Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

EXPLOITING WEB SERVICE TECHNIQUES FOR COMPOSING SIMULATION MODELS

Mathias Röhl
Florian Marquardt
Adelinde M. Uhrmacher

University of Rostock
Albert-Einstein-Str. 21
18059 Rostock, GERMANY

ABSTRACT

Two basic approaches to simulation model composition can be distinguished, depending on whether the unit of composition is a model specification according to a certain modeling formalism or the component is a simulation system wrapping the actual model implementation. Model-based approaches mainly provide user-friendly means for modular-hierarchical construction of complex models but come with limited capabilities for compositional reasoning. Simulation-based composition approaches explicitly distinguish between interface descriptions and model implementations and thereby facilitate a reasoning about compositions based solely on publicized interfaces descriptions. However, compositional reasoning about interoperability at the conceptual level and on overall model validity is currently not very elaborated. This paper takes a closer look at web services technologies and discusses how to adapt them to the problem of model composition.

1 INTRODUCTION

To reuse and combine simulation models within different experimentation contexts poses severe challenges, mainly induced by differences in assumptions, aims, and constraints (Davis and Anderson 2004, Overstreet, Nance, and Balci 2002), also called context dependencies (Yilmaz and Ören 2006). The importance of the abstractions that underlie a model have led to distinguish the experimentation context from the context of model realization (Yilmaz and Ören 2006). The latter focuses on the interfaces and the question: whether components can be put together and connected properly (Petty and Weisel 2003a). While the realization context makes explicit all direct and indirect dependencies to other models, the experimental context, aka experimental frame (Zeigler 1984), describes the conditions of experimentation. These two contexts are sometimes also referred to as the engineering and the semantic aspect of composability, respectively (Weisel, Petty, and Mielke 2004).

In the following we review existing approaches to compose simulation models and web services. Afterwards, we will explore how web service composition techniques may be exploited to support a conceptual sound and valid composition of models.

2 COMPOSITION OF SIMULATION MODELS

Two basic approaches to simulation model composition can be distinguished, depending on whether the unit of composition is a model specification according to a certain modeling formalism, or the component is a simulation system wrapping the actual model definition.

2.1 Composition within Modeling Formalisms

COTS simulation systems like Arena (Kelton, Sadowski, and Sturrock 2003) offer libraries of models that can be assembled together via graphical user interfaces. Modeling within these tools is mainly based on object-oriented modeling concepts supported by visual representation of model entities. Most of these tools focus on a particular application domain. The state of the art in this domain is set by Modelica (Elmqvist, Mattsson, and Otter 2001) as an open and standardized language, which aims at the description of complex physical systems. Models can contain sub-models that are connected via their ports, thereby, a hierarchically construction of models is supported. Modelica exploits intensively object-oriented concepts. Models are defined as parametrize-able classes and may be substituted according to class inheritance. The flexibility of Modelica has motivated the development of a set of model libraries, mainly for continuous and hybrid modeling.

DEVS (Discrete EVent System specification) is a general-purpose modeling language, which supports modular-hierarchical modeling in the realm of discrete-event

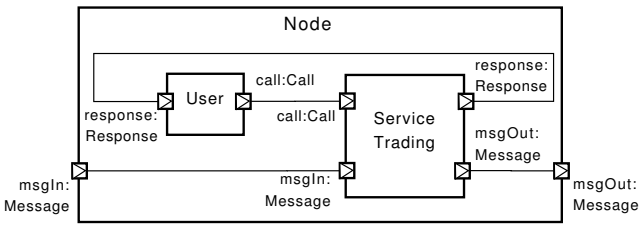

Figure 1: A network node as a coupled DEVS model.

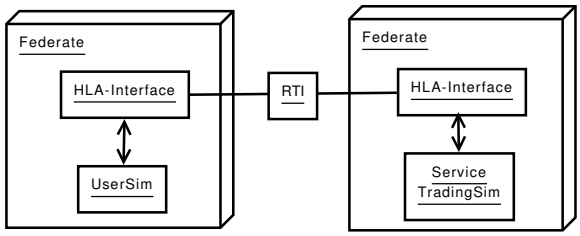

Figure 2: Composition of simulation models with HLA.

simulation. DEVS models are basically time-triggered state automatons with arbitrarily scaled state variables. DEVS distinguishes between atomic and coupled models. An atomic model is described by a state set, a set of input and output ports, an internal and external transition function, an output function, and a time advance function. The later defines after which time an internal event is due, whereas external events are triggered by the arrival of inputs (Zeigler, Praehofer, and Kim 2000).

Coupled DEVS models support the hierarchical, modular construction of models. A coupled model is described by an input and output set, the set of its component models, which may be atomic or coupled, and by the couplings that exist among them. Coupled models enable the structuring of large models into smaller ones.

In DEVS each model is only allowed to communicate via its ports. Models do not know of each other (unless such knowledge is explicitly communicated) and models do not know with which models they become coupled. Coupling schemes are always defined on the next higher level, i.e. within the surrounding coupled model. Thereby, coupling takes place at the time of deployment of a model and not at the time of model definition. This is called *delayed binding*. As closure under coupling holds for the DEVS formalism, models can be constructed modular-hierarchical. Figure 1 shows a simple model of a network node defined as a coupled model. The node contains a user and a service trading model as sub models. Both these models may be atomic or coupled ones.

Modelica and DEVS remain at the syntactical level for checking compatibility. Furthermore, with these formalisms, interface are not entities in their own right, but implicit in the definition of a model's input and output ports and thus are not separated from the model. Advertisements of interface descriptions as in web service approaches are hampered.

Therefore, explicit interface descriptions have been assigned to DEVS models (Yilmaz 2004). Role descriptions formalize patterns of interactions with respect to a subset of input and output ports, and focus on visible behavior. Roles are defined as non-deterministic partial automata. Two instances of such automata can be checked for compatibility.

Generally, semantic aspects can more easily be integrated if a formalism is restricted to a specific application domain. For example, the Systems Biology Markup Language (Hucka et al. 2004) refers to domain-specific ontologies and annotates variables and methods with additional information (Le Novère et al. 2005) to enable semantic compatibility checks.

2.2 Composition Based on Distributed Simulation

Simulation aims at the execution of operational models. Consequently, composition of models may be realized via coupling of (distributed) simulation systems. In trying to combine different simulation systems, compatibility manifests itself as the problem of interoperability. Prominent example for a general distributed simulation approach is HLA (IEEE 2000). HLA's basic concept are *federates*. A *federate* may be a model, observer, or an interface to a human actor. A combination of *federates* for a particular purpose is called a *federation*. The technical mean for connecting federates is provided by the *Run Time Infrastructure*, abbreviated RTI. Figure 2 shows the coupling of a user simulation and a service trading simulation via HLA.

Data to be exchanged via the RTI have to be defined explicitly based on the *Object Model Template (OMT)*. However, HLA was mainly designed to resolve heterogeneity at a technical level. In the context of simulation-oriented composition approaches a framework has been developed, which distinguishes several levels of interoperability: the Levels of Conceptual Interoperability Model, abbreviated LCIM, (Tolk and Muguira 2003). In its current form (Tolk 2006), LCIM comprises six levels of interoperability. Technical interoperability is reached if components are able to exchange data. Syntactic interoperability requires an unambiguously defined data format. Definition of meaning of data leads to semantic interoperability. Please note, that semantic herein refers to the semantics of exchanged data, not semantics of the model. Visibility of data usage leads to pragmatic interoperability. Dynamic interoperability requires successful interaction over time. For conceptual interoperability, a simulation model's underlying abstractions and assumptions have to be aligned.

The Base Object Model (SISO 2006), abbreviated BOM, is aimed at achieving higher levels of interoperability for simulation-oriented composition of models. BOMs enrich operational model components with descriptions at the se-

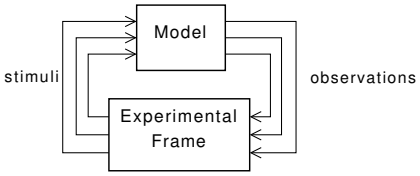

Figure 3: Formalization of validity by experimental frames.

mantic, dynamic, and conceptual level. As a first step to achieve valid simulation models, composition of BOMs may be based on a declarative description of an experimenter's intention, which is specified e.g. via SRML, the Simulation Reference Markup Language (Reichenthal 2002).

However, the conceptual level of interoperability, i.e. whether abstractions and assumptions realized in model candidates are compatible, is not yet addressed in a formal manner. Thereby, reasoning about a composition's validity is currently very limited.

2.3 Validity of Compositions

Validation is a process for substantiating that a model's accuracy is within acceptable bounds with respect to a specific purpose (Sargent 2005), also known as *experimental frame*.

The concept of experimental frame was introduced by Zeigler already in the 80s (Zeigler 1984). The experimental frame can be perceived as a system itself. An experimental frame reflects the experimental conditions explicitly and is intended to be coupled to the system to be validated, i.e. it provides input to the model under consideration and evaluates its output.

Zeigler (Zeigler 1984) introduced a hierarchy of system specifications to capture knowledge about timed systems at different levels of detail in a formal manner. At the lowest level a system is described as a black box. Climbing up the hierarchy knowledge about behavior, internal states, and structure are added successively. The hierarchy of systems specifications is accompanied with a hierarchy of system morphisms which express property preserving relations between systems at different levels. Morphisms at higher levels imply morphism at lower levels.

Given the hierarchy of system specifications, validity can be expressed as a relation between a reference model and the model under consideration. The reference model represents the model that fulfills all requirements defined in the experimental frame, i.e. all conditions to be fulfilled w.r.t. a certain objective. An experimental frame may be defined as a system that provides input values (stimuli) to the model and receives output values (observations) from the model (cf. Figure 3). To account for a certain degree of accuracy, morphism can be relaxed to approximate morphisms (Zeigler, Praehofer, and Kim 2000), e.g. equality of values may be relaxed to values being in a certain range.

Furthermore, different kinds of validity can be distinguished, depending on the level of the hierarchy that is used as a reference system (Zeigler 1984). Replicative validity requires equivalence at level 1 of the hierarchy, which essentially means replication of input/output data. According to Zeigler, predictive validity requires equivalence at level 2, i.e. incorporation of knowledge about the initial state of a system. Structural validity can be checked based on knowledge about internal states of a system. Compositions are expressed at a still higher level. As systems at higher levels can be mapped to system descriptions at lower levels, flattening a composed model is one possibility to make it in principal available to validity checks at lower levels. However, these concepts and their exploitation for model composition are currently of a rather theoretical nature.

Work in the area of theoretical computer science is aimed at providing a whole spectrum of relations between system descriptions amenable to algorithmic evaluation (van Glabbeek 2001). First steps to adopting these techniques for modeling and simulation has been done under the term Theory of Semantic Composability (Petty and Weisel 2003b).

Within the theory of semantic composability models are expressed as computable functions and simulation as execution of functions (Petty, Weisel, and Mielke 2005). Formally, simulation is defined via a deterministic labeled transition system (LTS). Validity of models can now be expressed as an equivalence relation over LTS, namely by *weak bisimulation equivalence*. Same as with Zeigler's approach a somehow perfect model is used as a reference and strict equivalence relations can be relaxed, e.g. based on a metric.

Formalization of validity is central to automation of validation. At the level of structural validity, bisimulation equivalences are an elegant means of reasoning about validity of models. Weak bisimulation equivalence formalizes the notion of observational equivalence in conformance with recent advances in theoretic computer science. The challenge is to find an appropriate reference system, i.e. a formal representation of the intended model. Validation of simulation models remains challenging. As it requires creativity and insight (Balci et al. 2002), it cannot be easily automated.

3 COMPOSITION OF WEB SERVICES

Technically interoperability of Web Services is ensured by the protocol stack of the Internet.

At the syntactical level, the Web Service Description Language (WSDL) describes a web service by its in- and outputs, its location, and potential error exceptions. WSDL is standardized by the World Wide Web Consortium (W3C) (W3C 2006). The semantic web initiative of the W3C aims to complement syntactical descriptions with computer processable semantic information (Fensel,

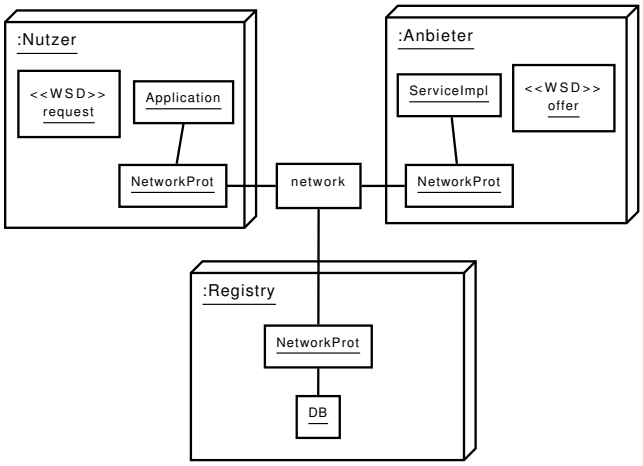

Figure 4: Composition of web services.

Hendler, and Liebermann 2003). Recent approaches like OWL-S take additionally the dynamic aspect of interoperability into account. It is an extension of basic OWL and consits of three parts; the services profile, its model and its grounding. The grounding represents the WSDL description of the service. The model contains the process description of the service, wherein atomic, combined and abstract processes are distinguished. Furthermore pre- and postconditions of the service are included. The service profile contains a short description of what the service does. It is used for service discovery.

Similar to validity in the area of modeling and simulation, web service compositions are subject to quality requirements, which pertain to the context of use, e.g. *connectivity*, *nonfunctional quality-of-service properties*, *correctness*, and *scalability* (Milanovic and Malek 2004).

3.1 Web Service Composition in General

Each composition method aims at combining existing services to fulfill a specific more complex purpose. These purposes may be defined as tasks or as goals. The wording primarily depends on the area of research and the involved algorithms. Composition in the web service domain focuses on creating more or less sequential plans for service executions also called service chains (Ponnekanti and Fox 2002). Composing services happens in a concurrent and dynamic manner (Kuter et al. 2004). The main challenges in web service composition are finding appropriate services (semantic dimension) and matching their in- and outputs (syntactic dimension). Among these a lot of smaller problems arise in web service composition. Each of which faces again a set of diverse problems. E.g. matching has to deal with multiple effects and potential side-effects of the services (Küster et al. 2007). Most approaches to web service composition utilize *workflow methods* or *AI-planning* methods (Rao and Su 2004).

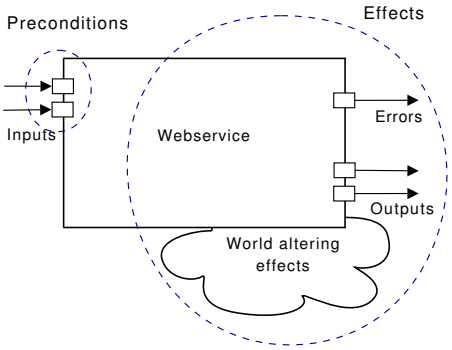

Figure 5: Plan operator representation of a web service.

3.2 Workflow Methods

Workflow methods originate in describing business processes (van der Aalst, ter Hofstede, and Weske 2003). A workflow usually represents a static model, which can be described as a graph whose nodes are definitions of services and whose edges represent the exchange of data between services. The workflow model comprises descriptions of required services and does not contain the concrete services themselves. Searching and matching of available services according to the description of each service node can be performed at the time of instantiation or during execution time.

To face the nature of high dynamic environments like the Internet where previously used services might no longer be available, searching and matching of a service must be performed every time a node is executed. Business Process Execution Language for Web Services (BPEL) is an OASIS standard to define workflows that integrate webservices (Andrews et al. 2003). Workflow methods for web services composition require a priori knowledge about the information flow and connections between the involved services. Major parts of the systems' semantics are included in the workflow description implicitly.

3.3 Planning

In order to express web service composition as a planning problem some requirements must be fulfilled. An initial description of the world, a set of acceptable goal states of the world, a list of available actions, their pre-conditions for being executable, and their effects are needed (Carman, Serafini, and Traverso 2003). All required information, including parameters, are defined as inputs to web services. The output of web services is made up of data, errors and possibly world altering effects (cf. Figure 5).

Planning assumes a "closed world." This assumption is one of the most discussed constraints in adopting AI-planning methods for web service composition as every knowledge of the Internet must be considered partial and dynamic. Despite this problem planning approaches have

been successfully applied to web service composition (Peer 2005, Küster, Stern, and König-Ries 2005). Among them, Hierarchical Task Networks (HTN) using OWL-S Service descriptions have shown to be a promising approach (Sirin et al. 2004).

4 ADAPTING WEB SERVICE COMPOSITION TECHNIQUES TO SIMULATION MODELS

Web service standards and techniques are currently mainly adapted to simulation-based approaches focusing on the lower levels of interoperability, i.e. the syntactical and technical level.

4.1 Adaptation at Lower Levels

Application of web-service techniques are already in use in the area of simulation-based composition (Tolk 2004, Pullen et al. 2005, Möller and Dahlin 2006). Adaption of web service techniques to simulation-based composition can be done in a rather straightforward manner. Technical and syntactical interoperability issues can be resolved by using SOAP and WSDL for simulation systems. This allows to integrate heterogeneous simulation systems and thereby different kinds of modeling formalisms. Model implementations can be fully encapsulated behind interfaces and kept private. Treating simulation systems as software entities is currently the predominant way of web service adaption. However, an a priori partitioning of a simulation model and communication via declarative messages at runtime hamper an efficient execution of a composed simulation model (Möller and Dahlin 2006).

While web services assume a rather coarse grained exchange of information, simulation models are characterized by frequent bi-directional interactions. Run-time overhead follows directly from the decoupling of implementations by means of declarative interface descriptions. Furthermore, as each model is wrapped by a simulator, the distribution of simulation components enforces the distribution of model components. However, computational complexity of models does not necessarily have to coincide with the component distribution scheme. Finally, simulation models often require multiple instances of a certain model component. While web service techniques naturally support functional modularization of execution logic very nicely, web services are from the experimenter's point of view sub-optimal.

There is an alternative, yet largely unexplored, way to adapt web service description techniques to the problem of simulation model composition. Web service descriptions may also be adapted to model-oriented composition approaches. Instead of letting an WSDL interface declare a set of operations of a simulation system, an interface definition may comprise a set of event port declarations that announce the exchange of typed events as defined in a

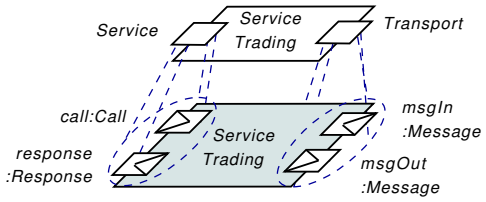

Figure 6: Model interfaces exhibiting roles.

certain modeling formalism, preferably represented in XML (Röhl 2006). Such model definitions are usually not directly executable pieces of software, but become simulated by a certain simulation engine. Therefore, interface descriptions of model components need binding elements. These will not define the technical details of information transmission but relate abstract event ports of a role definition to concrete ports of a model definition (cf. Figure 6).

Interface descriptions for model definitions need to be published to become queried and, eventually, to form the input to a composition engine. Figure 7 depicts an architecture for model components on global scale. Selection of model components is solely based on publicized interface descriptions. If a component is selected, the model implementation will be retrieved from the component provider and will be deployed by the experimenter. The executable simulation model is produced at the experimenter's site (Röhl and Uhrmacher 2006). Whether a simulation model becomes distributed at the time of execution, is completely independent of the distribution of component providers and the composition process. Thus, a flexible and efficient execution of the entire composed model becomes possible. The composed model can be executed in a sequential or parallel manner, on demand, only depending on the simulation facilities of the experimenter. In addition, it becomes possible to let certain parts of the model be executed by specific simulators. These simulators could then be coupled together by using HLA type of synchronization protocols.

Unlike with simulation-oriented composition approaches, which are designed for reaching technical interoperability, compatibility at the technical level is a challenging problem for model-oriented composition approaches. If models need to be integrated that are encoded in different kinds of modeling formalism, model components have to be equipped with a set of different model implementations, each for a certain formalism and simulator engine. Alternatively, transformations to a certain kind of super formalism, e.g. DEVS, may be done (Vangheluwe 2000, Sarjoughian 2006). This is avoided if the simulation engine at the experimenters site supports a sufficient number of formalisms and their combination (Eker et al. 2003).

Model-oriented composition approaches comes with a further drawback. Model implementations are not completely hidden as in the case of simulation-based approaches, where implementations reside at the providers site. In model-

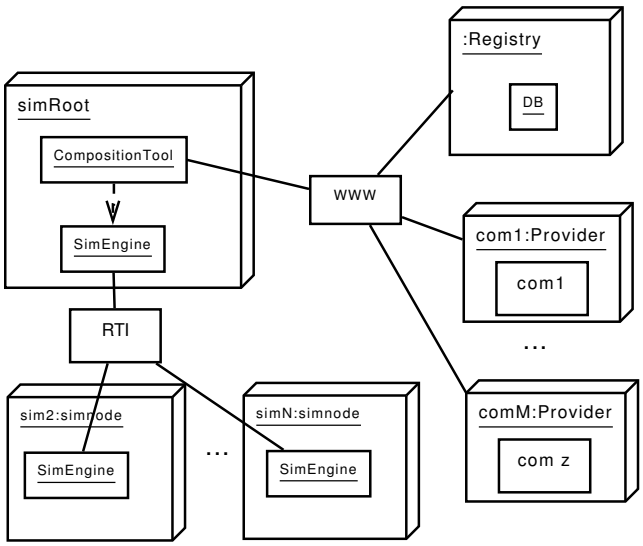

Figure 7: Architecture for model-oriented composition.

based composition approaches privacy issues need to be handled separately and require a special kind of security architecture.

4.2 Aiming at the Conceptual Level and Validity

Simulation-based composition approaches that build upon WSDL (Tolk 2004, Pullen et al. 2005, Möller and Dahlin 2006) may directly benefit from workflow composition techniques. Additional to the description of component interfaces, a description of the workflow is needed. Using interface descriptions, a model composer has first to find appropriate components, and then to match those into the nodes of a BPEL workflow. If a proper model component was found for every node of the workflow, the simulation model is complete.

In comparison to workflow approaches, planning approaches promises more flexible solutions, however also a lot more challenges. To this end, the model composition problem has to be encoded as a planning problem similar to web services. Web services and model components both encapsulate functionality and interact merely via their in- and outputs. Both are designed for special purposes that can be summarized in semantic descriptions. However, differences exist. The validity relation (quality of service) for web services refers to and depends on the time of execution, i.e. whether certain non-functional requirements are fulfilled. Once a composition for several web services is found, it cannot be guaranteed that all involved services will remain available. For simulation and model composition their "validity" can be checked during composition generation.

The biggest problem for applying planning to model composition stems from the nature of interaction between model components. Whereas composition of web services

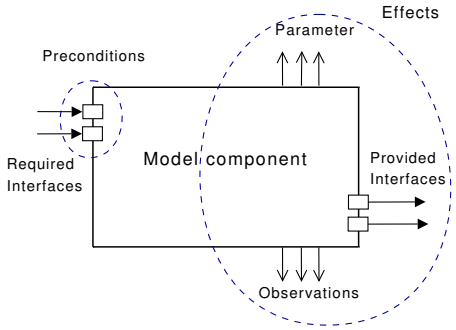

Figure 8: Representing a model component as a plan operator.

results mostly in a sequence of service executions, simulation models are not likely to be chained but act concurrently and interact with each other during the whole simulation run. Thus, planning over model compositions is only feasible if it is constrained to the conceptual level and validity. For composition, planning should operate on the compatibility of realized abstractions within composition candidates and the interaction of the composition with the experimental frame. All technical, syntactical, semantic, and dynamical details have to be left out. Compatibility checking at those levels has to be done independently.

Figure 8 illustrates the concept for encoding a model component interface description as a plan operator. Model components may have initial parameters and may provide a set of possible observations. With respect to the purpose of an experiment, parameters and observations are interpreted as a requirement of the user, i.e. a model has to provide a set of observations and a set of parameters to be suited for a certain experiment. Thus, for planning parameters and observations are considered to be effects of the planning operator representing a model component. To account for the conceptual level, exhibited roles of model components (cf. Figure 6) can be interpreted as indicating the provision of or the dependence on a certain abstraction (in addition to the port descriptions at the syntactical level). Required roles are encoded as pre-conditions and provided roles as effects of the plan operator.

Thus, a set of model components, more precisely their interface descriptions, are represented as planning operators and thereby serve as the backbone of planning. The concrete planning problem must be described by the set of available model components and a goal state G. The planning process will try to apply plan operators to the set of model components to reach the goal state. According to the definition of an experimental frame the goal state can be formulated as a set of parameters and a set of operations a simulation model has to provide.

No functional in- or outputs of model components may occur in the goal state G, i.e. G is orthogonal to the inter-component data flow. Furthermore, all variables in G must be disjunctive since it can not be allowed to have

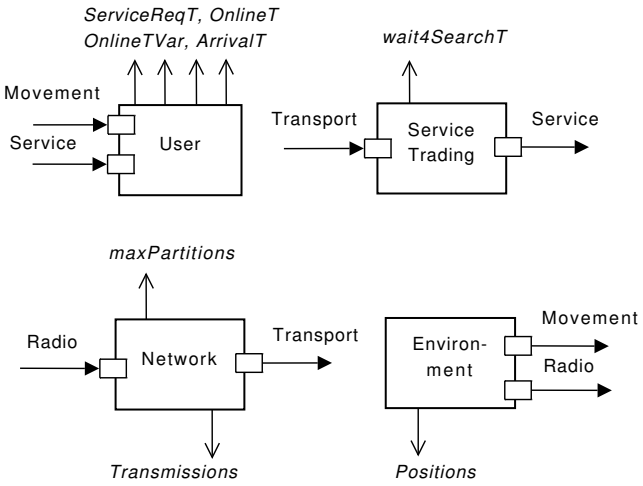

Figure 9: Example components.

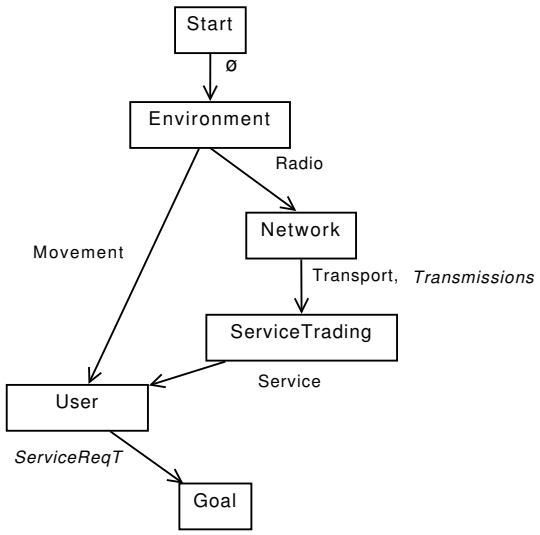

Figure 10: Example plan for a composition.

two or more different assignments of initial parameters or observations. Available model components have to be known at composition time. The selection of components happens depending on the goal definition G. As soon as a set of promising model components is identified, plan generation can start. In addition to checking whether all desired observations have been met, the plan generation takes the interface description of the individual model components into account. Required interfaces have to be matched to offered interfaces.

4.3 Example

Assume a network simulation should be conducted that investigates the effect of different service request frequencies on the number of network transmissions. These data variables form the goal state, i.e. $G = \{ServiceReqT, Transmissions\}$. The set of component candidates may be gained via a keyword or arbitrary meta-data based query to a model repository. Component descriptions are transformed to planning operators as depicted in Figure 9. Based on these operator definitions the planning process tries to fulfill the goal state. E.g. the component *Network* provides the observation *Transmissions* and the component *User* provides a parameter *ServiceReqT*. Both components depend on abstractions realized by other components, namely *Network* requires a model component providing *Radio* propagation and *User* requires other models offering *Movement* management and handling of *Service* calls. If planning is able to resolve all pre-conditions, it will come up with a plan representing the set of components to be used and the coupling scheme among them. Figure 10 shows a possible plan for the example setting.

5 CONCLUSION

Efficient experimentation poses specific requirements on the representation of models at the time of execution. These are partly at odds with the assumptions underlying the web service style to achieve interoperability. While web services aim to minimize interactions and strive for a coarse grained exchange of information, sub components of a simulation model usually exhibit a frequent, close style of interaction. To this end, adapting declarative web service descriptions to models rather than to simulation systems seems to be promising. Thereby, the execution of a simulation model may be completely decoupled from the distribution of component providers and the process of composition. However, model-oriented composition implies other problems that have to be solved, e.g. if model components shall be combined, which are defined in different formalisms, or the model implementation needs to be kept private.

Web service composition techniques have leaped forward to address higher levels of interoperability. Planning approaches promise flexible, however also challenging solutions for reasoning about compositions. We explored how planning techniques can be adapted for approaching the validity dimension of model composition. Transforming the model composition into a planning problem requires to enrich component interfaces with descriptions about parameters, observations, and required and provided abstractions. Still the question has to be answered, how this meta information about models shall be represented. Furthermore, the implementation of concepts and an evaluation based on a model library are required to assess the practical value of planning approaches for contributing to the challenge of model composition.

REFERENCES

Andrews, T., F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. 2003, May. Business process execution language ofr web services.

Balci, O., R. E. Nance, J. D. Arthur, and W. F. Ormsby. 2002. Expanding our horizons in verification, validation, and accreditation research and practice. In *Winter Simulation Conference*, 653–663.

Carman, M., L. Serafini, and P. Traverso. 2003, June. Web service composition as planning. In *Workshop on Planning for Web Services*. Trento, Italy.

Davis, P. K., and R. H. Anderson. 2004, April. Improving the composability of DoD models and simulations. *JDMS* 1 (1): 5–17.

Eker, J., J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong. 2003, January. Taming heterogeneity — the Ptolemy approach. *Proceedings of the IEEE, Special Issue on Modeling and Design of Embedded Software* 91 (1): 127–144.

Elmqvist, H., S. E. Mattsson, and M. Otter. 2001. Object-oriented and hybrid modeling in modelica. *Journal Européen des systèmes automatisés* 35 (1): 1–10.

Fensel, D., J. A. Hendler, and H. Liebermann. (Eds.) 2003. *Spinning the semantic web*. The MIT Press.

Hucka, M., A. Finney, B. Bornstein, S. Keating, B. Shapiro, J. Matthews, B. Kovitz, M. Schilstra, A. Funahashi, J. Doyle, and H. Kitano. 2004, June. Evolving a lingua franca and associated software infrastructure for computational systems biology: The systems biology markup language (SBML) project. *Systems Biology* 1 (1): 41–53.

IEEE 2000, September. Standard for modeling and simulation (M& S) High Level Architecture (HLA) — Framework and Rules. Document 1516-2000.

Kelton, W. D., R. P. Sadowski, and D. T. Sturrock. 2003. *Simulation with arena*. 3rd ed. McGraw-Hill.

Küster, U., B. Köning-Ries, M. Klein, and M. Stern. 2007. Diane - a matchmaking-centered framework for automated service discovery, composition, binding and invocation. Submisson for Special Issue of IJEC (International Journal of Electronic Commerce) on Semantic Matchmaking and Retrieval.

Küster, U., M. Stern, and B. König-Ries. 2005, December. A classifcation of issues and approaches in automatic service composition. In *Engineering Service Compositions: First International Workshop*, ed. C. Zirpins, G. Ortiz, W. Lamerdorf, and W. Emmerich, Volume WESC05: Yorktown Heights: IBM Research Devision.

Kuter, U., E. Sirin, D. Nau, B. Parsia, and J. Hendler. 2004. Information gathering during planning for web service composition. In *Proceedings of the Third International Semantic Web Conference*. Hiroshima, Japan.

Le Novère, N., A. Finney, M. Hucka, U. S. Bhalla, F. Campagne, C. J. Vides, E. J. Crampin, M. Halstead, E. Klipp, P. Mendes, P. Nielsen, H. Sauro, B. Shapiro, J. L. Snoep, H. D. Spence, and B. L. Wanner. 2005, December. Minimum information requested in the annotation of biochemical models (MIRIAM). *Nat Biotechnol* 23 (12): 1509–1515.

Milanovic, N., and M. Malek. 2004. Current solutions for web service composition. *IEEE* November December:51–59.

Möller, B., and C. Dahlin. 2006, June. A first look at the HLA evolved web service api. In *Proceedings of 2006 Euro Simulation Interoperability Workshop*. Simulation Interoperability Standards Organization. 06E-SIW-061.

Overstreet, C. M., R. E. Nance, and O. Balci. 2002. Issues in enhancing model reuse. In *International Conference on Grand Challenges for Modeling and Simulation, Jan. 27-31*. San Antonio, Texas, USA.

Peer, J. 2005. Web service composition as ai planning - a survey. Technical report, University of St. Gallen, Switzerland.

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

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

Petty, M. D., E. W. Weisel, and R. R. Mielke. 2005. Composability theory overview and update. In *Proceedings of the Spring 2005 Simulation Interoperability Workshop*, 431–437.

Ponnekanti, S. R., and A. Fox. 2002. Sword: A developer toolkit for web service composition. In *Proceedings of the 11th International WWW Conference (WWW2002)*. Honolulu, HI, USA.

Pullen, J. M., R. Brunton, D. Brutzman, D. Drake, M. Hieb, K. L. Morse, and A. Tolk. 2005. Using web services to integrate heterogeneous simulations in a grid environment. *Future Generation Computer Systems* 21:97–106.

Rao, J., and X. Su. 2004. A survey of automated web service composition methods. In *Proceedings of the 1st International Workshop on Semantic Web Services and Web Process Composition, SWSWPC2004, LNCS*. San Diego, USA.

Reichenthal, S. W. 2002. SRML – simulation reference markup language. W3C Note 18 December 2002.

Röhl, M. 2006, May 28th-31th. Platform independent specification of simulation model components. In *ECMS 2006*, 220–225. Bonn, Sankt Augustin, Germany.

Röhl, M., and A. M. Uhrmacher. 2006. Composing simulations from XML-specified model components. In *Proceedings of the Winter Simulation Conference*, 1083–1090: ACM.

Sargent, R. G. 2005. Verification and validation of simulation models. In *WSC '05: Proceedings of the 37th Winter simulation conference*, 130–143: Winter Simulation Conference.

Sarjoughian, H. S. 2006. Model composability. In *WSC '06: Proceedings of the 37th conference on Winter simulation*, 149–158: Winter Simulation Conference.

Sirin, E., B. Parsia, D. Wu, J. Hendler, and D. Nau. 2004. Htn planning for web service composition using shop2. *Journal of Web Semantics* 1 (4): 377–396.

SISO 2006, March. Base object model (BOM) template specification. SISO-STD-003-2006.

Tolk, A. 2004, April. Composable mission spaces and M&S repositories – applicability of open standards. In *Spring Simulation Interoperability Workshop*. Washington, D.C.: SISO. paper no. 04S-SIW-009.

Tolk, A. 2006. What comes after the semantic web – pads implications for the dynamic web. In *PADS '06: Proceedings of the 20th Workshop on Principles of Advanced and Distributed Simulation*, 55. Washington, DC, USA: IEEE Computer Society.

Tolk, A., and J. Muguira. 2003, September. The level of conceptual interoperability model. Fall Simulation Interoperability Workshop (SISO), Orlando.

van der Aalst, W., A. H. ter Hofstede, and M. Weske. 2003. Business process management: A survey. In *BPM '03*, 1–12: Springer-Verlag Berlin Heidelberg.

van Glabbeek, R. 2001. The linear time – branching time spectrum i: the semantics of concrete, sequential processes. In *Handbook of Process Algebra*, ed. J. A. Bergstra, A. Ponse, and S. A. Smolka, 3–99. Amsterdam: Elsevier.

Vangheluwe, H. 2000, September. DEVS as a common denominator for multi-formalism hybrid system modeling. In *Proceedings of the IEEE International Symposium on Computer Aided Control System Design*, 129–134. Anchorage, Alaska.

W3C 2006, March. Web services description language (WSDL) version 2.0 part 1: Core language. W3C Candidate Recommendation.

Weisel, E. W., M. D. Petty, and R. R. Mielke. 2004, April. A survey of engineering approaches to composability. In *Fall Simulation Interoperability Workshop*. Washington, D.C.

Yilmaz, L. 2004, July–August. Verifying collaborative behavior in component-based devs models. *Simulation* 80 (7–8): 399–415.

Yilmaz, L., and T. I. Ören. 2006, September. Prospective issues in simulation model composability: Basic concepts to advance theory, methodology, and technology. *The MSIAC's M& S Journal Online* 6 (2): 1–7.

Zeigler, B. P. 1984. *Multifacetted modelling and discrete event simulation*. Academic Press.

Zeigler, B. P., H. Praehofer, and T. G. Kim. 2000. *Theory of modeling and simulation*. 2nd ed. London: Academic Press.

ACKNOWLEDGMENTS

This research is supported by the German Research Foundation (DFG) within the context of the projects CoSA (Component based framework for an effective and efficient simulation of agent systems) and MuSAMA (Multimodal Smart Appliance Ensembles for Mobile Applications).

AUTHOR BIOGRAPHIES

MATHIAS RÖHL holds a MSc in Computer Science from the University of Rostock. His research interests are on component-based modeling and agent-oriented simulation. He is currently a research scientist at the Modeling and Simulation Group at the University of Rostock. His e-mail address is <mroehl@informatik.uni-rostock.de>. Web address of his homepage is <www.informatik.uni-rostock.de/~mroehl>.

FLORIAN MARQUARDT holds a Diploma in computational visualistics from the University of Magdeburg. His field of interesst are web services and smart environments. He received a research fellowship of the German Research Foundation (DFG) and is member of the Modeling and Simulation Group at the University of Rostock. His email adress is <fm@informatik.uni-rostock.de>

ADELINDE M. UHRMACHER is an Associate Professor at the Department of Computer Science at the University of Rostock and head of the Modeling and Simulation Group. Her research interests are in modeling and simulation methodologies, particularly agent-oriented modeling and simulation and their applications. Her e-mail address is <lin@informatik.uni-rostock.de> and her Web page is <www.informatik.uni-rostock.de/~lin>.