERING

In the area of software engineering, several component technologies are now being used. The best-known of these are Java Beans (java.sun.com/products/javabeans/) and Enterprise Java Beans (java.sun.com/products/ejb/) of SUN Microsystems, Component Object Model (COM, COM+, DCOM, .NET - www.microsoft.com/com/) of Microsoft, and the Corba Component Model of the Object Management Group (www.corba.com). For these existing technologies, no specification standards exist yet.

However, the software engineering community faces the same problems as the simulation community: the

advantages of components can be exploited only if component markets can be set up, where components can effectively be searched for, and whose implementation may be kept secret from component consumers. Therefore, proposals how to specify software components describing their semantics have been developed. We have analyzed three of these proposals to judge their applicability to describe simulation model components. The analyzed proposals are the Web Service Description Language (WSDL) (www.daml.org/services/), a specification standard for business components developed by a working group of the German "Gesellschaft für Informatik" (www.fachkomponenten.de) and an approach developed by one of the authors of this paper [Heisel and Souquières, 2004].

Even though a simulation model component is a special kind of software component, a model component is different from a software component due to its reduction of reality and its notion of scheduled time. Therefore, the focus of description needs to be different for simulation model components. In this section, we first provide an overview of the three above-mentioned software engineering structures of describing components, and in the next section make our selection of characteristics that are needed to describe a simulation model component.

Web Services
To facilitate the use of services over the web, the Web Service Description Language (WSDL) has been defined (www.w3.org/TR/wsdl). As communication protocols and message formats are standardized in the web community, it becomes possible and increasingly important to be able to describe the communications in some structured way as well. In this context, the web ontology language OWL is used to publish and share sets of terms called ontologies, supporting advanced web search, software agents and knowledge management.
In this context, OWL-S, an OWL-based web service ontology (www.daml.org/services), supplies web service providers with a core set of markup language constructs for describing the properties and capabilities of their web services in an unambiguous, computer-interpretable form. To support the automation of web service tasks, including automated web service discovery, execution, composition and interoperation, the ontology comprises three parts:

- the *service profile* for advertising and discovering services; it answers the question "What does the service require of the user(s), or other agents, and provide for them?"
- the *process model*, which gives a detailed description of a service's operation, and answers the question "How does it work?"
- the *grounding*, which provides details on how to interoperate with a service, via messages, and answers the question "How is it used?"

Simulation model components should be described according to a similar direction as the WSDL. However, simulation model components cannot be described exactly like the WSDL. The goal of the WSDL is to provide a better, unambiguous meaning of services and to support re-use of services or components by others, where the context of this re-use is not known in advance. However, simulation model components are not necessarily executable, but will be executed in specific simulation environments. This results in constraints that restrict the applications of model components, e.g. validity of the model component, or have a different connotation, e.g. underlying assumptions. These constraints and connotation do not apply in the field of web services, therefore the WSDL does not describe these issues.

Standardized Specification of Business Components
In 1999, the working group "Component Oriented Business Application Systems" within the German "Gesellschaft für Informatik" started to develop a standard specification for business components (www.fachkomponenten.de). The result of this working group is a description of a business component structured along seven levels. The *marketing level* describes business-organizational features of the business component as well as technical initial conditions. The *task level* contains the purpose of the business component and the tasks that it automates. On the *terminology level*, the functional terms of the business domain are explained. The *quality level* describes non-functional properties and quality features, and their corresponding measurement units and methods. On the *coordination level*, the succession relationships between the services of the components and the cooperation with other components are specified. The *behavioral level* contains invariants and pre- and post-conditions. Finally, the *interface level* describes the denomination of business components, services, parameters, data types and failure reports, as well as service signatures and assignment to business tasks.

These seven levels constitute an adequate specification structure for business components. However, for simulation model components, it cannot be used as it is, because business and simulation model components have different characteristics, and hence their descriptions have to stress different aspects. The purpose of business components is to support business processes. The underlying software systems are mostly information systems, where for example the occurrence of events and timing constraints do not play a major role, nor have these systems reductions of functionality compared to reality. We encountered lack of ability to describe those issues, which are important when a model developer makes the decision to use a simulation model component.

Specifiying software components for interoperability
Heisel and Souquières [2004] have developed a specification structure for software components that covers the functional aspects of such components. This structure is aimed at supporting the decision whether two components can be plugged together or not. The description of a software component consists of the specification of its export (supplied) interfaces, its import (required) interfaces, a usage protocol relating the export interfaces, and a relation between export and import interfaces. That relation states which export service relies on which import service.

The specification of an interface consists of an interface data model (IDM), a set of operation specifications, and a usage protocol for the interface operations. The IDM gives a definition of all data used by the interface operations. Every caller of an interface operation must adhere to the data format specified in the IDM. The IDM also contains invariants that express integrity constraints on the data. The specification of an operation consists of its signature (i.e., its name and the names and types of its input and output parameters), a precondition (characterizing the situations where the operation can be applied successfully), and a post-condition (specifying the effect of the operation). The usage protocol specifies the order in which the operations may be applied.

The structure provided by Heisel and Souquières [2004] focuses on the interfacing of components. This is very important for good composition of any type of components, but not the main issue in simulation model component composition. The interfacing becomes an issue once the semantics of functionalities and level of reduction are clarified. Therefore, solely using the structure of Heisel and Souquières [2004] is not enough for describing simulation model components.

OVERVIEW OF ITEMS FOR DESCRIBING SIMULATION MODEL COMPONENTS

All three specification frameworks described in the last section have some added value, but none of them goes into detail with relation to the representation of the simulation model components functionalities. However, model developers gain their trust and make decisions for use on this information about the simulation model component, because this will result in the level of reduction and abstraction that is realized in simulation models. Another difference between simulation model and software components is that simulation model components have several interface descriptions. In addition to “ordinary” interface descriptions also the interfaces to model developers, i.e. the user-interfacing for parameterization, is of high importance with simulation model components.

Based on the work of Heisel and Souquières, the “Gesellschaft für Informatik” and the WSDL we structure the description of the model component into three categories: *profile* (overview of possible use), *interfaces* (how to access the service and parameterize the simulation model component) and *working principles* (how does the simulation model component perform its functions). In the following we briefly discuss the different items that should be described in these three categories.

Profile of a model component

Application Domain: For a first decision whether or not a model component is suitable to be used in a simulation model, the application domain the component falls into must be known. Examples of such domains include computer networks, supply chain management, traffic, military, or health care simulation.
Name: A clear and representative name that enables model developers to grasp its intended purpose.
Existing ontology: Some domains have ontologies or taxonomies to follow. If such a standard is followed this should be identified.
Objective: Experiments that can be performed with models that are composed with this simulation model component.
Responsible persons/developer: In order to increase the trust that model component users have in the components, it may be important for them to know the source of the component. Is the developer of the component known as an expert in the application domain?
Purpose: A natural language description of the model component should include a rough sketch of the functional behavior and thus the purpose that the simulation model component can fulfill in a simulation model.
Simulation environment: many simulation model components are only usable within a certain modeling and/or runtime environment. For example, a component described as a DEVS atomic model [Zeigler et al, 2000] cannot easily be used to be part of an ARENA model (www.arenasimulations.com).
Underlying assumptions: The underlying assumptions identify when a model can be applied and when it cannot be applied. For example, if a traffic model component assumes that no blocking of crossings will occur, the model component should not be used for situations where this blocking plays a central role. Similarly, if a bus component is modeled as a mass point, then single passenger movement cannot be described with that component. The assumptions are often closely related to the overall objective of designing a model component.
Validation: Validation and verification (V&V) is one of the key issues when building and using a simulation model [Balci 1997]. Model validation makes sense only in the context of a specific purpose of the model and question to be answered with the model with known accuracy. Hence, using a simulation model component that is valid in a certain context does not guarantee that this component is valid in another context, too. Therefore the validation will show an overview of testing environments and experimental

frames that have been used to validate the behavior and representation of the simulation model component. Furthermore, case studies and problem descriptions that show the successful and unsuccessful use of the component may be valuable.

Classifications: there are a number of model (component) classifications that should be used in order to decide whether or not the component can be used in a given context. These include:

1 Discrete vs. continuous time, or state model components (or hybrid) respectively.
1.1 Continuous: partial/ordinary differential equations, linear/non-linear, stiff/nonstiff.
1.2 Discrete: event-based/process-based/activity-scanning
2 Scope and use of the model component: metrically scaled variables (e.g. reals), variables with ordinal or nominal scale, or a mixture of both.
3 Stochastic vs. deterministic
4 Modeling paradigm: for example, resource-queue-entity, state-charts, bond-graphs, Petri-nets.

Interfaces of a model component

In- and out-ports including associated signatures: This is the technical interface description as identified in all the software structures. It provides an overview of the triggers or messages the model component can receive and will call to other components. The associated signatures are an overview of the function or process started when something enters by an in-port.

Relations between interfaces and invariants: Sometimes models realize a simple reaction, which means they consume certain resources and generate certain products (e.g. in areas like Systems Biology). Invariants are known relations and integrity constraints of in- and output values should be listed. For example: "outputMoney <= taxes + savingsMoney".

Types of interfaces: In a software application, a software component can only communicate with another software component or a user. In a simulation model component, more types of interfaces can exist. Examples are a simulation model component with a software component or a simulation model component with a model user in a gaming situation.

Parameterization: Which values of the component can/must be parameterized by the model developer to enable a system specific representation? Parameterization information also should include the allowed ranges of values for parameterized attributes. Most often the model developer can parameterize the behavior and attributes of a simulation model component through a user interface of the simulation environment.

Expected effects of changes to parameters: The influence of parameterizable values should be clear to the simulation model developer

Visualization / performance indicators: Output generated by the simulation model component that provides insight in the behavior and state of the component. Visualization mainly refers to insight during the simulation run. Examples of visualization are console with messages or even Virtual Reality animations.

Working principles of a model component

Description of the functionality: Model developers will not use a simulation model component unless the behavior of the model component is absolutely clear to them. Model developers will have the risk of inconsistency in their model, different use of resources, different levels of abstraction, or a process flow that does not represent their system. For software components, pre-and postconditions and usage protocols suffice, but in simulation model components, the functional behavior should be described in detail, preferably with a standardized and formalized description, like for example the DEVS-framework [Zeigler et al., 2000]. Alternative ways of describing the functionalities of the simulation model component and the translations of input at in-ports to an output are state charts, Petri-Nets, timed automata, flow charts, sequence diagrams and pseudo code.

Illustrations of how it works: To support a faster decision whether or not to use a component, as well as an easier integration of a component into a composed model, the description of the functionality should be complemented by example models using the component, training material, or animation movies, respectively.

EXAMPLE DESCRIPTION OF MODEL COMPONENTS

We illustrate the description structure for simulation model components with two examples. The first example is a simulation model component representing a local authority. The second example is a simulation model component representing a retailer that should be usable in any supply chain.

Example 1: Local Authority

Profile of the local authority

Application domain: Simulation of social communities, i.e. sociology, demography, and economy.

Name: Local Authority - a model of a rational, social actor[1].

Existing ontology: . Unlike medical or biological application areas, application areas like sociology, demography, and economy do not maintain taxonomies to structure the most widely used terms and describe their semantics.

Objective: Analyse consequences of catastrophes

Responsible persons / developers: Mathias Roehl, University of Rostock, Department of Computer Science and Ulf Ewert, Technical University of Chemnitz, Department of History.

Purpose: The model component models a rational actor who controls a society and its economy. This control is based on

[1] We follow the definition of Rapoport [1980] of rational actor.

qualitatively assessing the current situation in the society, and based on preferences to derive a plan of activities that are executed stepwise. Each "qualitative step", e.g. the increase of taxes, is translated into quantitative information, e.g. an increase of 5 percent. The model component invokes an external planner for plan generation, and thus depends on its availability.

Simulation environment: The local authority is implemented in James (www.informatik.uni-rostock.de/mosi/en/research/projects/cosa.html). In the future, the component should be re-usable in most DEVS-based simulators, thanks to the standardization efforts of DEVS-based simulators (www.sce.carleton.ca/faculty/wainer/standard). Note that although external software is invoked during the simulation run, the basic DEVS is used. The computer time that is used for planning is ignored. Efficiency plays no major role. Hence, the planner is invoked synchronously within a transition. See Schattenberg and Uhrmacher [2001] for more sophisticated realizations.

Underlying assumptions:

1) The actor acts in a rational manner.
2) As long as a plan is executable, it will be executed. New options are only taken into account when a re-planning becomes necessary.

Validation: The model component has been tested for evaluating management strategies in overcoming the aftermath of disasters in pre-modern towns. The results are documented in Ewert et al [2001 and 2003].

Classification

1) Discrete, time-stepped model. Although implemented in a discrete-event simulation system, the model component executes one time step after each other. In the context of the pre-modern town, one time step of simulation time corresponded to 7 days in physical time.
2) Semi-quantitative. Whereas the model components' interaction with the model environment is based on quantitative data, its preferences and the developed decisions to act are kept in qualitative terms. The knowledge base is a collection of facts, and supports a simple deduction algorithm.
3) Deterministic. The model component makes no use of stochastic variables. Given the same initial state and the same inputs, the local authority will develop the same plans assuming that the invoked planning mechanism works in a deterministic manner.
4) Process-Based World View using State Charts and, Agent-Oriented Modelling Paradigm. The Local Authority model component is based on a B(eliefs)D(esire)I(ntention) Agent, which is a sub-class of deliberative agents.

Interfaces of the local authority

In- and outports: The model component expects inputs for commodities and information. Ewert et al. [2001 and 2003] mention the commodities grain, labour, and money. The class information subsumes the following classes: grain price, labour price, taxes, migration, public opinion, and marriages.

The model component produces the class commodities, and the class power as output. Ewert et al. [2001 and 2003] describe that if money is sent to the grain market the local authority simulation model component assumes that this will be interpreted as a request to buy grain. If money is sent to the labor market the local authority simulation model component assumes that it initiates a job creation program. The class power subsumes the following classes: minimum and maximum prices for grain market and labor market, taxes for the goods markets and the population (stove tax), marriage restrictions for the marriage market, and migration rules.

Relations between interfaces and invariants: The amount of commodities is larger than or equal to zero. Further money spent in the community is always less than or equal to the current tax income added to the saved money. Thus, the local authority does not spend more money than it has. Taxes are always smaller than or equal to one hundred percent. The sold grain is always less than or equal to the stored grain, and the stored grain diminishes with a certain depreciation rate.

Types of interfaces: Only the interfacing between simulation model components is described above. The local authority also has a connection to external software to generate the plan to be executed. The model component expects a partial plan as input from the external software. It consists of a sequence of actions, each denoted by its name, and numbers that indicate in which sequence the plan steps have to be executed.

Parameterization: The possible parameterization of the model component reflects its overall complexity. State variables, as well as the local authority's beliefs about itself and its environment are initiated. This includes how much money, how much grain etcetera are available. Variables that do not change over the simulation run are goals, the preferences among them, and the set of operators that can be applied to reach these goals. Quantitative information is transformed into qualitative information based on certain parameters, e.g. when are prices to be considered low, when are savings high etc. The set of initial plan operators can be extended. Qualitative decisions are transformed to quantitative measurements that can be applied in the town, e.g. if an increase of taxes is decided, what will be the increase, if grain is to be sold, what percentage of the grain on stock shall be put on the market, et cetera.

Expected effects of change to parameters: The preferences among the goals definitely have an impact on the plan generated and the single activities, and thus on the development of the monitored and controlled community. For example, the default preference of the model component favors the income of the local authority above the fast recovery of the population [Ewert et al., 2003].

Visualization / performance indicators: Current beliefs, goals and the developed plan can be inspected during the simulation.

Working principles of the local authority

The behavior of the local authority follows the typical BDI architecture. The following pseudo-code explains its behavior:

```
If inputs then currentBeliefs :=
   updateBeliefs(beliefs, input);
If nonempty(plan) then
   currentPlanStep := selectNextPlanStep(plan);
If noPlan or notExecutable(currentPlanStep) then
   begin
    goals := selectGoals(beliefs, goals);
    plan:= generatePlan(beliefs, operators,
   goals);
    currentPlanStep := selectNextPlanStep(plan);
    output := currentPlanStep;
   end
outputToModelComponent:=quantisize(beliefs,current
   PlanStep);
```

Example 2: Retailer in Supply Chain

Profile of retailer in supply chain

Application Domain: Supply Chain Management
Name of simulation model component: Retailer
Objective: Modelling of information exchange in supply chains.
Responsible persons/developer: Designed by Corver [2001] and implemented by Hee [2002].
Purpose: The retailer receives orders from customers and places orders for goods at its suppliers. The complete process is followed from initial order quotations to billing.
Simulation environment: The component is implemented in eM-plant; Arena and Java to be used with D-SOL.
Underlying assumptions: The retailer always has enough money; employment costs are not considered.
Validation: Component has been applied in several test-cases; so far no serious use.
Classifications:

1) Discrete event
2) Money is in "units"; further sizes are not considered
3) Several sub-functionalities can contain stochastics
4) Modeling paradigm is resource-queue-entity

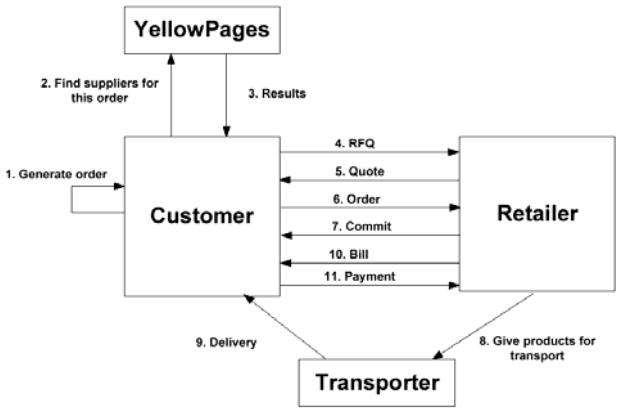
Interfaces of retailer in supply chain

In- and out-ports including associated signatures: Signals are identified for exchanging information with customers and suppliers. Too much to mention all in and out-ports in this article. For more details, see Hee [2002].
Relations between interfaces and invariants: Incoming signals will result in execution of selected functionalities, such as ordering, calculate price or start transportation.
Types of interfaces: During the execution of the model, only signals are exchanged between simulation model components. The implementation in Java is prepared for exchange with real databases or even applying in a simulation game.

Parameterization: Model developers can adjust almost everything. Each functionality of the retailer contains its own additional user-interfacing. Some examples are the time for reacting to a request for quotation or the profit margin.
Visualization / performance indicators: Overview of the functionalities and selected performance indicators, such as stock size and number of orders to be processed.

Working principles of retailer in supply chain

The retailer has several functionalities. The figure below shows the high-level process of an order. Actions 1 to 4 and 6 will be performed by the retailer when it orders goods at its supplier. Hee [2002] describes in more detail how these functionalities are performed.



CONCLUSIONS

In this paper, we have proposed a description structure for simulation model components. Such a description structure is crucial for fully exploiting the advantages a component-based approach can offer. It supports the re-use of components developed by third parties, without being obliged to inspect every detail of the component itself. Only with concise and meaningful descriptions can developers of simulation models really save time and effort by using components. The descriptions may also serve as a contract between component supplier and component consumer. Moreover, they support the development of a functioning market for simulation model components.
The description structure we have developed is based on specification approaches for software components and web services. These specification approaches put the stress on the semantics of components. Describing the semantics of a component is a common goal for software and simulation model components. However, software and simulation model components are so different in nature that a specialized description structure for simulation model components is needed.

Our description structure is tailor-made for simulation model components. It contains sections where the simulation-specific characteristics of a model component

can be described, mainly focusing on the semantics of interfaces and functionalities to represent reduction and abstraction. On the other hand, it was our goal to accommodate all kinds of simulation model components in different paradigms, for example discrete as well as continuous and stochastic as well as deterministic ones.

With two examples, we have demonstrated the usefulness of our description structure. In the future, we intend to perform further case studies, covering all relevant kinds of simulation components.

The current description is a first attempt to describe model components. Many of the proposed slots are filled in unstructured text form. We made a first step towards clarifying what information is needed to support reuse of model components. The next step will be to find more suitable representations, which make it easier to search for and analyze this information.

ACKNOWLEDGEMENT
The authors want to thank the organizers of the Dagstuhl seminar "Component-Based Modeling and Simulation" for the energetic and motivating discussion environment they prepared for us, which was the basis for the presented framework.

REFERENCES

Balci, O. "Principles of Simulation Model Validation, Verification, and Testing" In: *Transactions of the Society for Computer Simulation International*, Volume 14, Issue 1, pp. 3-12, 1997

Bruneton, E.; T. Coupaye; J.B. Stefani "Recursive and Dynamic Software Composition with Sharing" In: *Proceedings of the 7th ECOOP International Workshop on Component-Oriented Programming*, 2002

Corver, A. *Supply chain visualization: Simulation as a means to gain insight in the supply chain*. Master Thesis, Delft University of Technology, Netherlands, 2001

Davis, P.C.; P.A. Fishwick; C.M. Overstreet; C.D. Pegden. "Model Composability as a Research Investment". In: J.A. Joines; R.R. Barton; K. Kang; P.A. Fishwick (Eds.), *Proceedings of the 2000 Winter Simulation Conference*, pp.1585-1591, 2000

Diamond, R.; J.O. Henriksen; C.D. Pegden; A.P. Walker; C.R. Harrell; W.B Nordgren; M.W. Rohrer; A.M. Law. "The Current and Future Status of Simulation Software (Panel)" In: E.Yücesan; C.H. Chen; J.L. Snowdon; J.M. Charnes (Eds.) *Proceedings of the 2002 Winter Simulation Conference*, pp.1633-1640, 2002

Ewert, U.C.; M. Röhl; A.M. Uhrmacher. "The role of deliberative agents in analyzing crises in pre-modern towns", *Sozionik* , Volume 1, Issue 3, 2001

Ewert, U.C.; M. Röhl; A.M. Uhrmacher. "Consequences of Mortality Crises in Pre-Modern European Towns" In: A.Fürnkranz-Prskawetz; F.C. Billari (Eds.) *Agent Based Computational Demography*, pp.175-196, 2003

Hee, R.van der. *Building blocks for Real-Time Supply Chain*, Master Thesis, Delft University of Technology, Netherlands, 2002

Heisel M.; J. Souquières "Spécification de composants pour assurer leur interopérabilité" Submitted for publication, 2004

Keller, L.; C. Harrell ; J. Leavy. "The three reasons why simulation fails". In: *Industrial Engineering*, Volume 23, Issue 4, pp.27-31, 1991

Nance, R.E. "A history of discrete event simulation programming languages" In: *ACM SIGPLAN Notices*, Volume 28, Issue 3, pp.149-175, 1993

Overstreet, C.M.; R.E. Nance; O. Balci. "Issues in Enhancing Model Reuse", In: W.H.Lunceford; E.H. Page (Eds.) *First International conference on Grand Challenges for Modeling and Simulation*, pp.1-5, 2002

Page, E.H.; J.M. Opper. "Observations on the complexity of composable simulation" In: P.A. Farrington; H.B. Nembhard; D.T. Sturrock; G.W. Evans (Eds.) *Proceedings of the 1999 Winter Simulation Conference*, pp.553-560, 1999

Pater, A.J.G.; M.J.G. Teunisse. "The Use of a Template-Based Methodology in the Simulation of a New Cargo Track from Rotterdam Harbor to Germany" In: S. Andradottir; K.J. Healy; D.H. Withers; B.L. Nelson (Eds.) *Proceedings of the 1997 Winter Simulation Conference*, pp.1176-1180, 1997

Rao, A.S.; M.P. Georgeff. "Modeling rational agents within a BDI-architecture". In: J. Allen; R. Fikes; E. Sandewall (Eds.) *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pp. 473-484, 1991

Rapoport, A. *Mathematische Methoden in den Sozialwissenschaften*. Physica Verlag, Wuerzburg, 1980

Robinson, S. "Three sources of simulation inaccuracy (and how to overcome them)" In: P.A. Farrington, H.B. Nembhard, D.T. Sturrock, G.W. Evans (Eds.) *Proceedings of the 1999 Winter Simulation Conference*, pp.1701-1708, 1999

Schattenberg, B.; A.M. Uhrmacher. "Planning Agents in James" *Proceedings of the IEEE*, Volume 89, Issue 2, pp. 158-173, 2001

Valentin, E.C.; A. Verbraeck. "Simulation using building blocks" In: F.J.Barros; N.Giambiasi (Eds.) *Proceedings conference on AI, Simulation and Planning*, pp.65-71, 2002

Zeigler, B.P.; H. Prähofer; T.G. Kim. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press, San Diego, 2000

Zeigler, B.P. *Multi-Facetted Modelling and Discrete Event Simulation*. Academic Press, San Diego, 1984