Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, eds.

SERVICES MODELING AND DISTRIBUTED SIMULATION DEVS / HLA SUPPORTED

Gregory Zacharewicz
Thècle Alix
Bruno Vallespir

Laboratoire IMS UMR CNRS 5218 Groupe Productique (GRAI)
University of Bordeaux
351, cours de la Libération, 33405 Talence cedex, FRANCE

ABSTRACT

Nowadays, Service design and development is more than ever key challenge for enterprises. Nevertheless, the set up of new services activities is not entirely formalized and guided by a clear methodology or commonly recognized standard. These facts can lead on wrong services definitions and practices that penalize the enterprise, in particular in the case this evidence arrive late. The idea introduced in this prospective paper is to model and simulate services before realizing them in order to validate desired properties and anticipate wrong behavior. Considering this proposal, a problem appears: no whole part of the service (actors, software, machines) can be included in the model, real actors are wanted to act in the loop. In addition, the simulation requires to interoperate and synchronize with heterogeneous and distributed actors of the service. The goal of the paper is to present a Service Modeling Environment based on a graphical Service Modeling Specification language and on Distributed DEVS Simulation. Having selected the essential concepts in the elaboration of Service Modeling, we present a description language to define Service Modeling processes. Then, we describe a method to transform Service Modeling specifications into simulation models; DEVS formalism is chosen for its formal properties. Based on the experience of distributed simulation, we propose to address the interoperability by conforming to the distributed simulation standard HLA. Finally, we define a distributed Service Modeling Environment that interfaces components of the Service Models and others actors in HLA compliant Federation.

1 INTRODUCTION

Since the fast growing of service field, the idea to model this concept, to try at formalizing and managing the definition of new services, is becoming significant consideration. Nevertheless, the Service Modeling does not provides one common and exclusive standard for developing frameworks to manage services processes. The specification of a Service Modeling may involve different (i.e. heterogeneous) applications and / or actors which are essential to its execution. The specification of standards are numerous, we propose to chose the graphical definition of the Service-Oriented Modeling Framework (SOMF) (Bell 2008). The reason is the user friendly design, it does not require expertise on service specification and on programming code (it can be eventually generated from a graphical specification). The problem is the correctness checking or the scenario evaluation on the service models. Concretely, this field is missing Simulation methods, we believe that simulation can give predictive information on the correctness of the models. In detail, an identified gap is the SOMF does not define one simulation semantics associated to the Service Modeling. This fact may lead to difficulties to determine if simulation result errors have to be imputed to a wrong model or to an incorrect implementation of the Service Modeling engine.

We propose to describe Service Models in a formal approach and to validate these models by simulation. DEVS, State Charts, Petri nets are well-known formalisms to describe the behavior of complex systems. DEVS clearly separates modeling and simulation processes to be more flexible and general. In fact Artificial Intelligence techniques can be combined with DEVS. Also, DEVS privileges the use of two concepts event and timed state. However Service Modeling users are not familiarized with DEVS. Thus we propose to transform automatically a Graphical Service Modeling Specification into a DEVS model. These rules prevent the user from learning DEVS.

In addition, actual complex industrial service processes are composed of heterogeneous and distributed resources (material, immaterial and human in the loop). To address these requirements of interoperability, we introduce a Service Modeling Environment HLA compliant.

The paper is organized as follows. Section 2 & 3 give an overview of Service, Service Modeling, DEVS and HLA. Section 3 illustrates the proposed environment and the approach to transform a Service Modeling specification into DEVS. Section 4 and 5 proposes a new Service Modeling and Simulation Environment HLA compliant. Finally, we conclude with exposing our future works and conclusion.

2 SERVICE OVERVIEW

2.1 Definitions and characteristics

Many suggestions stemming from different points of views have been made by authors to define the concept of service, and its characteristics. The general understanding defines a service as a free or charged advantage proposed to an individual, a company or a public or private organizations, its objective is to create changes or transform some entities regarding their form, place, time or availability. Entities can refer to material objects, goods, people, the natural environment or symbolic representations. Afterwards, authors have based the definition of a service on the basis of the role and responsibilities of the service provider and seeker and its contrast with a tangible good. For Hill (1977) "A service may be defined as a change in the condition of a person, or of a good belonging to some economic unit, which is brought about as the result of the activity of some other economic unit, with the prior agreement of the former person or economic unit". On this base, Gadrey (1996) has proposed to define a service activity as "an operation that aims to change the state of a reality C, owned or used by a consumer (customer or user) B, performed by a provider A on the demand of B and most of the time in interaction with him, but that does not lead to the production of a good able to circulate freely economically independently of the support C". Gronroos for his part defined a service as "an activity or series of activities of more or less intangible nature that normally, but not necessarily, take place in interactions between the customer and service employees and/or systems of the service provider, which are provided as solutions to customer problems" (Gronröos,1999). All these definitions have common input among which: (i) an intangible or immaterial characteristic, (ii) a contact between a contractor or provider and a customer or entity having a need, (iii) a participation of that customer or entity in the production process and (iv) a definition of objectives stemming from a compromise between the interacting elements that render the output of the service production process quite "fuzzy". Gallouj and Weinstein (1997) summarized the various concepts that turn around the services in order to better understand and analyze them. They concern: an interface (point of contact between customer and service provider), an interaction or co-production (extensive and balanced interaction), a servuction or service delivery system (a process of creating a service by linking up various elements: a customer, a physical medium, a contact personnel and a service, considering environmental elements), and finally a socially regulated service relationship.

The rapid development of service activities in all economic area and the raising of competitiveness led service providers to search for differentiation, effectiveness and innovation. As a result the service delivery system or service production system is been changed to be more industrialized and automated.

2.2 From a service delivery system to an industrial service production system

In (1987), Eiglier and Langeard proposed a representation (see figure 1) and a definition of the service delivery system as a systematic and coherent organization of all the physical and human elements coming from the customer/enterprise interface that are necessary for service delivery according to predetermined commercial characteristics and quality levels of appreciation.

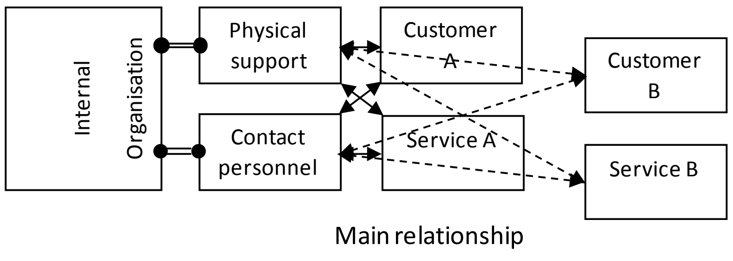

Figure 1: Service delivery system

The five elements required to deliver a service are the customers, the contact personnel, the physical support, the service and the internal system or back office of the organization. The customer is the consumer implied in the production of the service; that without whom the service does not have an existence. It is responsible mainly for the quality of the service of which he is the sleeping partner. The contact personnel represent the personnel inside the company whose work is to be directly in relation with the customer in order to concretize the act of service. These elements are considered as cctors or resources of the service process in the following. The physical support consists of all the material elements necessary to the realization of the service. The physical support can be split in two categories: the tools that are necessary for the service achievement (objects, furniture, machines...) and the material present in the environment in which the service happen (office, scenery...). It can be used by the customer or by the contact personnel. The customer is there mandatory for service existence while the contact personnel can be replace by material. The service is at the same time the objective and result of the relationship between the four previous elements. It represents the benefit that might satisfy the customer 'requirements. This benefit is a part of the value that the customer associates to the service. The back office gathers the main economic functions of the company such as financial, marketing, human resource management, etc as well as specific functions necessary for the service realization.

The service delivery system above mentioned was mainly addressed in comparison with the product production process. Since competitiveness has increased, that representation as evolved to a more industrialized process where sub-processes corresponding to part of a service can be automated and gathered to gain in reactivity.

2.3 Service modularization

For a long time, a service has been considered as unique and heterogeneous because the conditions of the relation relationship where supposed to be unique depending on the state of mind and behavior of the customer, the contact personnel and the interaction and also because it was thought to be highly customized. Customization allows to be aware of value for the customer and to propose a solution that might fulfill its requirements. Service industrialization and innovation and service value analysis leads to control customization. Service industrialization leads to the search for process modeling tools dedicated to services and management or simulation tools to manage service operations. Service innovation mainly impacted by the development of ICT gives the possibility to automate some parts of a service in back office; sort of modules that can be gathered in front office according to the relationship. Finally, the service value and coherence analysis has make it obvious that the behavior of the customer and the contact personnel is comprised in a sort of template and that even if there exists service variants, the possibilities around a service are not infinite. In the following parts of this paper, we will analyze how can service delivery system and service process be modeled and simulated using Service Modeling and DEVS / HLA in order to run distributed and heterogeneous Simulation.

3 MODELLING AND SIMULATION RECALLS

3.1 Service Modeling

In our point of view, Service Modeling must define a set of activities (which compose a Service) and its interrelation with resources involved in the service process, i.e. client and actors (contact personnel, physical support...). the activities can consist of sub services, named also atomic Services, composed services or clustered services, and logical expressions or controllers that describe the roads of information flow that describe the status of the in-progress service. Bell (2008) (with Oriented Modeling Framework) and OMG (with Service Modeling Language) have proposed standards in the field of Service Modeling in association with the main actors of the domain. Literature broadly describes Service Model with graphical specifications in which sub services are represented with geometrical forms, controllers with nodes and the flow of information over sub services by arrows; we believe it is closed to process or workflow representation concepts (Zacharewicz 2008).

3.1.1 Graphical Service Modeling

One of the Graphical Service Modeling more commonly admitted is the Service-Oriented Modeling Framework (SOMF) that introduces three major service components. For largely adopted representation reasons, we will follow and enrich it in our approach These structures are entities that consistently comes from the computing environments description (Bell 2008):

Atomic service *An indivisible entity that can constitute a service process. An atomic formation is typically a single concept not subject to decomposition analysis activities and its business or technological functionality does not justify breakdown to smaller components during service life cycle. E.g. A customer checking bank account balance service.*

Composite service *A composite service structure aggregates fine-grained services and smaller services which contribute collectively a technical or business solution. A composite service can aggregate consisted services the atomic or other and/or other composite services. E.g. Basically, an application that is composed of sub-components, a customer bank checking which aggregates smaller checking account and savings account services, a weather simulation system that is composed of routing, synchronization, several simulators and data transformation components.*

Service cluster *A set of distributed and related services that are gathered because of their mutual business or technological commonalities. A service cluster both affiliates services and combines manage and synchronize their capacities to provide a business for an enterprise. A cluster structure can federate atomic as well as composite services. E.g. A Mutual bank service cluster that is composed of interoperating and distributed mutual finances services. A Customer Support service cluster that offers automated bill payments, online account statements, and money transfer capabilities.*

The SOMF also introduces a simple notation to describe the three major service formations, as illustrated in the Figure 2.

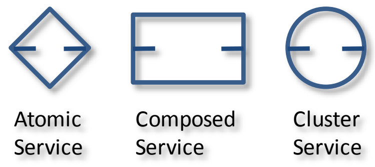

Figure 2: Service-oriented analysis notation

Bell (2008) completes service-oriented description and analysis notation by proposing eight simple icons, each of which enables us to operate on a service (atomic or composite service) or on a group of services (service clusters) in different ways. These operations promote core SOA best practices. The Figure 3 depicts the analysis operations relations, they are employed to model the links between SOA Services.

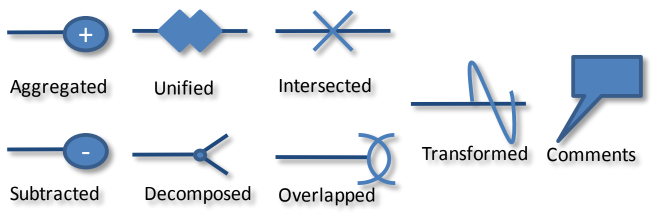

Figure 3: Service-oriented analysis notation

The limitation of this representation comes from the fact that it is not necessarily based on strong formal concepts. Thus, it does not allow properties of semantic correctness verification and validation of the model. Furthermore, these models are rarely simulated and consequently could not be straightforwardly compared in terms of efficiency regarding to others. One solution is to use or to define a unified language for the specification of Service Modeling in order to be applied as a common output of Service Modeling editors. This language will support algorithms to transform a Service Modeling model into a classical formal specification for simulation, regardless of the Service Modeling editor.

3.1.2 Service Simulation

Several environments for Modeling at large "Services" coexist but few are proposing simulation facilities. However, one interesting solution is proposed in Sarjoughian (2008) works. In these works, an SOA-compliant DEVS (SOAD) simulation framework is proposed for modeling service-oriented computing systems. The environment is basically based on DEVS formal modeling, so they do take profits of concepts offered by the formal discrete event simulation theory (Zeigler 2000). In fact, this theory separates the modeling phase from simulation allowing the reuse of the validated specifications to other domains. The models are portable and can be formally checked and compared. We believe that a simulation tool based on the DEVS formalism enhances the simulation and validation processes. B.P. Ziegler (2000) presents DEVS as a general formalism due to the following synthesized facts:

- DEVS approach to model complex systems is modular,
- DEVS integrates naturally time,
- DEVS offers a formal definition of the simulator.

Nevertheless the DEVS models proposed in (Sarjoughian 2008) are limited to computer oriented related services, we believe M&S environments must be extended to feet general definition of service presented in § 2.

3.2 DEVS

B.P. Zeigler (1976) defined the Discrete Event System Specification (DEVS) formalism. A key contribution of DEVS lies in decomposing the traditional transition function into two sub-functions, i.e. internal transition δ_{int} and external transition δ_{ext}. When the elapsed time of a model reaches the life time (noted *lt*) of the current state value, the output function is activated and an internal transition occurs, it captures the autonomous evolution of the model. When an external event occurs, the external transition function is activated.

DEVS coupled models (Zeigler 2000) are structural models defined by interconnection of basic models. Each basic model interacts with other models to produce a global behavior. The basic models are, either atomic models, or coupled models stored in a library. The models are coupled in a hierarchical way. Moreover, in (Zeigler 2000), the concept of abstract simulator is developed. This architecture of simulation is diverted from the hierarchical DEVS models.

Let us focus on Figure 4 that represents two simple DEVS atomic models (Light Actor, Light) using the graphical representation of (Song 1997). The discrete state of the models considered is defined only by the *phase* state variable (represented by circles, e.g. Perceive Dark, Lighted...). For that reason, a lifetime value can be associated to each *phase* value (referenced by numbers inside circles). Solid arcs represent external transitions δ_{ext}.; for instance, mark "*In?OFF*" on an arc of this type describes that the model state will transit by receiving an input event of "*OFF*" value on the input port "*In*". Dotted arcs represent internal transitions dint; if it elapses lifetime length in the source phase of this type of arc, mark "*out!ON*" shows, for instance, that the model state will transit and emit an output event of "*ON*" value on output port "*out*". Triangles represent input and output ports. The two models are connected by coupling relation.

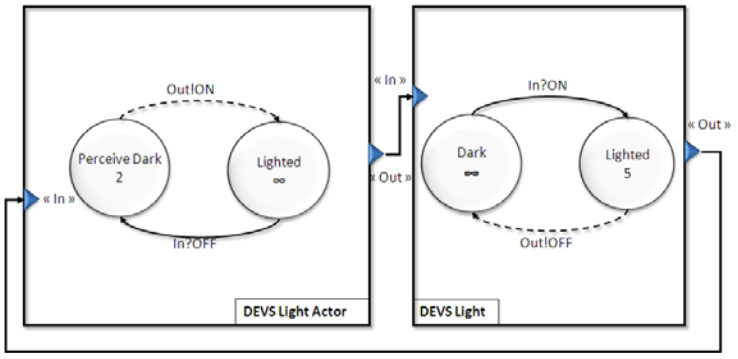

Figure 4: Graphical DEVS Models.

3.3 Interoperability Concepts

Enterprise Interoperability refers to the ability of interactions between enterprise systems. The interoperability is considered as significant if the interactions can take place at least at the three different levels: data, services and process, with a semantics defined in a given business context (IDEAS 2002).

Interoperability extends beyond the boundaries of any single system, and involves at least two entities. Consequently establishing interoperability means to relate two systems together and remove incompatibilities. Incompatibility is the fundamental concept of interoperability. It is the obstacle to establish seamless interoperation. The concept 'incompatibility' has a broad sense and is not only limited to 'technical' aspect as usually considered in software engineering, but also 'information' and 'organization', and concerns all levels of the enterprise (Chen 2007).

In creating a distributed Service M&S environment, we will face the interoperability problems and we will need to solve them through the identification and the removing of barriers (incompatibilities) which prevent interoperability to happen.

3.3.1 Dimension of interoperability approaches

Enterprise Interoperability takes into consideration three admitted approaches to develop interoperability:

Integrated approach *there exists a common format for all models. This format must be as detail as models. The common format is not necessarily a standard but must be agreed by all parties to elaborate models and build systems.*

Unified approach *there exists a common format but only at a meta-level. This meta-model is not an executable entity as it is in the integrated approach but provides a mean for semantic equivalence to allow mapping between models.*

Federated approach *there is no common format. To establish interoperability, parties must accommodate on the fly. Using federated approach implies that no partner imposes their models, languages and methods of work.*

Today, most of the approaches developed are unified ones such as for example in the domain of enterprise modeling, we can mention UEML (Unified Enterprise Modeling Language) and PSL (Process Specification Language) which aim at supporting the interoperability between enterprise models and tools.

Nevertheless, the federated Enterprise Interoperability is the most challenging approach, in particular because few research works have been performed in this direction. The federated approach aims to develop full interoperability and is particularly suitable for an inter-organizational environment (such as networked enterprises, distributed services, etc.). In the Enterprise Interoperability roadmap published by the European Commission, developing federated approach for interoperability is considered as one of the research challenges for the years to come.

3.4 Distributed Simulation System: HLA (High Level Architecture)

The High Level Architecture (HLA) is a software architecture specification that defines how to create a global simulation composed of distributed simulations. In HLA, every participating simulation is called federate. A federate interacts with other federates within a HLA federation, which is in fact a group of federates. The HLA definitions set gave place to the creation of the standard 1.3 in 1996, which then evolved to HLA 1516 in 2000 (IEEE 2001).

The interface specification of HLA describes how to communicate within the federation through the implementation of HLA specification: the Run Time Infrastructure (RTI).

Federates interact among them using the services proposed by the RTI. They can notably "Publish" to inform about an intention to send information to the federation and "Subscribe" to reflect some information created and updated by other federates. The information exchanged in HLA is represented in the form of classical object oriented programming. The two kinds of object exchanged in HLA are Object Class and Interaction Class. The first kind is persistent during the simulation, the other one is just transmitted between two federates. These data structures are implemented with XML format. More details on RTI services and information distributed in HLA are presented in (DMSO 1998) and (IEEE 2001).

In order to respect the temporal causality relations in the simulation; HLA proposes to use classical conservative or optimistic synchronization mechanisms (IEEE 2001)

3.5 DEVS / HLA Components mapping

Zacharewicz (2006) proposed an environment, named LSIS_DME, for creating DEVS models HLA compliant and simulating them in a distributed fashion.

In LSIS_DME, a DEVS model structure can be split into distributed federate component models (Figure.5) in order to build a HLA federation (i.e. a distributed DEVS coupled model). The environment maps DEVS Local Coordinator and Simulators into HLA federates, it maps Root Coordinator into RTI. Thus, the "global distributed" model (i.e. the federation) is constituted of federates intercommunicating.

The DEVS models federates intercommunicate by publishing and subscribing to HLA interactions that map the coupling relations over the global distributed coupled model. This information is routed between federates by the RTI in respect to time management and Federation Object Model description. The federation execution is based on conservative synchronization algorithm and event-driven mechanism.

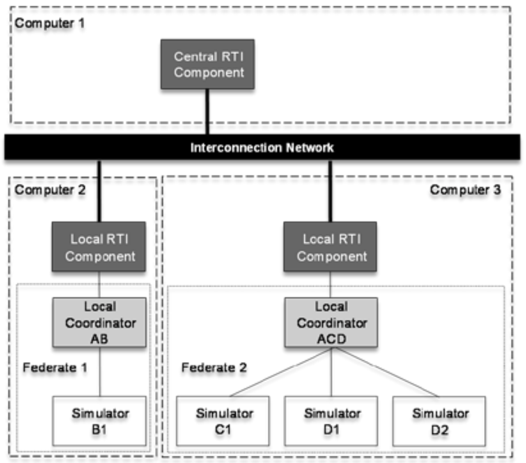

Figure 5: DEVS/HLA distributed simulation structure.

4 NEW PROPOSAL FOR MODELING SERVICES

The W3C proposed a XML representation of Service Modeling Language (SML) (W3C 2007) that is accepted as a standard in Service Modeling community. The XML Service Modeling process model structure correctness can be certified by referring to a Service Modeling Document Type Definition (DTD). However, this XML representation is not fully convenient for the XML specification of a Servuction Modeling introduced in §2.2. The description is more oriented Computer Science Service Modeling than industrial process Services oriented. For instance, the Service Modeling participant are not taken into account and being associated to tasks.

Thus, we propose to use the simple language, introduced by Bell (2008) to represent the components involved in that kind of Service Modeling. We add to this description the notion of Resource that is key to represent the service. The Resource will be part of the model if possible, if not, thanks to interoperability we will involve in the loop human or external machine or software. We describe a Service Model Structure SMS as composed of the following basic components, we recall in the following, Bell (2008) Service components definition and we add the Resource component:

SMS <Service Name, Ty, Sub, R, A, S, U, D, I, O, T, C>

Type Resource *Atomic, Compose, Cluster*
Sub Services *Services that compose a considered service (All others components not defined if atomic)*
Resource *Actor (Human, machine, software) invoked in the service*
Aggregated S*ervice containment*
Subtracted S*ervice retirement or elimination*
Unified S*ervice consolidation*
Decomposed S*ervice decoupling*
Intersected J*unction point between two or more service groups (known as service clusters)*
Overlapped C*ommon functionality and processes that services provide (typically between service clusters)*
Transformed T*ransformation of a service type to another (e.g. atomic to composite)*
Comment *Comments on the model*

A XML Service Modeling process model is composed of sub service components that treat in-progress service and controllers components that route in-progress service information between sub services. In-progress services pass over a sequence of these components. This model could be transformed into a coupled DEVS model by coupling DEVS atomic models representing Service Modeling basic components. This DEVS model takes advantage of formal properties enounced in § 2.2 and 2.3 and can be simulated.

4.1 IMS_SME Models Tool

A graphical Service Modeling Model Editor is under specification and test at Engineering Team lab in IMS Bordeaux. The Environment name is the acronym (IMS_SMT). The aim of this Service Modeling editor is to simplify and sound the creation of Service Modeling specification.

This editor will supply items to represent a sub-service flow model of a service with a graphical notation. The model is respecting grammar constraints based on the extended SOMF standards (i.e. § 4.) to assure the explicit validation representation. This software will be developed under Eclipse java programming environment; it will reuse J-GRAPH libraries of Java.

IMS_SMT will implement a set of graphical features from SOMT specification with extended component resource (as specified in § 4). The creation of a Service graphical specification is based on a hierarchical approach. The models will be stored in library and available for reuse. We present a screen shot of future IMS_SMT Tool in Figure 6 to detail the proposed features. The modeling window of the software will possess a horizontal toolbar for positioning, arranging the graphical components of the model. A vertical toolbar will contain buttons to create all the basic components introduced in § 3. Basic and hierarchical services can be drag and drop to be added to the model. The different kinds of management controllers of service flow are proposed (i.e. the service facilities described in Figure 2 and 3). Finally, Arrow tool can be selected to link the components.

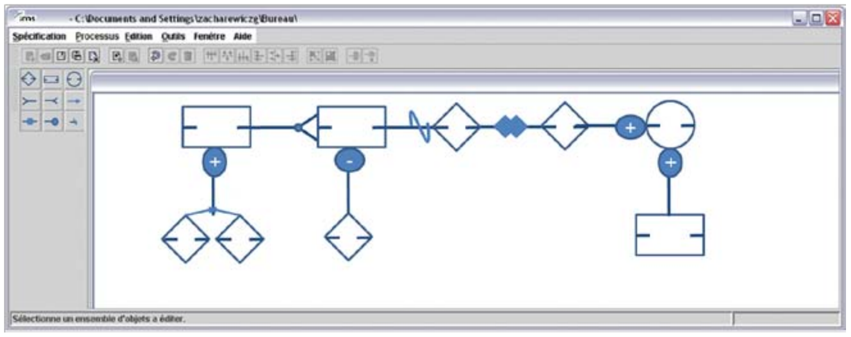
ims - C:\Documents and Settings\zacharewiczg\Bureau\
Spécification
Processus
Edition
Outils
Fenêtre
Aide
+
-
+
+
+
Sélectionne un ensemble d'objets a éditer.

Figure 6: Specification of IMS_SMT interface

The model depicted in Figure 6 represents the high-level Composite Service model (under IMS_SMT) of a Small business Service process extracted and adapted from Bell (2008) case studies. This model consists of 8 services (atomic or composed). We precise, in particular, that the edition of a "Atomic Service" component, done by double clicking on the Atomic service icon in the model will be proposed to set up service properties. In the pop up window appearing, the Service parameters can be configured (i.e. Name, Type, Priority, Description, Action and Resources allocation). The environment will possess an on-line syntactic check-in Service Modeling. This feature is required to check of the syntactic validity of the connections between components. This structure checking is proposed to clarify and facilitate the validation of Service Models.

This specification of the software need actually to be completed to fulfill perfectly our requirements. The advanced management of the resources can be improved. For instance the resources are actually directly connected to the tasks. We plan to define a Resource Manager component to handle centrally the resources and give a high level view of the resources allocation status. The transitions components route the service flow of information based on flow value, it could be interesting to parameter the routing of the flow in the transition component based on user specified functions. The creation of an advanced edition window box for transition could reach this requirement. Finally, we envisage to export Service Modeling models towards different formats, for instance to exchange models with commercial Service Modeling tools.

4.2 DEVS Models Library

Service models are just graphical models, they are not associated to Simulation language. In order simulate them, our idea is to transform them into models that posses formal properties for simulation. We have chosen DEVS for the reasons evoked in §3.1.2. From services models, we are developing DEVS models libraries for service modeling. These stored models can be reused to instantiate new services and enhance the user modeling process. A DESV library of small services processes is being implemented in our lab. Once the library will be finalized, we will model small services in order to validate properties on the model simulated. Properties targeted are: time management to realize a service, Resource allocation (under use, deadlock, performance indicators,...) and classification of service scenario regarding list of properties desired by the user.

We illustrate in the figure 7, the DEVS atomic model of an atomic Service, this model behavior is designed to deal with a service expressed by a client, the resources (humans, material, software...) allocation to supply the service and the time constraints of the service. The model will receive a demand of service and will produce the service and information associated to as an output. To realize a service, this model requests the availability of a resource (by coupling relation to a resource model). Then, when receiving the availability and the capacity of the resource the model describes the service to realize in terms of time duration, resource involvement and exchange with the client. At the end model release the service result and free the resource. Details on the accomplished service is outputted as a message, this message interpreted to the service modeler or relayed to the next Service model if part of a composite or cluster service.

The user can connect this Service DEVS atomic model to others in order to define composite services using coupling DEVS coupling relation of DEVS coupled models. For that purpose, the models are stored into libraries to be reused. The models have been realized using LSIS_DME (Zacharewicz 2008) DEVS editor and simulator. The library of Service DEVS models can be seen in the left column of LSIS_DME editor tool. We are working at proposing one sufficiently generic (to be instantiated regarding the specification of the service) DEVS model equivalent to each graphical Service Model Component presented in § 4.

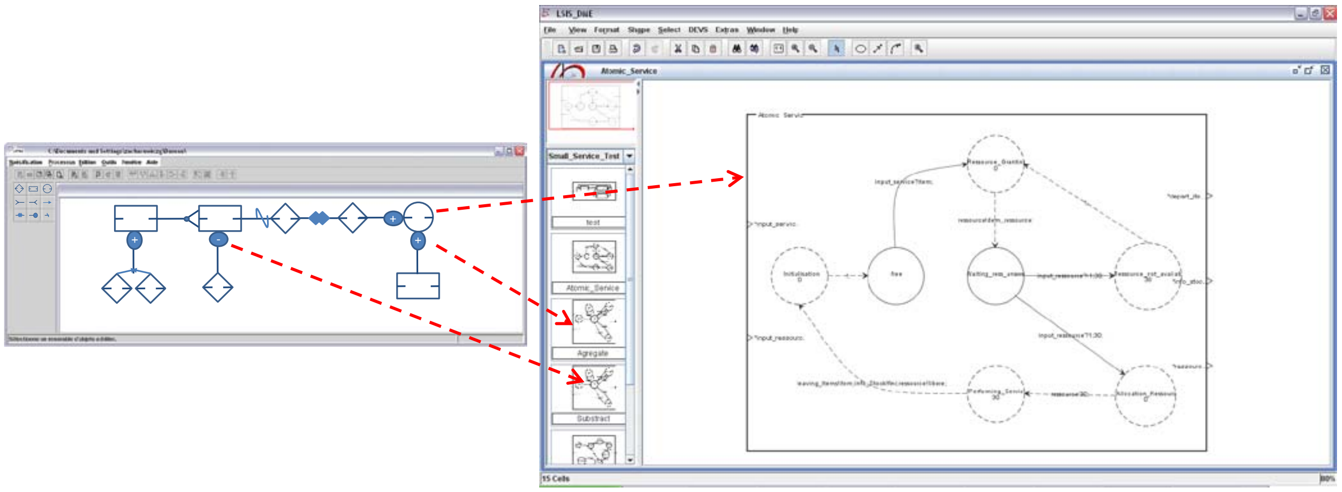

Figure 7: DEVS Model of Atomic Service

5 FROM SERVICE MODELING TO DEVS SIMULATION

5.1 Steps to transform Service Model into DEVS Model (Build Time)

We have proposed a set of Service modeling components in §4.1 and equivalent DEVS models in §4.2. We propose a general method in three steps, described in Figure 8, inspired from Workflow models transformation (Zacharewicz 2008), to automate the transformation process of a Service model into a DEVS model.

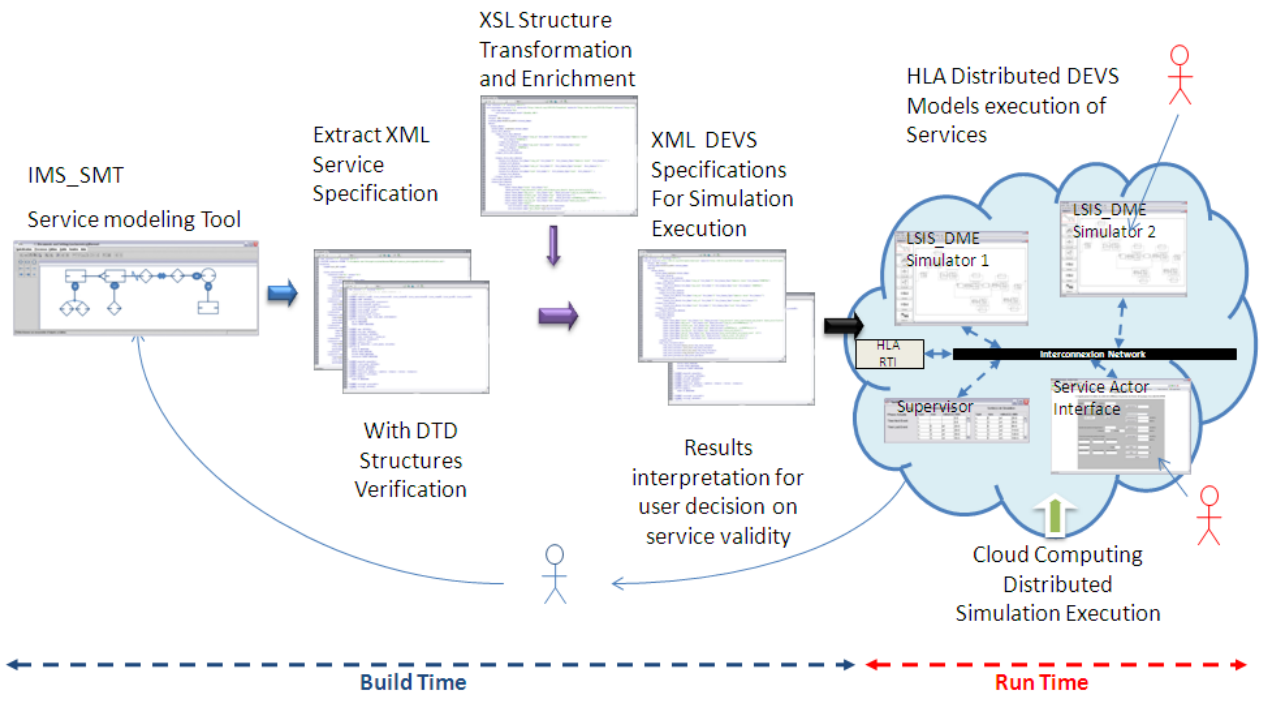
XSL Structure Transformation and Enrichment
IMS_SMT
Service modeling Tool
Extract XML Service Specification
XML DEVS Specifications For Simulation Execution
HLA Distributed DEVS Models execution of Services
LSIS_DME Simulator 1
LSIS_DME Simulator 2
HLA RTI
Interconnexion Network
Supervisor
Service Actor Interface
With DTD Structures Verification
Results interpretation for user decision on service validity
Cloud Computing Distributed Simulation Execution
Build Time
Run Time

Figure 8: Service Modeling to DEVS / HLA Big Picture

In a first step, the Service graphic model is exported from IMS_SMT in the Service Modeling XML format adapted from Latest draft of the SML-IF (2007) specification (cf. Figure 8 blue arrow). The correctness of the structure of this XML Service Modeling is verified by a Service Modeling XML DTD.

In a second step (cf. Figure 8 purple arrows), we apply an Extensible Stylesheet Language (XSL) on the XML structure of the Service Modeling in order to modify its tree presentation to a XML DEVS coupled model structure. The algorithm defines the coupling between atomic models that represent Service Modeling tasks and controllers. It adds also in the XML DEVS implicit information from Service Modeling. For instance, the coupling of resources on Atomic Services with resources invoked that does not appear graphically in the Service Model (for their allocation and synchronization) is defined as DEVS coupling relation. Also temporal information needs to be sometimes clarified from the semi formal graphical notation of service modeling that is missing information required for simulation; in that case some syntactic information can be automatically added; in addition some others semantically ambiguities on the models, not automatically tackled, are asked to be resolved by the modeler. From these statements, the XSL algorithm instantiates a DEVS atomic model for each service, resource and controller able to be modeled by a XML DEVS description. In order to illustrate the end model representation for simulation, we have presented previously in Figure 7 the DEVS atomic model of an Atomic Service of the Business model.

In a third step (c.f. Figure 8 black arrows), The DEVS simulator loads data from the XML description for each DEVS atomic models (e.g. service duration functions, required resources, aggregations, unification, decomposition...) to instantiate atomic DEVS models. In addition, it reads the XML description of included DEVS models coupling, to generate DEVS coupled model representing the global Service Modeling process (e.g. Figure 8 DEVS Simulators in the cloud). The execution will be specified to run in cloud computing environment. The idea is to run distributed simulation able to call distributed and heterogeneous actors during run time. For e.g., in the Figure. 8 human are involved in the loop. The synchronization of the various component will be coordinated by HLA RTI.

Indeed, implicit notions in the Service Modeling have to be explicitly specified in the model for the simulation. For instance, every Atomic Service component is thus represented by a DEVS Service atomic model coupled with models of resources (human, material not modeled resources) which process services satisfaction within the framework of this service (e.g. cloud group in Figure 8).

Furthermore, Service Modeling in-progress information are mapped into DEVS coefficient events (Zacharewicz 2006) with a list of values containing for e.g. service reference, values, time of processing and routing information. These services to perform, planned in the Service Modeling, are inserted in the input event scheduler of the DEVS simulator.

Finally, the simulation of the Service Modeling DEVS coupled model is run (We give details on the next section). The dynamical model decomposition to run in the cloud by taking into account computers load balance during run time is still a challenging perspective. Right now the simulation are just HLA distributed, the online load balance is not yet addressed. The result of the simulation gives a log report containing the output events generated (cf. Figure 8 Thin blue arrows). The results of the simulation are interpreted and sent back to IMS_SMT in order to be understandable by the user of the Service Modeling tool. The Service simulation supplies: service processing time, delay due to missing resources, deadlocks, bottlenecks, resource allocation and synchronization statistical feedbacks.

These results are employed to assist human-decision making to create or modify a Service flow definition. It permits to prevent errors that could occur when modifying to a wrong Service flow and implementing it. It allows also comparing quantitatively several modifications propositions of a Service (Service Scenarios) in order to sort the most efficient ones regarding user service specification.

5.2 DEVS Service Simulation Environment HLA Compliant (Run Time)

We presented, in § 3.2, that DEVS models can be run from several computers thanks to the capability of LSIS_DME to create HLA federates. This capability matches with the distribution and interoperability requirements of up to date service definition (Bell 2008). From this postulate and conforming to the structure presented in Figure 8, we propose to use LSIS_DME (Zacharewicz 2008) as the execution engine of a distributed Service Modeling and Simulation environment. Then, we propose to generalize the HLA compliance to the whole Distributed Service Modeling environment by adding heterogeneous federates (users interface, software...). This new architecture is presented in the Figure 8 cloud. Regarding interoperability science, we situate the interoperability provided by this solution as unified interoperability (i.e. §3.3.1) because we will share information based on common Ontologies. The detail on information shared is given below.

Therefore, during build time, we incorporate the modeling tool IMS_SMT into a HLA federate to display updated animated in progress service information. For that purpose we add RTI ambassador code to the tool to become HLA compliant (IEEE 2000). Then the models defined in XML by IMS_SMT, generated by the automatic process as described in§ 5.1, are integrated into HLA objects and shared with LSIS_DME. In detail, IMS_SMT publishes to HLA objects that represent the Service models and LSIS_DME subscribes to them to receive the DEVS equivalent models. The updates of model structure information is routed by a HLA RTI. If the Service Modeling model is modified by the user of IMS_SMT, LSIS_DME is informed of these changes; it could take them into account in its DEVS model and reruns the simulation with the new model structure and atomic models settings.

During run time (simulation or execution), LSIS_DME updates, in a HLA object, the log of events resulting of the simulation. These results are subscribed by IMS_SMT to give to the users the simulation animation and results. For this reason, this software can be seen as the modeling, control and administration tool of the Service Modeling environment.

Some Services can be simulated some not, for this reason, the environment will propose to involve actors users, client, and resources in the loop during run time. The requirement is coming from the fact some intervention of human or machine are two complex and not tackled by the models and their simulators, so some external choice can be invoked pausing the simulation and then restarting it updated by external information.

On the one hand, we propose to integrate human in the loop, to do qualitative choice during the simulation. For that purpose, we implement web interfaces called during the simulation, by the Service Modeling engine, in order to specify for instance some routing of items in the process. Data exchanged during the call are defined in HLA objects.

On the other hand, some complex mathematical computation of data treatment, are not taken into account in transition/output functions of the DEVS model described with LSIS_DME. In that case the simulation is interrupted and the data are transferred to specific software by publishing to an object. This software computes and sends back data to the process definition tool by publishing to a HLA object or interaction.

6 FUTURE WORK

A beta version IMS_SMT software is still under specification and test; it will propose graphical features to describe and to interconnect all the basic components of a Service Model introduced previously. However evolutions concerning resource, service and transition management are under development to improve it.

To finalize the development of the proposed environment, the interoperability is a key research domain, especially when relying the concept of ontology sharing in the context of service definition (Zacharewicz 2009). The addition of others "actors" and "invoked applications" components into the Service Modeling environment are considered, their adding will be possible and facilitated by making these components referring to a common HLA compliant ontology.

The actual future internet 3.0 is looking at defining most of the web as efficient services, in particular the computing power will be dynamically reconfigured in a cloud of distributed computers to distribute the service load on resources computer distributed all around the world. The distributed simulation concepts used in our environment will facilitate the use of this computing concepts. We will implement our modeling and simulation tool based on Grid and next generation Cloud Computing Services (Gubala 2006).

Finally, the plug on a real process and human on the loop is central consideration to define our real time Service M&S environment. This solution will require operating sensors on real systems and interfaces with users, we believe the internet of the things will be one key for this environment. At the end, distributed simulation will exchange messages between distributed components and so it will need efficient synchronization algorithm (Fujimoto 2000) (Zacharewicz 2006) to respect functional and temporal causality and to run as fast as real time.

7 CONCLUSION

The service value and coherence analysis is making obvious that the behavior of the customer and the contact personnel involved in service accomplishment is influencing the service quality and must be part of the service specification. These behaviors are stereotyped (bounded to templates), and that even if there exists service variants. In consequence the possibilities around a service definition in real world are not infinite, so we are able to model and simulate a service and its interoperability with resources in a unified model to sort the most efficient solutions to be developed and implemented.

From that postulate, this position paper presents the new perspective of a Service Modeling & Distributed Simulation Environment IMS_SMT. It provides a set of concepts to describe a graphical specification of a Service Model. In addition we propose methodological transformation steps from Service Modeling XML specification to DEVS models. We argue that the use of the DEVS formalism allows the verification and validation of Service specifications properties by simulation.

We propose, in addition, to simulate HLA compliant DEVS services models. The use of the HLA specification facilitates interoperability, by connecting new distributed HLA compliant actor interfaces, machines or software components to the Service Modeling environment.

The application field of the Service Modeling is new and the fast increasing of uses and researches in this domain, these last years, let us conclude on the potential of this domain. Finally, the weak bibliography in the crossing of Service Modeling, Distributed Simulation and next generation of the internet gives evidence of relevant perspectives.

ACKNOWLEDGMENTS

This research has been financially supported in part by RAUDIN project, Feder N°31462, Conseil Régional d'Aquitaine and Bordeaux 3 University.

REFERENCES

Bell M. 2008. *Service-Oriented Modeling*. p. 158-159

Chen, D., M. Dassisti, and B. Elvesæter, Enterprise Interoperability Framework and Knowledge Corpus - Final report Annex: Knowledge Pieces, Contract no.: IST508 011, Deliverable DI.3Annex, May 21, 2007.

Defense Modeling and Simulation Office (DMSO), U.S. Dept of Defense. 1998. High Level Architecture Rules Version 1.3. IEEE P1516.2, Standard for M&S, April.

Eiglier P., Langeard E., 1987. *Servuction, le marketing des services*. Mc Graw Hill, Paris.

Fujimoto R. M. 2000. *Parallel and Distributed Simulation System*. Wiley Interscience, NY.

Gadrey J., 1996, "L'économie des services". La Découverte, collection Repères, 2e éd.

Gallouj C., Gallouj F., 1997, "L'innovation dans les services et le modèle du cycle du produit inversé", *Revue Française de Gestion*, mars avril-mai, p. 82-97.

Grönroos C. 1999. Le marketing des services : consommation et marketing des processus. *Revue française de marketing*, n°171.

Gubala T., M. Bubak, M. Malawski. Support for Automatic Workflow Composition Based on Ontologies in Semantic Grid Environment. In *5th Cracow Grid Workshop (CGW05) -- Workshop Proceedings*, ACC Cyfronet AGH, (2006), pp.34-40.

Hill T.P. 1977. On Goods and Services. *The Review of Income and Wealth*, vol. 4, December, p. 315-338.

Hong K. J., J-K. Lee, D. H. Kim, T. G. Kim. 1999. DEVS Framework Instrumented with Database for Web-Based Service Modeling Simulation. *WMC'99 Websim*, LA, CA, USA, January 17-20

IDEAS Consortium. 2002. Thematic Network, IDEAS Interoperability Development for Enterprise Application and Software Roadmaps. Annex 1 – Description of work, May.

Institute of Electrical and Electronic Engineers 2001. High Level Architecture (HLA) - Federate Interface Specification IEEE Standard for Modeling and Simulation (M&S). std 1516.2-2000, March.

Sarjoughian, H., Kim, S., Ramaswamy, M., and Yau, S. 2008. A simulation framework for service-oriented computing systems. In *Proceedings of the 2008 Winter Simulation Conference*, eds. S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler, 845-853. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

SML-IF specification, Available via http://www.w3.org/TR/sml-if/, [accessed April 12, 2009].

Song H.S. and T.G. Kim, "The DEVS framework for discrete event systems control", In 5th Conference on AI, Simulation and Planning in High Autonomous Systems:228-234, Gainesville, USA, 1994.

W3C, Service Modeling Language (SML) Submission request to W3C, Available via http://www.w3.org/Submission/2007/01/, [accessed 25 March 2009].

Zacharewicz G., N. Giambiasi and C. Frydman. 2006. Lookahead Computation in DEVS/HLA Environment, *Simulation News Europe Journal (SNE) special issue "Parallel and Distributed Simulation Methods and Environments"*, vol. 16, n° 2, pp. 15 - 24, September 2006. ISSN, 0929-2268.

Zacharewicz, G., C. Frydman and N. Giambiasi. 2008. G-DEVS/HLA Environment for Distributed Simulations of Workflows, *Transactions of the SCS, Simulation*, Vol. 84, No. 5, pp.197-213.

Zacharewicz G., Chen D., Vallespir B. 2009. Short-Lived Ontology Approach for Agent/HLA Federated Enterprise Interoperability. In *proceedings of International Conference I-ESA China 2009 Interoperability for Enterprise Software and Applications - International Conference I-ESA*, China

Zeigler B.P., 1976. *Theory of Modelling and Simulation*. Ed. John Wiley, New York.

Zeigler B.P.; H. Praehofer; T. G. Kim. 2000. *Theory of Modeling and Simulation. 2nd Edition*, Academic Press, New York, USA.

AUTHOR BIOGRAPHIES

GREGORY ZACHAREWICZ is Associate Professor in Bordeaux 1 University (IUT MP). His research interests include Discrete Event Modeling (e.g. DEVS), Distributed Simulation, Distributed Synchronization Algorithms HLA, and Workflow. He recently focused on Enterprise Modeling and Interoperability. He has published more than 20 papers in Conference and

International Journals. He is reviewer in many Conferences (SCS Conferences, WinterSim...) and SCS Simulation journal. His email is <gregory.zacharewicz@ims-bordeaux.fr>.

THÈCLE ALIX is Associate Professor in Bordeaux 4 University (IUT GLT). Her research interests include Product Service Systems, Service value analysis and service delivery system. She was involved in many international projects (RAUDIN, Research on the use for the development of numeric disposals (FEDER project), 2003 - 2007: INTEROP NoE (IST 508011), CENNET (China)). She has published more than 20 papers in Conference and International Journals. Her email is <thecle.alix@ims-bordeaux.fr>.

BRUNO VALLESPIR is Full Professor at University Bordeaux I and member of IMS. Within this framework, he is today responsible of the research group about production systems. Bruno VALLESPIR's research topics are production management, performance evaluation and enterprise modelling including modelling tools and design methodologies. On these topics, he published more than 60 papers in conferences, journals and books, was involved in around 20 international, European (from FP1 to FP6, mainly ESPRIT) or French projects and working groups. Recently, he was involved in INTEROP NOE. He is member of several working groups and steering committees of IFIP and IFAC. His email is <bruno.vallespir@ims-bordeaux.fr>.