Component-based Simulation Model Development using BOMs and Web Services (Invited paper)

Farshad Moradi

Division of Command and Control Systems
Swedish Defence research Agency, FOI
S-164 90 Stockholm, Sweden
Email: farshad.moradi@foi.se

Abstract

Modelling and Simulation (M&S) is a valuable tool and provides means for reducing, amongst others, manufacturing and training costs. However, development of simulation models is a multi-disciplinary and time/resource consuming process. Building simulation models through composition and reuse of predefined and already existing validated simulation components is an approach to reduce the associated costs and improving the usability of the models.

The Base Object Model, BOM, is a new standard for defining reusable and composable simulation components. However, BOMs lack the necessary expressive power to ensure semantic matching of simulation components.

In Web Service Composition (WSC), composite services are built by assembling existing services in order to address functionalities required by users. In WSC much emphasis has been on including the semantic aspects of the composition, through among others utilization of the Semantic Web concept.

In this paper we describe a process that has been developed at the Swedish Defence Research Agency (FOI) with the aim to speed up and improve the development of simulation models. This process utilizes the BOM concept and extends it by taking advantage of the techniques used in WSC. We will present our approach and findings based on our implementation of the proposed process.

1. Introduction

Modelling and Simulation is a powerful tool that can be used to support development of concepts, decision support as well as training, studies and analysis. Simulations can help us understand the dynamics of the systems and give us a tool to study them before they are even being developed. Employing simulations early in the design phase of a product life cycle can detect and avoid costly errors in later stages of the development process.

However, development of simulation models can be a time and resource consuming process and involves some initial costs. Although, the benefits are many the initial costs may be discouraging to decision makers and project managers. Beside the initial cost there is also the issue of quality and usability of the simulation models. A simulation model development process involves different phases including, requirement specification, conceptual modelling, design, development, test, verification and validation. The process requires involvement of different actors such as modellers, subject matter experts, validation and verifications experts and end users. Handling the issue of quality and usability gets more difficult as simulation models get larger and more complex.

An approach to reduce the costs associated with the process and improve the usability of the developed model is to reuse predefined and already existing validated simulation components [18]. Using this method the simulation model is built in a component-based fashion instead of developing it from scratch. Through reusing these validated simulation components we will be able to reduce the costs of development and validation of simulation models. The component-based concept has been successfully deployed in manufacturing, hardware and software industry. Today cars are built in one company using parts and pieces developed by other companies. These parts are probably used in building other types of cars as well. The companies which develop these parts can minimise the development cost through mass production. This in turn helps car manufacturers to reduce the production costs significantly.

Similar methods are being used in software development process through employment of object-oriented methodology and techniques. Some examples are programming languages such as Java and C++, programming environments such as, JBuilder, Visual Studio and NetBeans, and distributed programming architectures, such as Enterprise Java Beans (EJB) and

Proceedings of the First Asia International Conference on Modelling & Simulation (AMS'07)
0-7695-2845-7/07 $20.00 © 2007 IEEE

CORBA (Common Object Resource Broker Architecture) [19].

Although component-based model development is a desired method for improving the model development process and reducing the development costs, it is not trivial and there is no system today which manages to fully employ and make use of this method.

2. Composability

Composing sub-models in order to build new models raise the non-trivial issue of composability. Composability is the capability to select and assemble reusable simulation components in various combinations into simulation systems to meet user requirements [8], [17].

A composable simulation component is a software element of a simulation model with well-defined functionalities and behaviours that conforms to a component model and can be independently deployed, and is subject to third-party composition with or without modification and conforms to a composition model. There are two main types of composability, syntactic and semantic. Syntactic composability is concerned with the compatibility of implementation details, such as parameter passing mechanisms, external data accesses, and timing mechanisms. It is the question of whether a set of components can be combined [27], [6]. Semantic composability, on the other hand, is concerned with the validity of the composition [8].

Within a single organisation, application-oriented and domain-specific model components are already implemented, and syntax specifications for model components for composition are available [25]. Hence, There have been some significant achievements in syntactic composability both within software engineering and simulation communities, but semantic composability is a much harder problem [18], [19] and has inspired many researchers to conduct both theoretical and experimental research.

PACC is an initiative at the software Engineering Institute at CMU to predict runtime behaviour of a composed component [4]. DEVS (Discrete Event System Specification) is a formalism, introduced by Bernard Zeigler et al [2] and [3], for component based model development. Petty et al [17] have developed a mathematical theory for composability, where a model is defined as a function, a simulation is viewed as execution of a function, and model composition is seen as a composition of functions. Some researchers have tried experimental approaches to simulation composability. For instance, OneSAF (One Semi-Automated Forces) is a High Level Architecture (HLA) compliant simulation framework targeted at military simulations, allowing composition of entity level (e.g., soldier, platoon, tanks, etc) models, as described in [20]. HLA is the most widely used architecture for distributed simulations today [31]. HLA provides a simulation environment and standards for specifying simulation parts via Simulation Object Models (SOMs) and interactions between simulation parts via Federation Object Models (FOMs). An HLA simulation is named Federation, which is composed out of Federates, or simulation parts. These parts are all specified by the SOM and FOM documents, and are executed by a Runtime Infrastructure called RTI. The problem with HLAs current standards for formalizing how a federate functions (by the use of a SOM) and how interaction between federates take place within a federation (by the use of a FOM) is that they contain only enough information for an underlying runtime implementation to assure each federate is behaving properly. Although a well-versed federate developer might be able to read a FOM or SOM and deduce how it works, neither of the formats were created to contain semantic information about what they intend to simulate. Hence, HLA provides interface specifications and rules which only ensure syntactic composability [31], and there is little support for semantic composability. The simulation community has recently formulated a standard, the Base Object Model (BOM), to ease reusability and composability [35].

In this paper, we investigate how BOMs can be efficiently used to develop simulation models in a component-based fashion. This work is a continuation of our previous work on a process for component-based simulation development using BOMs [9]. Here we have explored utilization of Semantic Web and Web Service (WS) technologies for further refinement of the process by improving the semantic composition of BOMs

3. BOM

A BOM is fundamentally an XML document that encapsulates the information needed to describe a simulation component. BOMs are structured into four major parts [11], [28], [35]. The first part is the Model Identification, where metadata about the component is stored. This part includes Point of Contact (POC) information, as well as general information about the component itself – what it simulates, how it can and has been used as well as descriptions aimed towards helping developers find and reuse it. The second part is the Conceptual Model. The Conceptual Model contains information that describes the patterns of interplay of the component. This part includes what types of actions and events that take place in the component, and is described by a pattern description, a state-machine, a listing of conceptual entities and events (Conceptual entities and events correspond to how real-world objects and phenomena are modelled in the simulation. For more information see the details on the Conceptual Model part of BOMs.) - together describing the flow and dependencies

of events and their exceptions. The third part is the Model Mapping where conceptual entities and events are mapped to their HLA Object Model representations. This part essentially bridges the Conceptual Model with the HLA Object Model that is described in the fourth part of the BOM, the HLA Object Model. The fourth part, the HLA Object Model, contains the information that is found in a normal FOM/SOM – objects, attributes, interactions and parameters - and should conform to the HLA OMT. There are two additional parts in the BOMs that are not mentioned in-depth in this paper, namely Notes and Definitions. These two parts contain semantic information about events and entities as well as actions that are specified in the Conceptual Model, and are used to provide a human readable understanding of the patterns described in the BOM, figure 1.

BOMs were created based on ideas from HLA, meaning they are formed to include HLA Object Model information. However, BOMs may be used without this information to describe any type of simulation component, a feature which makes BOMS even more versatile.

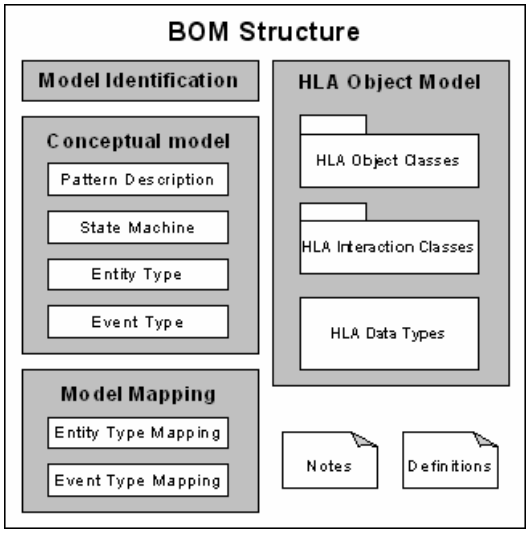

Fig. 1. BOM Structure.

4. Semantic Web and Ontology

In order to compose a simulation out of components, the components need to contain information about their internal structure and how they can be used. This information is called meta-data and contributes to simplified use of a component by others [16]. Generally, the concepts and terminologies used in various components may vary substantially and thus can lead to misunderstanding.

Ontology is used to help creating a common understanding among components and to improve communication among them. In a computer science context, ontology is a description of terminologies and frames of references between entities that interact with each other. Thus, ontology creates a shared understanding of entities and events and contributes to reaching an agreement on meanings of what is communicated between the components. In the Semantic Web initiative, Ontology can be described by the Resource Description Framework (RDF) and Web Ontology Language (OWL), as explained in [38] and [39]. In [13], the authors have investigated the use of ontologies for discrete event simulation and modelling, and have proposed a prototype OWL-based ontology.

5. Web Services

The current Web is a distributed source of information. Introduction of Web Services (WS) extends the Web to a distributed source of services. According to IBM [32], *Web Services are self-contained, modular applications, accessible via the Web through open standard languages, which provide a set of functionalities to businesses or individuals*. This definition is somewhat unclear and a better definition is provided by the World Wide Web consortium (W3C):

A Web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML-based messages conveyed by Internet protocols. [37]

WSs are designed to provide interoperability between diverse applications, i.e. the platform and language independent interfaces of WSs allow the easy integration of heterogeneous applications.

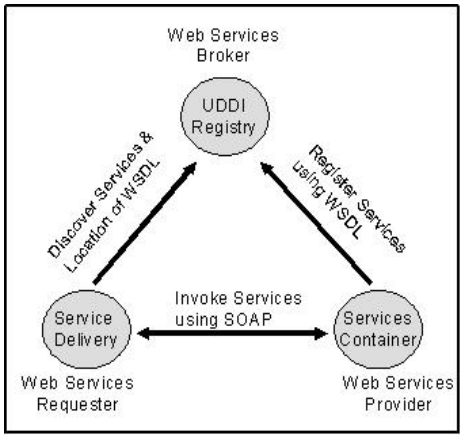

Fig. 2. Web Service roles and operations.

A service description is a standard, formal XML notation that is used in order to describe a WS. It provides all the necessary information to interact with the WS,

including message formats, location and transport protocols. The descriptions are expressed in Web Service Description Language (WSDL). A WS and its definition is created by a service provider and then published with a service registry based on a standard called the Universal Description, Discovery, and Integration (UDDI) specification. A service requester may find a published service via the UDDI interface. The UDDI registry provides the service requester with a WSDL service description and a URL (Uniform Resource Locator) pointing to the service itself. The service requester may then use this information to directly bind to the service and invoke it, as illustrated in figure 2.

5.1. OWL-S

OWL-S is based on Web Ontology Language and aims at establishing a framework for describing Web Services in the context of the Semantic Web. OWL-S is an extension of the DARPA Agent Meta Language for Services (DAML-S). It has been developed to provide support for discovery, composition, invocation and interoperation of Services. OWL-S consists of three parts: the service *profile*: "for advertising and discovering a service", the service *process model*: "for detailed description of a service operation", and finally *grounding*: "for describing how to interoperate with a service", fig. 3.

There are two constraints, a *service* can be described by at most one *service model* and a *grounding* must be associated with exactly one service. There is no restriction for service *profile* and service *grounding*; in fact it is very useful sometimes to have multiple service *profile* and/or service *grounding* [33].

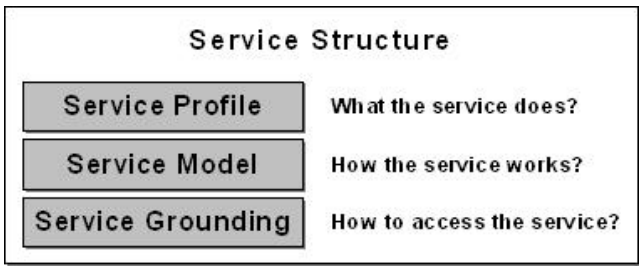

Fig. 3. OWL-S structure.

5.2. Web Service Composition

One of the biggest challenges regarding WS is developing automatic or semi-automatic techniques for discovery and composition of those services [12]. It is essential to find a way to utilize these key features together to convert the Web into a distributed source of computation. Various techniques and approaches have been presented by different researchers such as: template-based techniques [10], interface-based techniques [12], logic-based techniques [23], ontology-driven techniques [41], quality-driven techniques [15], automata-based techniques [24] and Petri net-based techniques [22], [26].

The main goal of these techniques is to provide automatic ways to discover and reason about combination of services. They often apply software synthesis and composition methods to WS composition, which first of all requires a compiler that is able to translate the web service description language, e.g. WSDL, into formal logic or other formal component description language, a synthesis mechanism which automatically selects, adapts and composes WSs, and a manager that invokes the WSs and transfer data between them

5.3. Web Services and Simulations

Simulation components can be seen as a special case of WSs. As described earlier a simulation component (SC) is a software element of a simulation model with well-defined functionalities and behaviours as is the case with WSs. A SC conforms to a component model and can be independently deployed. A WS has public interfaces and bindings are defined and described using XML and can deliver a service. A SC is subject to third-party composition with or without modification and conforms to a composition model. WSs can be composed together and aggregated to deliver functionalities according to user requirements. There are well-adapted standards for publication, registration and discovery of WSs, while in the case of SC the M&S-community has just started to set up standards. These standards are not generally adapted. Here, there is a potential and the M&S-community has the opportunity to take advantage of the progress and experiences gained by the WS-community. This potential has been pointed out by among others [21]. However, the emphasis there was mainly on simulating WS processes for the purpose of correcting/improving the design.

6. Component-based model development using boms

Network-based Modelling and Simulation (NetSim) is an ongoing project at the Swedish Defence Research Agency (FOI), which aims to provide simulation modellers and developers with a set of services to improve the simulation development and execution process [7]. Component-based simulation model development (CBMD) is one of these services which the NetSim project focuses on. The objective here is to identify and develop methods and techniques for implementing a framework for CBMD. For this purpose we studied the BOMs standard [35], focusing on the following BOM capabilities, BOM composability and BOM-based model development.

6.1. A general framework

Since BOMs contain higher-level information and have been designed to support reuse, via the Metadata contained in them [1], it was found interesting to investigate how BOMs *compatibility* and *comparison* can be automated. The ideal procedure would be that first a model developer formulates the intent of a simulation in a high-level language such as the Simulation Reference Markup Language (SRML), [36]. In the next step all matching BOMS within a repository are identified and a subset of the matching BOMs are combined into a new BOM (so called BOM assembly). Finally, the new BOM assembly is checked for validity.

Given a BOM assembly one could automatically create code-skeletons for federates and a whole federation. As BOMs contain high-level information as well as HLA OMT information it might be possible to reduce the time needed to develop code that is not included in the actual simulation logic by developing tools that use BOM information to automate the generation of such code.
Another approach would be to include executable programs associated with each BOM in the repository, called BOM-implementations. In this way, the BOM-implementations could also be assembled in the same manner as the BOMs within a BOM assembly and create an executable simulation model of the composition.

6.2. Model composition

A process model for CBMD, was developed based on the BOM concept. The idea was that a simulation developer has a vision of a simulation. This developer condenses the simulation idea down into simulation parts and components, and specifies these in a SRML document. After this, the process of finalizing the simulation is made up out of three parts: Discovery, Matching and Composition (abbreviated as DMC).

The first activity is BOM Discovery, which is broken down into two sub-activities. The first phase is to process the provided SRML document and identify the components (BOMs) that are needed in the simulation. It is done by parsing out all the components mentioned in the SRML document, along with an Ontology that accompanies the SRML document and states its frame of reference. The second activity is to utilize Ontology-information to fetch the identified BOMs from a repository. The BOMs fetched should be relevant to the simulation in some way, either as explicit components or components related to the simulation in some other way (loggers, rendering components etc). How this fetching is done depends on how the BOM Repository is constructed and how the interface to it is implemented. The search should be done based on Ontology information to ensure that the components and keywords describe what they intend to describe. The listed BOMs are then fetched and used as input to the BOM Matching activity. This activity identifies BOMs only on a very high level. BOMs, that roughly fit the intent of the simulation or matches the components specified in the simulation document, are simply fetched from the repository.

A finer selection of BOMs actually used will be done in the BOM Matching phase. The second activity, BOM Matching, compares the fetched set of BOMs and decides which BOMs might be suitable for the simulation. This is a more complex activity that needs to take into account the simulation intent (as described in the SRML document). One has to handle issues such as, what components fit together semantically and practically and how it is done. In order to compare BOMs, additional information such as ontologies and reference documents will also be used in this activity.

The activity's goal is to, given a set of BOMs and a SRML document describing a simulation intent, map up the entire simulation and fit in components so that the simulation can be composed. This is similar to how a puzzle is solved; the SRML document describes the finished puzzle, and BOMs provide pieces of the puzzle that can be used to complete it. The BOM Matching activity is the placement of pieces so that the puzzle is solved. The BOM Matching activity idea is to use the SRML document as a mapping of components and interaction between them. This mapping could then be analyzed for matching compatibility using the metadata that is available inside the BOMs, coupled with Ontology information. A further breakdown of this activity can be seen in figure 4. In this figure, the first step is to make a simulation mapping of the components described in the SRML document. The next step is then to create a number of permutated mappings using the BOMs that were found in the BOM Discovery step, and then analyze each BOM mapping to check for compatibility.

In the first version of our implementation the actual compatibility check used all the available meta-data inside the BOMs and reference documents such as Ontologies to create a Compatibility Score between a pair of BOMs [9]. This score indicates if the two BOMs refer to the same types of events and entities, and if they publish or subscribe to events that have been specified in the SRML document. The score is also affected by previous use history, overlapping application domain and other general metadata, which has been used to reinforce compatibility scores between BOMs.

In our second approach we instead of only including ontology descriptions as reference documents, describe entire BOMs in OWL-S. That means using OWL-S as an infrastructure for describing BOMs. This way a BOM is seen as a service.

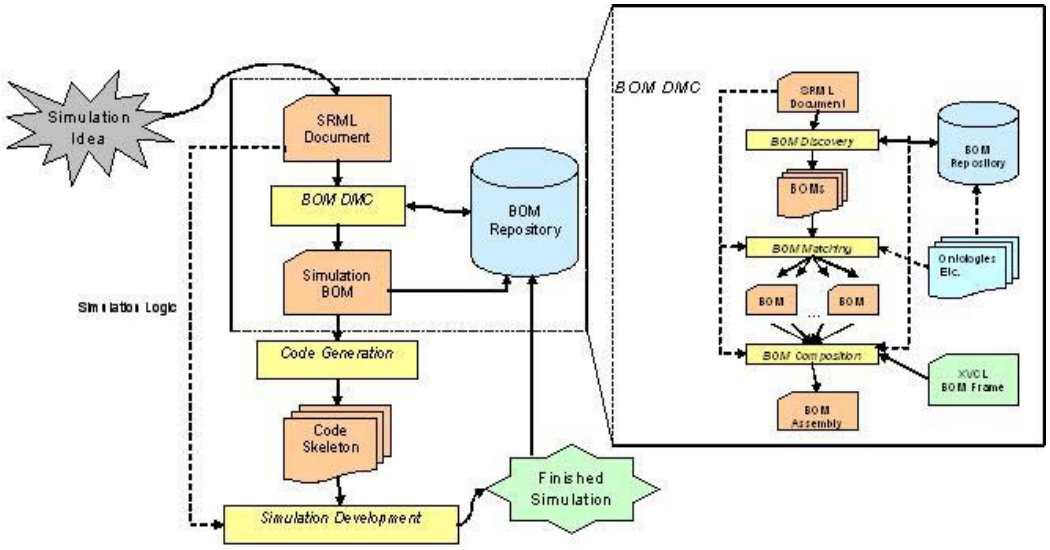

Fig. 4. BOM Discovery, Matching and Composition process

It seems only logical to look at BOMs as simulation services. This way, we are able to include ontology in a more natural way and also reason about semantic aspects of BOM compositions by taking advantage of experiences and techniques developed in WSC. We can also utilize inference engines to reason about syntactic and semantic composability of BOMs. Describing simulation components as WSs also increases the availability of those components through Web interfaces and improves the discovery phase of our process. A challenge here has been to map BOM descriptions into OWL-S documents.

Lastly the selected components are assembled into a composite BOM, a BOM Assembly, in the BOM Composition activity. The BOM Composition step would take a number of BOMs as input, as well as the SRML document and mapping data to determine how to merge the BOMs together (as parsed out from the Mapping Suitability step). This information is used to create a BOM Assembly. This BOM Assembly is later intended to be used to either produce code for the execution of the simulation, or serve as a blue-print for composing BOM-implementations associated with the identified BOMs.

The DMC procedure is visualized in Figure 4, where each of the three phases is shown in square boxes, and is further discussed in the subsequent sections.

6.3. Mapping between BOMs and OWL-S

The purpose of mapping BOM descriptions into OWL-S is to first of all use OWL (Web Ontology Language) as the underlying language for describing BOMs and also utilize language features of OWL-S to improve the semantic expressiveness of BOMs and hence facilitate semantic discovery and compositions of BOMs.

As mentioned earlier BOM consists of three main parts, model identification, conceptual model and model mapping. The model identification part can be mapped into the service profile part of the OWL-S without any major adjustments. These two parts contain more or less identical information. The main difference is that the service profile contains information about the functionality of services, such as input, output, parameters, preconditions and results. This information can be obtained from the conceptual model part of BOMs.

Mapping between the conceptual model and the service profile is not straight forward and requires a deeper understanding of how simulation models and services work.

The main part of BOM conceptual model is “Pattern of Interplay” which includes sequence of “Pattern Actions”. Each Pattern Action is a reference to state behaviour of a conceptual entity. In the figure 5, the top-level of OWL-S Service Model is constructed based on the concept of “Process”, the aim is to make it easier to understand and interact with different services. In order to map BOMs on OWL-S we assume that the BOM Pattern of Interplay to be replaced by a single Service Model, and hence each BOM Pattern of Action to be replaced by a Process. In order to construct a Process-tree for representing sequence of Pattern Actions, we use composite processes, to better define the state behaviour of each conceptual entity if the action is done in a multi-step manner; or we use several atomic processes defined by OWL-S flow, if the actions are done independent of each other. Finally, the state behaviour of each conceptual entity which is defined in BOM Conceptual Model as several State-Machines could be defined as several Control Constructs, Figure 5.

The grounding part contains information about the implementation of a WS. It is left out at this stage, since it

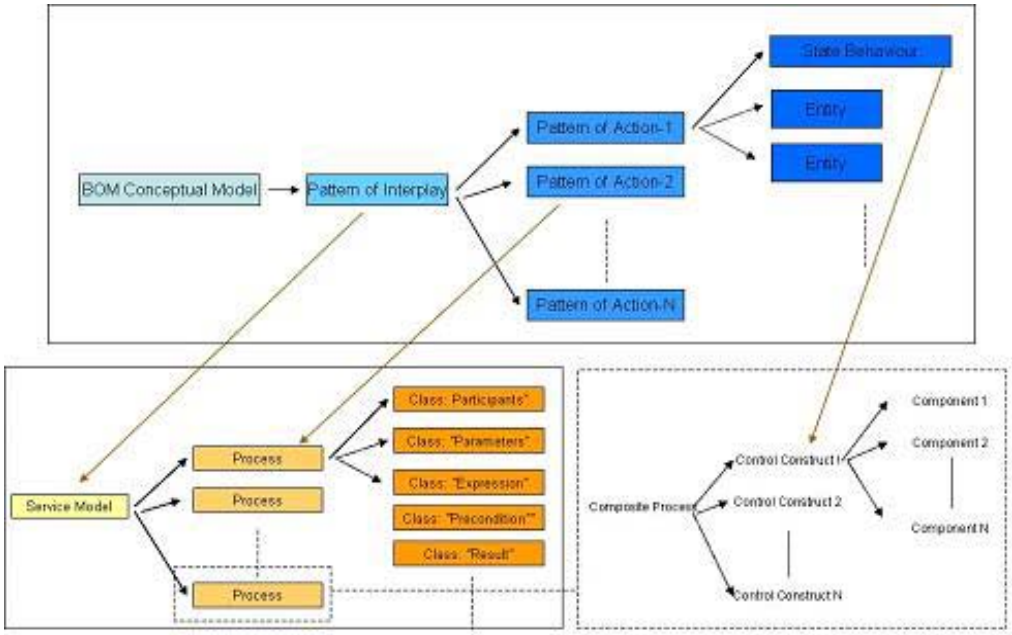

Fig. 5. Mapping between BOM conceptual model and OWL-S service model.

does not have any impact on the matching phase of the BOMs.

7. Conclusions and future work

In this paper we have presented our approach to model composition using BOMs and Web Services. The conclusion is that BOMs contain a great deal of information that significantly support a component based simulation development process. This is due to the fact that BOMs contain numerous useful metadata and definitions of events and entities. We have also pointed out that inclusion of semantic definitions and frame of reference (via ontologies) is essential. However, BOMs do not specify that this type of information should be included in any predefined way, which is something that would be interesting to review in upcoming versions of the BOM standard. Our proposed approach for inclusion of semantic definitions is to utilize OWL as the language for describing BOMs. And in order to improve the semantic expressiveness and provide semantic discovery and composition of BOMs we have utilized OWL-S as the framework for describing BOMs. Our preliminary results show that it is possible and feasible to map BOM descriptions into OWL-S. However, more experiments are required in order to draw general conclusions. Thus, future work includes further examining to what extend Web Services can improve the BOM standard in order to automate semantic discovery and the composition of simulation models.

8. References

[1] Bachman, J., Gustavson, P., Lutz, R., Scrudder, R., Understanding the BOM Metadata and Making It Work For You, in *Proceedings of 2005 Simulation Interoperability Workshop*, 2005.

[2] Bernard P. Zeigler, Herbert Praehofer, and Tag Gon Kim. Theory of Modeling and Simulation, *Academic Press*, 2nd edition, 2000.

[3] Bernard P. Zeigler and Hessam S. Sarjoughian. Introduction to DEVS Modeling and Simulation with JAVA: Developing Component-Based Simulation Models, *Arizona Center for Integrative Modeling and Simulation, University of Arizona and Arizona State University,* Tucson, Arizona, USA, January 2005.

[4] Carnegie Mellon Software Engineering Institute, *Predictable Assembly from Certifiable Components (PACC)*, http://www.sei.cmu.edu/pacc/.

[5] S. Chandrasekaran, G. Silver, J.A. Miller, J. Cardoso and A.P. Sheth, "Web Service Technologies and their Synergy with Simulation," *Proceedings of the 2002 Winter Simulation Conference*, San Diego, CA, December 2002.

[6] C. Szabo, Y.M. Teo, An Approach to Syntactic Composability and Model Reuse in Composable Simulation, Department of Computer Science, *National University of Singapore, Technical Report*, September 2006.

[7] Eklöf M, Ulriksson J, Moradi F, NetSim – A Network Based Environment for Modelling and Simulation, *NATO*

Modeling and Simulation Group, Symposium on C3I and M&S Interoperability, Antalya, Turkey, (2003).

[8] E. W. Weisel, M. D. Petty, and R. R. Mielke, "Validity of Models and Classes of Models in Semantic Composability", in *Proceedings of the Fall 2003 SIW*, Orlando FL, Sept 14-19 2003.

[9] Farshad Moradi, Rassul Ayani, Peder Nordvaller, "Simulation Model Composition using BOMs",in *Proceedings of The 10-th International Symposium on Distributed Simulation and Real Time Applications, DS-RT '06*, October 2006.

[10] F. Casati, S. Ilnicki, L.-J. Jin, V. Krishnamoorthy, and M.-C. Shan, "Adaptive and Dynamic Service Composition in eFlow", *Proc. of the Intl. Conf. on Adv. Info. Systems Engineering*, Sweden, 2000.

[11] Gustavson, P., Hancock, J., McAuliffe, M., Base Object Models (BOMs): Reusable Component Objects for Federation Development, in *Proceedings of 1998 Fall Simulation Interoperability Workshop,* 1998.

[12] I. B. Arpinar, R. Zhang, B. Aleman-Meza, and A. Maduko., "Ontology-driven Web Services Composition Platform", *IEEE Intl. Conf. on e-Commerce Technology*, San Diego, California, July 6-9, 2004.

[13] J. A. Miller, P. A. Fishwick, Investigating Ontologies for Simulation Modeling, *Proceedings of the 37th Annual Simulation Symposium*, pp. 55-63, 2004.

[14] J. A. Miller and Gregory Baramidze, *Simulation and the semantic web*, University of Georgia, US, 2005.

[15] J. Cardoso, *Quality of Service and Semantic Composition of Workflows*, Ph.D. Dissertation, Dept. of Computer Science, Univ. of Georgia, Athens, GA, 2002.

[16] K. Morse, M. Petty, P. Reynolds, W. Waite, P. Zimmerman, Findings and Recommendations from the 2003 Composable Mission Space Environments Workshop, 2004.

[17] M. D. Petty, E. W. Weisel, R. R. Mielke, Overview of a Theory of Composability, *Virginia Modeling Analysis & Simulation Center,* Old Dominion University, 2004.

[18] Paul K. Davis and Robert H. Anderson, "Improving the Composability of DoD Models and Simulations", *JDMS: The Journal of Defense M&S: Applications, Methodology, Technology*, Vol1, Nr 1, January 2004.

[19] Robert G. Bartholet, David C. Brogan, Paul F. Reynolds, Jr., Joseph C. Carnahan, "In Search of the Philosopher's Stone: Simulation Composability Versus Component-Based Software Design", in *Proceedings of the 2004 Fall SIW,* Orlando, FL, September 2004.

[20] Robert L. Wittman Jr. and Cynthia T. Harrison. Onesaf: A product line approach to simulation development, Technical report, *MITRE Corporation and US Army Simulation, Training and Instrumentation Command,* 2001.

[21] S. Chandrasekaran, G. Silver, J.A. Miller, J. Cardoso, A.P. Sheth, "Web Service Technologies and their Synergy with Simulation," *Proceedings of the 2002 Winter Simulation Conference*, San Diego, CA, December 2002, pp. 606- 615.

[22] S. Narayanan and S. Mcllraith. Simulation, "Verification and Automated Composition of Web Services", *Proc. of the 11th Intl Conf. on WWW*, Hawaii, 2002.

[23] S. R. Ponnekanti, and A. Fox, "SWORD: A Developer Toolkit for Web Service Composition", *Proc. of the 11th Intl Conf. on WWW*, Hawaii, 2002.

[24] T. Bultan, X. Fu, R. Hull, and J. Su, "Conversation Specification: A New Approach to Design and Analysis of E-Service Composition", *Proc. of WWW Conference*, 2003.

[25] Weisel, E. W., M. D. Petty, and R. R. Mielke. "A Survey of Engineering Approaches to Composability", in *Proceedings of the Spring 2004 SIW*, Arlington VA , April 18-23, 2004.

[26] X. Yi and K. Kochut, "Process Composition of Web Services with Complex Conversation Protocols: a Colored Petri Nets Based Approach", *Proc. of Design, Analysis, and Simulation of Dist. Sys. Symposium*, 2004.

[27] Y. Hu, G. Tan, F. Moradi, "Automatic SOM Compatibility Check and FOM Development", in *Proceedings of 7th IEEE Distributed Simulation and Real-time Applications,* Delft, The Netherlands, October 2003.

[28] BOM, http://www.boms.info.

[29] BOMworks, http://www.simventions.com/bomworks/.

[30] DEVS standardization group, http://www.sce.carleton.ca/faculty/wainer/standard/.

[31] HLA at DMSO, https://www.dmso.mil.

[32] IBM Web services tutorial. Online : http://www-106.ibm.com/developerworks/webservices/.

[33] OWL-S: Semantic Markup for Web Services, http://www.w3.org/Submission/OWL-S/.

[34] Semantic Web, http://www.w3.org/2001/sw/.

[35] Simulation Interoperability Standards Organization (SISO), *Guide for Base Object Model (BOM) Use and Implementation*, SISO-STD-003.0-DRAFT-V0.11, SISO, 2005.

[36] SRML, http://www.w3.org/TR/2002/NOTE-SRML-20021218/.

[37] W3C. Web services architecture requirements. http://www.w3.org/TR/wsa-reqs.

[38] World Wide Web Consortium, *Resource Description framework*, http://www.w3.org/RDF/.

[39] World Wide Web Consortium, *Web Ontology Language*, http://www.w3.org/2004/OWL/.

[40] World Wide Web Consortium, *Simulation Reference Markup Language,* http://www.w3.org/TR/2002/NOTE-SRML-2002121.

[41] WSDL-S, W3C Member Submission on Web Service Semantics, http://www.w3.org/Submission/WSDL-S/.