Journal of Computational Science
Manuscript Draft

Manuscript Number: JOCS-D-14-00076

Title: TIME-BASED ORCHESTRATION OF WORKFLOW, INTEROPERABILITY WITH G-DEVS/HLA

Article Type: SI: I3M 2013

Keywords: Workflow, Taverna, Interoperability, Discrete event simulation, G-DEVS, HLA

Abstract: Enterprises information systems (EIS) take benefits of latest advanced of web services and internet of things to improve information retrieving and gathering for decision making. Furthermore, EIS should permit a more comprehensive information routing in the company within an electronic workflow in order to save time, cost and to reduce production impact on the environment. Such software has to interact frequently with real world data acquired by different sensors. Nevertheless this combination of software and hardware devices frequently faces interoperability problems. Also, testing and validating the EIS is not trivial without testing in real condition that can lead to deploy the large system. Authors assumed that testing and validating part of the system behaviour can be anticipated progressively by simulation, permitting then more progressive and confident system integration. This paper proposes to introduce a new workflow demonstration platform to combine simulation world with real world interacting with sensor, human interfacing and web service calls. In detail, this paper proposes to combine the Taverna Workflow tool, which handles and triggers web services call proposed by a platform server, to other software components. This combination has revealed one drawback of major workflows orchestrators; they do not provide time management facilities to handle synchronization during parallel execution of interdependent workflows. To overcome that limitation a clock ordering solution has been added by reusing G-DEVS/HLA to synchronize workflows running in parallel. The imbrication of G-DEVS M&S with Taverna workflow is now operating thanks to HLA. This work is validated by demonstrating the interoperability and the complementarity of these approaches on a logistic platform study case.

Gregory ZACHAREWICZ is Associate Professor at the University of Bordeaux (IUT MP) Lab. IMS. His research interests include Discrete Event Modelling (e.g. DEVS, G-DEVS), Distributed Simulation, Distributed Synchronization Algorithms, HLA, FEDEP, MDA, Short lived Ontologies, ERP, BPMN, and Workflow. He recently focused on Enterprise Modeling and Interoperability. He has published more than 50 papers in Conference and International Journals. He is Chair of TMS/DEVS Conference and is frequent Reviewer in Conferences (SpringSim, SummerSim, WinterSim...) and SCS Simulation journals. He is involved in several French, European and Transatlantic projects.

Judicaël Ribault is Postodctoral fellow at the University of Bordeaux (IUT MP) Lab. IMS. His research interests include Component-based software engineering and simulation, Discrete Event Modelling (e.g. DEVS, G-DEVS), Distributed Simulation, Large-Scale Simulation, Simulation Interoperability and Workflow. He has published several papers in Conferences and is frequent Reviewer in Conferences (SpringSim, TMS/DEVS, etc.). He is involved in European projects.

***Biographies (Photograph)**

***Biographies (Photograph)**

TIME-BASED ORCHESTRATION OF WORKFLOW, INTEROPERABILITY WITH G-DEVS/HLA

Judicaël Ribault, Gregory Zacharewicz

Univ. Bordeaux, IMS, UMR 5251, F-33400 Talence, France

judicael.ribault@u-bordeaux1.fr, gregory.zacharewicz@u-bordeaux1.fr

ABSTRACT

Enterprises information systems (EIS) take benefits of latest advanced of web services and internet of things to improve information retrieving and gathering for decision making. Furthermore, EIS should permit a more comprehensive information routing in the company within an electronic workflow in order to save time, cost and to reduce production impact on the environment. Such software has to interact frequently with real world data acquired by different sensors. Nevertheless this combination of software and hardware devices frequently faces interoperability problems. Also, testing and validating the EIS is not trivial without testing in real condition that can lead to deploy the large system. Authors assumed that testing and validating part of the system behaviour can be anticipated progressively by simulation, permitting then more progressive and confident system integration. This paper proposes to introduce a new workflow demonstration platform to combine simulation world with real world interacting with sensor, human interfacing and web service calls. In detail, this paper proposes to combine the Taverna Workflow tool, which handles and triggers web services call proposed by a platform server, to other software components. This combination has revealed one drawback of major workflows orchestrators; they do not provide time management facilities to handle synchronization during parallel execution of interdependent workflows. To overcome that limitation a clock ordering solution has been added by reusing G-DEVS/HLA to synchronize workflows running in parallel. The imbrication of G-DEVS M&S with Taverna workflow is now operating thanks to HLA. This work is validated by demonstrating the interoperability and the complementarity of these approaches on a logistic platform study case.

Keywords: **Workflow, Taverna, Interoperability, Discrete event simulation, G-DEVS, HLA**

1 INTRODUCTION

The effectiveness of enterprise information system (EIS) depends not depend anymore only on the internal interconnectivity of its inner software components, but more and more on its ability to exchange data, so to collaborate, with every day new tools developed and updated in the environing digital world. This requirement led to the development of the concept called interoperability that intends to improve collaborations between EIS companies. No doubt, in such context where more and more networked enterprises are developed; enterprise interoperability is seen as one of the most wanted feature in the development of an EIS. Also, data treatment calls actions of both human processing and automatic treatments. The sequencing of these actions should be controlled or orchestrated by a high level application that can decide the human resource and/or component to solicit. The sequence of actions is commonly entitled Workflow (WF) and its administration Workflow management. This field is studied and standardized by the Workflow Management Coalition (WfMC) [WfMC, 1999] [WfMC, 2005].

Several research-works have been launched since the 90's in the field of WF. Workflow was first designed to formalize and improve enterprise business process. A production workflow is a set of linked steps required for developing a product until it is put on the market [Weske, 2012]. The workflow steps are based on observing a number of steps that were originally manually enchained then formalizing them to be computer assisted. The research on the WF initiated by the Workflow Management Coalition [WfMC, 1999] [WFMC, 2005] and used for instance in [Zacharewicz et al., 2008] was a premise to current WF modelling (e.g. with Business Process Model and Notation (BPMN) [OMG, 2011]). It has permitted for instance the development of Build Time models used for setting Enterprise Resource Planning (ERP) systems.

Deploying such WF is a critical task for the companies that continue to rely on their EIS during the setting. Moreover the proper functioning is difficult to achieve because the installing team doesn't have vision or access to the whole system (EIS environment) during settings, so the final global behaviour is difficult to predict. Executing the WF on part of the real system, while simulating some critical parts that will then be deployed, can be a good option to test the WF behaviour and reduce risk and cost. However, most WF tools and service orchestration are limited in the handling of time management. But without time consideration, executing a parallel simulation with disjunction and junction gateway between tasks is difficult. Distributed simulation has a long time experience in this field and can be an answer for this problem. Few approaches combine efficiently Modelling and Simulation (M&S) and real executions in the WF domain. Main reasons are: slowing-down due to synchronization of the simulation engine, that is usually constrained by pessimistic causality [Chandy and Misra, 1979] between real and simulated time, and interoperability barriers that are faced between different hardware and software [Chen and Doumeingts, 2003].

Recent improvements in web-based development propose new facilities to connect applications in a more convenient way. For instance, web services can solve part of the interoperability question. WF can be used as an interoperability layer between services, and especially Web Services (WS). This paper proposes to use web services and WF for interoperability between simulation and real-world application. Web services enable the integration of applications or data from heterogeneous sources (i.e. Mash-up). Nevertheless the synchronisation is not solved; this work proposes an additional component from the High Level Architecture standard named Run Time Infrastructure (RTI) reused from the G-DEVS/HLA works. In the end, this paper proposes to apply the use of WF Web services and simulation to the transport domain application through the PRODIGE project to validate the approach.

Section 2 describes the necessary background needed to understand how WFs of services and simulation can drive real application. Section 3 presents the scientific contribution while section 4 put it into practice in a real application framework.

2 BACKGROUND

In this section, we first present the enterprise interoperability concept. Then we briefly present the HLA standard for interoperability of simulation and how WF can be used for experimentation. We recall the DEVS formalism and G-DEVS. Finally we introduce the Taverna WF management system to orchestrate the experimentation.

2.1 Enterprise interoperability

Enterprise Interoperability [Chen and Doumeingts, 2003] refers to the interaction ability 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 semantic defined in a given business context.

Interoperability extends beyond the boundaries of any single system, and involves at least two entities. Consequently establishing interoperability implies relating two systems together and removing incompatibilities. Concepts related to enterprise interoperability are classified into three main dimensions as described in [Chen and Doumeingts, 2003]:

- The integrated approach demands all partners to have the same description of information.
- The unified approach asks partners to prepare the data to exchange in order for it to be compliant with a Meta model but local description can be kept.
- The third approach is federated. Here, interoperability must be accommodated on-the-fly between partners without considering a pre-existing Meta model.

The goal of these approaches is to tackle interoperability problems through the identification of barriers (incompatibilities) which prevent interoperability. The first kind of barrier concerns the nonexistence of commonly recognized paradigms and data structure, for that, clarification is required to propose a sound paradigm. The second requirement is the synchronization of data. The order of exchanged data is important, ignoring this can lead to misunderstanding and malfunction of the model. Finally the enterprise modelling must take into account the confidential management of data. In this privacy context, concurrent enterprises must define data sharing strategies. The interoperability can be considered between concurrent enterprises in that context, a strategy of data sharing/not sharing between these must be defined. In the presented work the interoperability is focused between WF simulation and service calls. In the simulation domain, the HLA is established as the interoperability reference.

2.2 Simulation interoperability with HLA

The High Level Architecture (HLA) [IEEE, 2000] [IEEE, 2003] is a software architecture specification that defines how to create a global software execution composed of distributed simulations and software applications. This standard was originally introduced by the Defense Modelling and Simulation Office (DMSO) of the US Department Of Defence (DOD). The original goal was the reuse and interoperability of military applications, simulations and sensors.

2.2.1 HLA concepts

In HLA, every participating application is called federate. A federate interacts with other federates within a HLA federation, which is in fact a group of federates. The HLA set of definitions brought about the creation of the standard 1.3 in 1996, which then evolved to HLA 1516 in 2000 [IEEE, 2000] and finally to 1516 Evolved [IEEE, 2010].

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 using the proposed services by the RTI. They can notably “Publish” to inform on the intention to send information to the federation and “Subscribe” to reflect 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 run time, the other one is just transmitted between two federates. These objects are implemented with XML format. More details on RTI services and information distributed in HLA are presented in [IEEE, 2000] and [IEEE, 2010]. In order to respect the temporal causality relations in the execution of distributed computerized applications; HLA proposes to use classical conservative or optimistic synchronization mechanisms [Fujimoto, 2000]. In HLA 1516 Evolved [IEEE, 2010] the service approach is demanded as core feature. Nevertheless no software addresses completely that goal at the moment [Tu et al., 12].

2.2.2 HLA Implementation Components

An HLA federation is composed of federates and a Run time Infrastructure (RTI) [IEEE, 2000].

A federate is a HLA-compliant program, the code of that federate keeps its original features but must be extended by other functions to communicate with other members of the federation. These functions, contained in the HLA-specified class code *FederateAmbassador*, make the information received resulting from the federation interpretable by a local process. Therefore, the federate program code must inherit the *FederateAmbassador* class code, complete the abstract methods defined in this class, to be able to receive information from the RTI.

The RTI supplies services required by distributed executions, it routes messages exchanged between federates and is composed of two parts. The “Local RTI Components code” (LRC, e.g. in Figure 1) supplies external features to the federate to use RTI call back services such as handling objects and time management. The implementation is the class *RTIAmbassador*, which transforms the data coming from the federate in an intelligible format for the federation. The federate program calls the functions of *RTIAmbassador* in order to send data to the federation or to ask information to the RTI. Each LRC contains two queues, a FIFO queue and a time stamp queue to store data before delivering to the federate.

Finally, the “Central RTI Component” (CRC, e.g. in Figure 1) manages the federation notably by using the information supplied by the FOM [IEEE, 2003] to define Objects and Interactions classes participating in the federation. Object class contains object-oriented data shared in the federation that persists during the run time, Interaction class data are just sent and received.

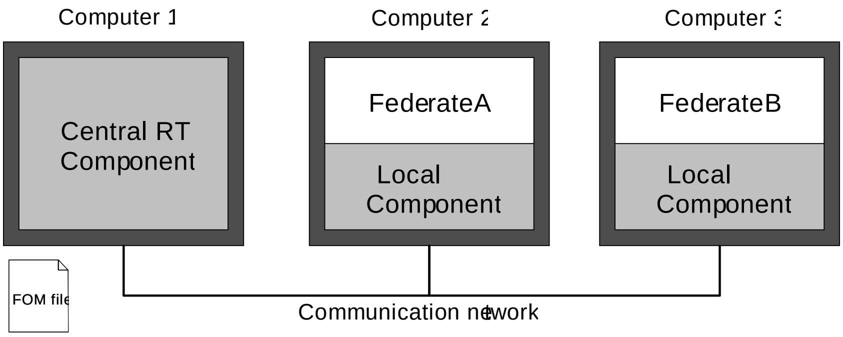

Figure 1. HLA implementation components

A federate can, through the services proposed by the RTI, "Publish" and "Subscribe" to a class of shared data. "Publish" allows to diffuse the creation of object instances and the update of the attributes of these instances. "Subscribe" is the intention of a federate to reflect attributes of certain classes published by other federates.

2.3 DEVS and G-DEVS M&S

Discrete EVent Specification (DEVS) was introduced by [Zeigler et al., 2000]. This Moore based language describes a dynamic system with a discrete event approach using some typical concepts. In particular it represents a state lifetime.

When a lifetime is elapsed an internal transition occurs that changes the state of the model. The model also takes into account the elapsed time while firing an external state transition triggered by an event received from outside the considered model.

The behavioural models are encapsulated in atomic models that are completed with input and output ports. Then, these models can be composed with others by connecting inputs and outputs. The composed models are called coupled models.

Generalized DEVS (G-DEVS) emerged with the drawback that most classical discrete event abstraction formalisms (e.g. DEVS) face: they approximate observed input–output signals as piecewise constant trajectories. G-DEVS defines abstractions of signals with piecewise polynomial trajectories [Giambiasi et al., 2000]. Thus, G-DEVS defines the coefficient-event as a list of values representing the polynomial coefficients that approximate the input–output trajectory. Therefore, an initial DEVS model is a zero order G-DEVS model (the input–output trajectories are piecewise constants). In fact G-DEVS was the pioneer DEVS extension proposing a multi value event.

G-DEVS keeps the concept of the coupled model introduced in DEVS [Zeigler et al., 2000]. Each basic model of a coupled model interacts with the others to produce a global behaviour. The basic models are either atomic or coupled models that are already stored in the library. The model coupling is done with a hierarchical approach (due to the closure under coupling of G-DEVS, models can be defined in a hierarchical way).

On the simulation side, G-DEVS models employ an abstract simulator [Zeigler et al., 2000] that defines the simulation semantics of the formalism. The architecture of the simulator is derived from the hierarchical model structure. Processors involved in a hierarchical simulation are: Simulators, which implement the simulation of atomic models; Coordinators, which implement the routing of messages between coupled models; and the Root Coordinator, which implement global simulation management. The simulation runs by sending different kind of messages between components. The specificity of G-DEVS model simulation is that the definition of an event is a list of coefficient values as opposed to a unique value in DEVS.

Zacharewicz et al. proposed in [Zacharewicz et al., 2008], an environment, named DEVS Model Editor (LSIS_DME), to create G-DEVS models that are HLA compliant and simulating them in a distributed fashion. In LSIS_DME, a G-DEVS model structure can be split into federate component models in order to build a HLA federation (i.e. a distributed G-DEVS coupled model). The environment maps DEVS Local Coordinator and Simulators into HLA federates and it maps Root Coordinator into RTI. Thus, the “global distributed” model (i.e. the federation) is composed of federates intercommunicating.

2.4 Workflow

Workflow was first designed to improve the business process. A production workflow is a set of steps required for developing a product until it is put on the market [Weske, 2012]. The workflow steps are based on observing a number of steps that are usually repeated manually and formalizing them. Workflows can be useful in Modeling and Simulation (M&S) for several reasons. The first one is that they allow building a blueprint of the simulation experiment, ensuring its replayability. The second one is that they allow building a simulation experiment independent from the simulation environment [Ribault and Wainer, 2012]. The Workflow Management Coalition (WMC) standard group (WMC 2009) proposes a WF reference model in which the WF is in the centre and interacts with other surrounding applications or WF components.

Several surveys have compared different workflow management systems. In [Deelman et al., 2009], the authors analyzed and classified the functionality of workflow system based on the needs of scientists who use them. In [Yu and Buyya, 2006], the authors focused on the features to access distributed resources. In [Curcin and Ghanem, 2008], four of the most popular scientific systems were reviewed. We can abstract from the literature that:

- Kepler [Altintas et al., 2004; Lud¨ ascher et al., 2006] is a scientific workflow system with a graphical interface to create, execute and share workflows. Inputs and outputs are typed, which allows to validate semantically the workflow prior to execution. Kepler uses an archive format for sharing workflows, and a repository. Kepler can invoke a web service through a dedicated actor, and broadcast the response through its output port.
- Triana [Churches et al., 2006] is a problem solving environment with a graphical interface to create and execute workflows. Workflows are data-driven, and special elements enable branching, parallelism and looping. Triana uses scripts to control sub-workflows and it can invoke web services.
- Taverna [Hull et al., 2006] is a workflow management system dedicated to the integration of services. Taverna has a graphical interface for the creation, execution and sharing of workflows. Taverna is interfaced with the myExperiment service [Goble and De Roure, 2007] to share workflows.
- Worms [Rybacki et al., 2011] is a flexible and extensible workflow system dedicated to M&S. It is plug-in-based which offers the possibility to extend its features. Worms also comes with its own workflow repository.

In [Tan et al., 2009], the authors compare the service discovery, service composition, workflow execution, and workflow result analysis between BPEL and a workflow management system (Taverna) in the use of scientific workflows. They determine that Taverna provides a more compact set of primitives than BPEL and a functional programming model that eases data flow modelling.

We decided to use Taverna to demonstrate the feasibility of our methodology because Taverna eases the interoperability with other services.

2.5 Taverna

Taverna [Hull et al. 2006] is an application that facilitates the use and integration of a number of tools and databases available on the web, in particular Web services. It allows users who are not necessarily programmers to design, execute, and share WFs. These WFs can integrate many different resources in a single experiment.

Taverna WF can contain services including:

- A service capable of running Java code directly within Taverna.
- A service to run a remote application via the REST protocol.
- A service to run a remote application via the SOAP/WSDL protocol.

A Taverna service can take inputs and produce outputs. The value of an entry can be part of the WF (hardcoded) or a parameter to provide information during the execution of the WF. Taverna offers the possibility to automatically format the input and output based on the type of parameters required by the service.

WFs are particularly suited to automate experiments, but all necessary parameters cannot always be specified in advance. In these cases, it is desirable to interact with users for decision making. Taverna offers several graphical interfaces for interacting with the user.

A Taverna WF can also contain nested WFs in a hierarchical manner. In this way, a set of simple WFs easily allows to design more complex WFs. These WFs can then be shared, reused, and adapted to new needs.

3 CONTRIBUTION

We propose to use WF of services as the interoperability layer among several services. In addition, we propose to integrate the G-DEVS/HLA engine as a specific WF engine. G-DEVS is a formalism based on a state machine automaton. WFs differ from state machines as state machine can be cyclic graphs while WFs are usually acyclic. WF proceeds down different branches until done. Thus, using G-DEVS coupled to another WF engine to process a WF could benefit from the DEVS formalism while keeping the top to bottom behaviour of the main WF manager. Interoperability between WF engines and applications are done using web services.

3.1 Workflow Orchestration Architecture

The Figure 2: Workflow global orchestration architecture.Figure 2 presents the proposed orchestration architecture that is based on the WF architecture introduced by the WfMC [WfMC 99] and [WFMC 05] . We detail in the following the specific architecture tailored to “timed” WF.

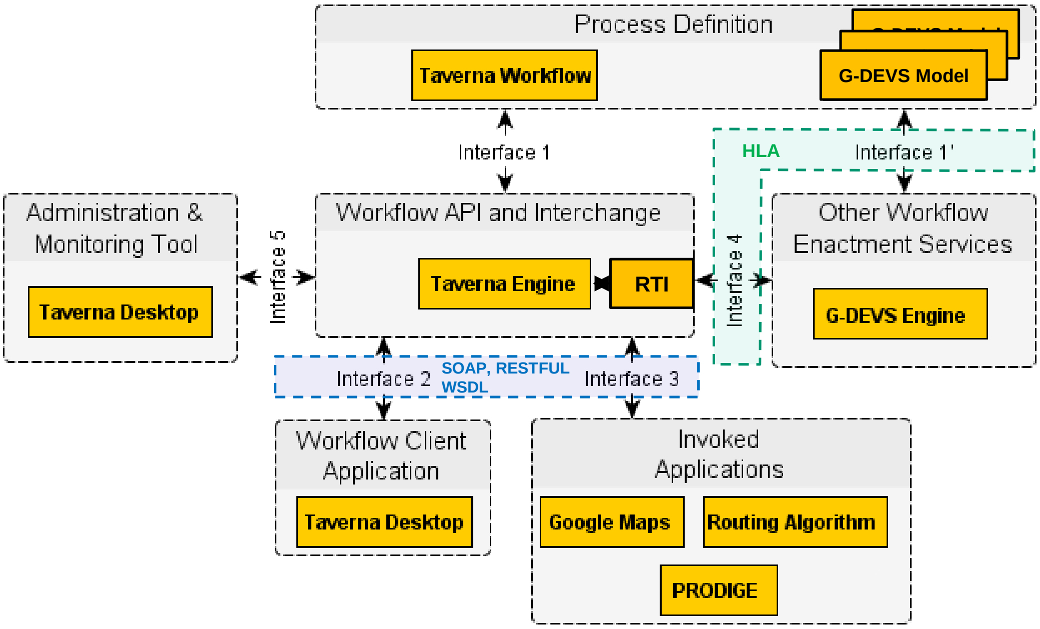

Figure 2: Workflow global orchestration architecture.

We propose to use Taverna and G-DEVS/HLA as the process definition formalism to express WFs. Taverna WF represents the main WF that organizes all tasks and enables interoperability between services. Taverna WF process definition will be executed by the Taverna Engine (Interface 1). G-DEVS model will be executed by the G-DEVS Engine (Interface 1’) as another WF enactment services. Communication between both engines (Interface 4) will be garanteed by HLA/RTI. Taverna interprets G-DEVS WF events through HLA based Interface 4 and enables the interoperability with other services using RESTful or SOAP/WSDL Web services protocols through Interface 3 and with users through the use of the Taverna Desktop through Interface 2.

3.2 G-DEVS/HLA Workflow Model

In previous work [Zacharewicz et al., 2008], several G-DEVS models have been already coupled thanks to a HLA connection. The idea was to establish distributed simulation for G-DEVS Models but also to open G-DEVS to interoperability with other software components.

In the PRODIGE platform project [Zacharewicz et al., 2011] the main components of a transport and logistic system that interact in a WF have been specified using G-DEVS models. The goal was to study the hardware and human behaviour and to test their dialog with the platform. For instance smartphone behaviour has been formalized. The behaviour of the smartphone was to take into account hardware delay due to 3G or Edge bandwidth during communication between the platform and users. In addition the driver responsiveness with its environment and the truck movement were also simulated. In this approach the simulation synchronization was given by an HLA RTI.

In [Tu et al., 2012], the Portico RTI has been used to facilitate the connection between RTI and the calls to web services. This RTI has been reused in our work.

We propose the use of the GDEVS/HLA interoperability with other software components and propose to couple it with the Taverna tool. The Figure 3 focuses on the proposed WF simulation architecture (regarding Figure 2 it zooms on Taverna, RTI and G-DEVS). The HLA RTI becomes the information scheduler and the clock leader of the simulation. More details are provided in the next section.

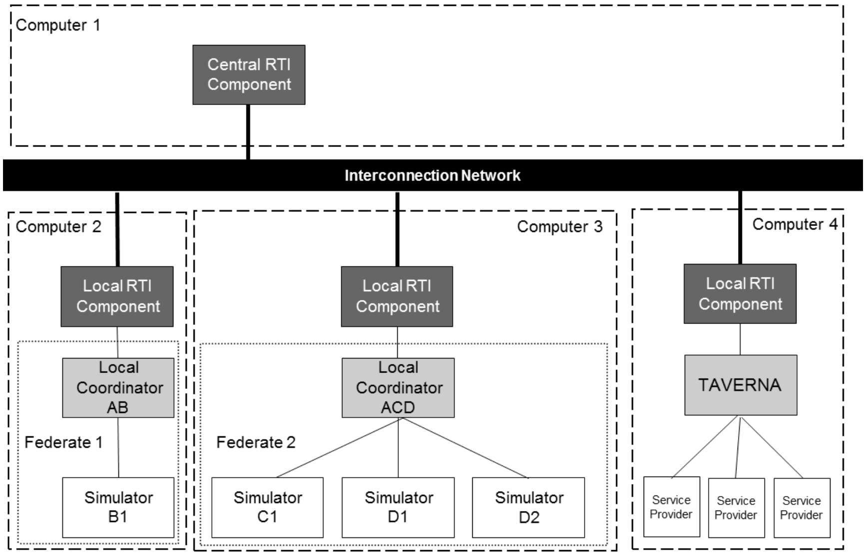

Figure 3 Workflow Simulation main components view

3.3 G-DEVS/HLA, Clock and Message Sorting

In this paper the functional interoperability with web service dealers is mainly assumed by the Taverna engine that calls the services and links the different applications. Taverna allows defining a sequence of service calls. Nevertheless this tool (as most service WF builder/runner) does not allow defining the time constraints (that exists in real situation) in the sequence and does not provide time synchronization for the simulation of services calls in a defined sequence. For instance, the access to a data base too early or too late can be a problem, i.e. with obsolete values or too recent values. This problem can arise when parallel process is executed. To address this issue, two options have been considered.

On the one hand, using a Run Time Infrastructure tool (RTI) to build an HLA federation as used in [Al-Zoubi and Wainer, 2010a]. This option requires the use of a systematic and direct connection of all components to the RTI. This setup, with RTI as a mediator of all messages exchanged, can be interesting from the interoperability point of view can cause overheads and slow down the communication like discussed earlier in [Al-Zoubi and Wainer, 2010b].

On the other hand, the idea proposed in this research is to use the Taverna WF interoperability facility as the main interoperability layer between services (including applications and simulations). It is executing the main scenario script. In addition we propose to use a G-DEVS/HLA to be the time based message scheduler for the WF scenario. In that case the use of the RTI is not systematic; it is only solicited when the WF is communicating with a simulated component. On demand, the RTI and the GDEVS model are playing the behaviour of the simulated component with time spent achieving an action in order to reproduce real time reaction delay. [Zacharewicz et al., 2011] and [Tu et al., 2012] already uses G-DEVS models and HLA RTI simulators to simulate the behaviour of several components by mixing HLA and web services in a previous study. The distributed simulation principle of [Tu et al., 2012] is based on the original pessimistic algorithm described in [Chandy and Misra, 1979], but adding recent advances on lookahead described in [Zacharewicz et al., 2008]. The RTI is defining the ordering of the actions regarding their occurrence time. It stores the information before releasing them regarding the scenario definition played in Taverna. It can be also considered as the script clock and blocker/releaser of the simulation. Regarding time synchronisation, the GDEVS/HLA models are already prepared since [Zacharewicz et al., 2008] to inform the RTI about their Lower Bound on Time Stamps (LBTS) [IEEE, 2000] to compute the Lookahead (minimal treatment delay) and unblock the simulation. Taverna was not defined for that. The idea has been to define minimum treatment duration in each workflow step to be communicated to the RTI. Thanks to this information taken by the RTI as the Taverna LBTS, the distributed simulation can be run without deadlocks.

In detail, this paper proposes that the RTI collects simulation messages, sorts them and triggers the services call right in time to the applications or forwarding the message to the G-DEVS models that simulate the behaviour of the WF components according to defined scenario, i.e. a timed sequence of service calls. This model can receive messages both from the server as a service answer or from a G-DEVS model that sends an output message as a simulation result of a local behaviour. The messages received from the server are service answers. They possess time stamp information to be used by the RTI to add the message at the right place in the queue. Then depending on the execution state of the global clock it will sort the message and direct it to the proper receiver. The RTI status can be treating a message or be available. In the first case, the approach is inspired from the conservative algorithm of [Chandy and Misra, 1979]. It is based on the G-DEVS/HLA algorithm, proposed in [Zacharewicz et al., 2008], in particular if a message arrives late. The message temporary blocks the simulation and will not be ignored. Then simulation is unblocked when it passes to process the next message, it shows the interest of providing an accurate value of LBTS to the RTI. The receiver can be a web server trough Taverna. In that case it prepares an output message. This output message is addressed to Taverna that transforms it to service call and then triggers the web service server. If the message is addressed to a G-DEVS model to trigger component behaviour, the message is sent through the RTI to the appropriate G-DEVS component using the coupled model structure. In the second case (no input event to be treated) the RTI is waiting inputs.

3.4 GDEVS/HLA Taverna Interoperability

The interoperability between G-DEVS model and web applications, ensured by Taverna WF plus the HLA RTI, is concretely tested in this section. The Figure 4 presents the sequence diagram among 2 Taverna WFs, a generic G-DEVS model entitled “Clock”, used to represent a basic time dependent behavioural model; the HLA communication and an application. Taverna WFs represent 2 experimentations executed in parallel to test the application. The G-DEVS model represents the clock scheduling and is waiting for 2 Taverna instances (“nb=2”).

The sequence is expressed as follow:

1. The Taverna WF model 1 instantiate and ask the GDEVS clock model to be woken-up at 8:45.
2. The Taverna WF model 2 instantiate and ask the GDEVS clock model to be wock-up at 8:30.
3. The clock model wake-up the Taverna WF model 2; time = 8:30.
4. The Taverna WF model 2 invokes the “setResources” service on the application and then ask the GDEVS clock to be woken-up at 10:00
5. The clock model wake-up the Taverna WF model 1; time = 8:45.
6. The Taverna WF model 1 invokes the “getResources” service on the application and then ask the GDEVS clock to be woken-up at 10:15

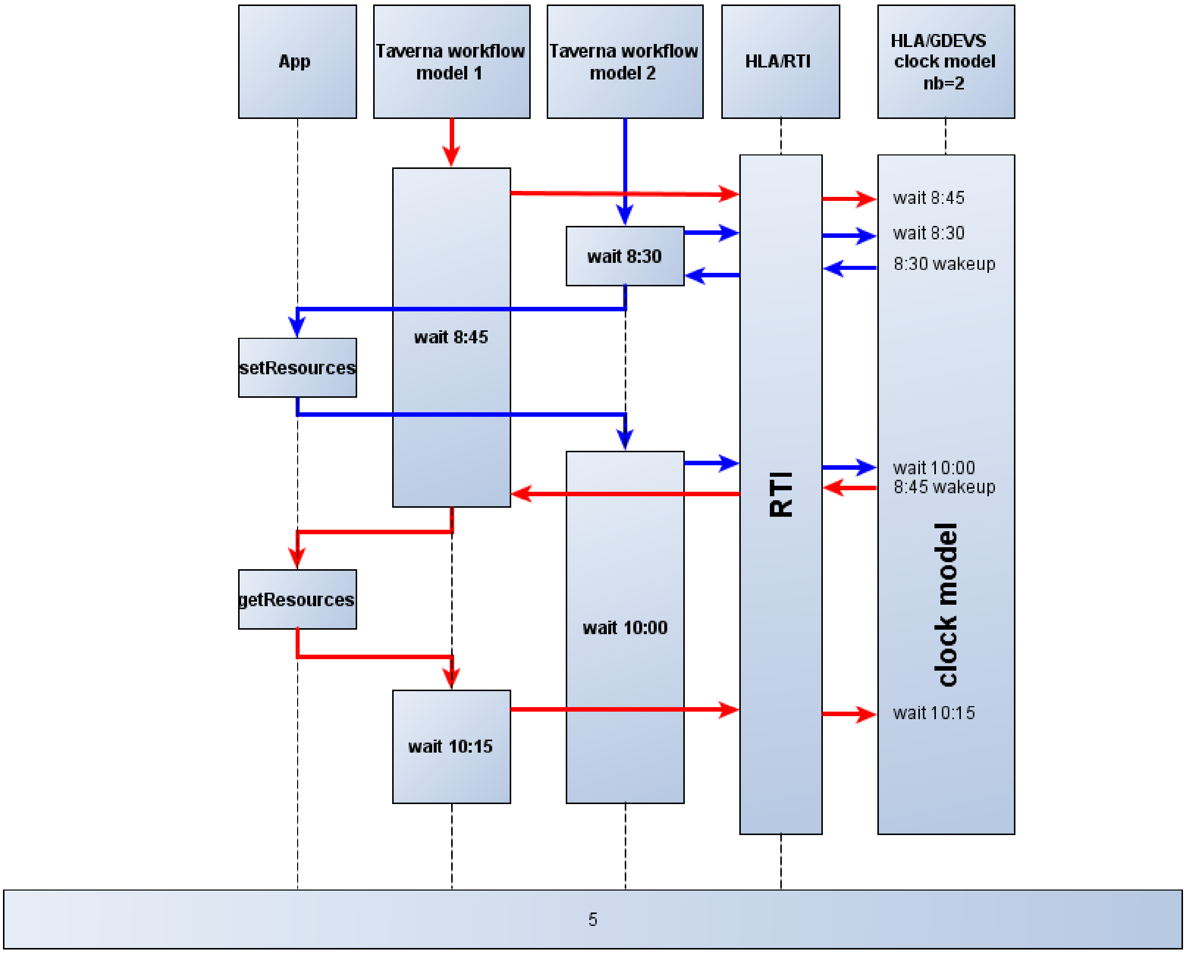

Figure 4 Interoperability sequence diagram.

This interoperability provided by Taverna and HLA allowed us to schedule independent WF running in parallel. In the next section, we experiment interoperability in a real case study through the PRODIGE project.

4 EXPERIMENTATION

The contribution presented in section 3 is applied to the product transportation use case through the PRODIGE Project. We will present the PRODIGE Project, the use case scenario and finally the experimentation framework.

4.1 PRODIGE Project context

The PRODIGE project aims to prepare the future of physical products transportation, placing the reflection at the organizational level that controls the flow of merchandises in order to provide a technical and organizational solution helping the reduction of the travelled distance, optimization of the tours and transported volumes and taking into account new issues related to sustainable development.

The base of the work, proposed in this paper, starts from a transportation Web application released in the project. This platform is composed of a server where several truck users are remotely contacted to display their positions thanks to GPS and GSM communication. The server is offers an algorithm to optimize truck routing. It exposes its methods through the use of SOAP Web services in order to promote interoperability (set a tour, view the results, etc.). The idea is to test the function of the PRODIGE platform regarding a sequence of dynamic calls. For this purpose, a simulation tool making the WF alive is required in order not to launch all the trucks on the roads for each test.

A scenario in PRODIGE can have several objectives:

- Quantitative: calculating and comparing several variables such as the number of kilometres travelled by products or the amount of CO2 emissions produced for a set of deliveries
- Qualitative: following the different steps of the delivery of a product (e.g. respect of delivery times, compliance with cold chain, etc.)
- Analytics: observing a special non understood case, difficult or impossible to reproduce with the real system, mainly for scientific purposes.

In this objective, demonstration scenarios are added, to explain PRODIGE to public audience and to follow graphically the movement of vehicles depending on the scenario chosen.

4.2 Use case scenario

The PRODIGE platform is easy to setup (a server connected to internet to provide SOAP services), but it’s very expensive to rent a truck and a driver. We decided to simulate the driver/truck and setup a real PRODIGE platform. We propose a simple scenario case to illustrate the simulation of truck, driver, and smartphone regarding the global system. *Earlier in the day, the driver connects to the PRODIGE platform and retrieves information about the tour he was assigned. He leaves the deposit and drive to the first client. He loads the goods then leads to the second client where he will unload the goods. Finally, he returns to the depot at the end of the tour.* For each step described above, there is some communication with the PRODIGE platform to ensure traceability and reactivity of the PRODIGE platform. To test this scenario, we need to setup the PRODIGE platform and drive a truck across 2 destinations. The behaviour of the driver/truck can be formalized in a WF which would pass itself for a real truck to the PRODIGE platform. To send GPS tracking data to the PRODIGE server, the WF uses Google Map[1] services to get the direction from a place to another. Then, we can get GPS coordinates of every little part of the road. The WF, like the real application embedded in a truck, sends a couple of GPS coordinates every 30 seconds to the PRODIGE platform and information about load and unload.

---

[1] http://maps.google.com

At this point, we have a basic scenario involving only one truck and executing in real-time. This means that a delivery from Bordeaux to Paris will take 5 hours.

Authors introduce the main contribution of the paper by using the G-DEVS "clock" model presented in section 3; we can synchronize the logistic WF with virtual time. That means a delivery from Bordeaux to Paris will take few minutes. Every 30 seconds in virtual time, the WF sends GPS coordinates (which include timestamp) to the PRODIGE platform. The PRODIGE platform can then display the tour exactly as if it had taken 5 hours thanks to the timestamp embedded with the GPS coordinates, e.g. departure from Bordeaux at 8:00 AM and arrival in Paris at 1:00PM.

From now on, we can create an advanced scenario involving several drivers/trucks. Each driver/truck is represented by a new WF instance, which keeps the WF very simple. WFs instances run in parallel and are synchronized thanks to the G-DEVS clock model. This use case demonstrates the benefits from mixing WF and simulation and how WF of services like Taverna can handle interoperability between application services and simulation.

We created several data input sets as well as several WFs to simulate different situations to experiment the PRODIGE solution before putting it on the market. Packages must be picked up and delivered regarding the two following situations:

- The delivery time windows are wide enough for it to be feasible with a single truck.
- The delivery time windows overlap and several trucks are needed to make the delivery on time.

Those two situations are done using the same generic WFs. We built another WF to take into account hazards such as traffic jams or a breakdown. Indeed, in those cases the WF must take into account specific decision that could involve creating new delivery.

4.3 Experimentation Framework

We have implemented the architecture and concept described in the previous sections. Figure 5 represents the solution framework. The virtual experiment is defined using Taverna WF and G-DEVS simulation. The Taverna WF mimics the behaviour of managers, clients and drivers while the G-DEVS simulation act as a WF scheduler. Communication between Taverna and G-DEVS are done through HLA RTI. The Taverna WF communicates with the PRODIGE application and Google Maps through Web service. The real experiment needs people to manage the PRODIGE application (manager, clients) and drive the trucks (drivers). Communication between people and PRODIGE is done using a light web application (manager, client) or a mobile application on smart device (driver).

At simulation time, during transition due to message treatment by the G-DEVS models, an output message from Taverna is frequently generated in order to give the order to refresh the positioning of the trucks and product to the server according to the roadmap and geographical information extracted from Google maps. During the setting of the simulation the pace can be tuned in order to accelerate the simulation execution. Also, at any simulation time the execution can be stopped to show a particular case.

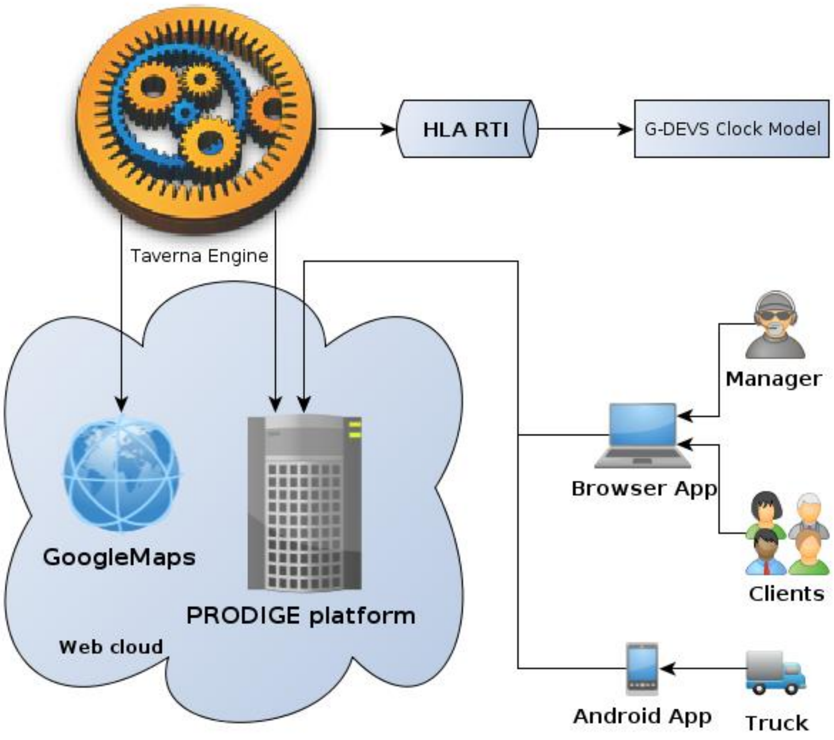

Figure 5 PRODIGE and simulation framework architecture.

The Taverna WF plays the role of each actor (manager, clients, and drivers) and interacts with the PRODIGE platform. Several WF instances are executed in parallel for each driver involved. The WF retrieves the information needed on the PRODIGE platform, on Google Maps and using G-DEVS clock model to mimic the behaviour of a real truck. The result of the execution of this WF is directly visible in the PRODIGE web application on which you can view the current path of a truck making its tour in the Bordeaux area (France), as shown on Figure 6.

5 CONCLUSION

This work has permitted to introduce a new architecture for simulation of WF including time constraints. It has been validated on logistics and transportation platform. It recalled existing works that already proposed to use the G-DEVS formalism for the description of the logistic platform components. Then, it introduces the Taverna tool that will be the interoperability link to connect the services and the simulation components. Then it describes the G-DEVS model that has been proposed to serve as the clock ordering component in the system since Taverna and more generally the services do not consider the time synchronization. The main demonstration of this paper was to show the interest of interoperability in such simulation. Here the approach was still pragmatic but the future works should make the G-DEVS Clock model more generic so as to be reused in several service handling tools.

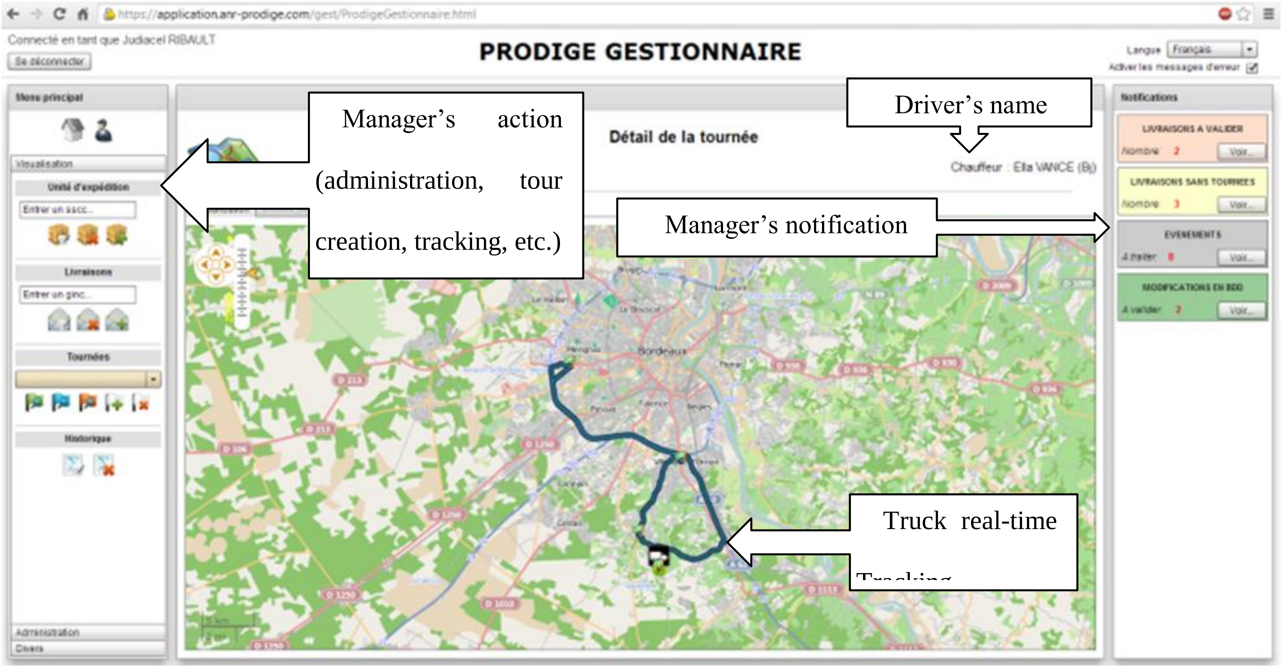

Figure 6 PRODIGE Web application.

6 ACKNOWLEDGEMENT

This research was partially funded by the PRODIGE (ANR-09-VTT-09-01) project and Region Aquitaine Funding for Research.

7 REFERENCES

Al-Zoubi, K., & Wainer, G. (2010a, December). *Managing simulation workflow patterns using dynamic service-oriented compositions.* Proceedings of the 2010 Winter Simulation Conference (WSC), (pp. 765-777). IEEE.

Al-Zoubi, K., & Wainer, G. (2010b, December). *Rise: Rest-ing heterogeneous simulations interoperability.* In Simulation Conference (WSC), Proceedings of the 2010 Winter (pp. 2968-2980). IEEE.

Altintas I, C Berkley, E Jaeger, M. Jones, B. Ludascher, and S. Mock. Kepler: An extensible system for design and execution of scientific workflows. In Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on, pages 423–424. IEEE, 2004.

Chandy, K. M., & Misra, J. (1979). Distributed simulation: A case study in design and verification of distributed programs. Software Engineering, IEEE, (5), 440-452.

Chen, D., Doumeingts, G. 2003. European Initiatives to develop interoperability of enterprise applications - basic concepts, framework and roadmap, *Journal of Annual reviews in Control*, 27, no.2, 151-160.

Churches David, Gabor Gombas, Andrew Harrison, Jason Maassen, Craig Robinson, Matthew Shields, Ian Taylor, and Ian Wang. Programming scientific and distributed workflow with Triana services. Concurrency and Computation: Practice and Experience, 18(10):1021–1037, August 2006.

Curcin V and M Ghanem. Scientific workflow systems can one size fit all? Biomedical Engineering Conference, 2008.

Deelman Ewa, Dennis Gannon, Matthew Shields, and Ian Taylor. Workflows and e-Science: An overview of workflow system features and capabilities. Future Generation Computer Systems, 25(5):528–540, May 2009.

Fujimoto, R.M. (2000) '*Parallel and Distributed Simulation System*'. Ed. Wiley Interscience, New York.

Giambiasi N., Escude B., Ghosh S. (2000) GDEVS A Generalized Discrete Event Specification for Accurate Modeling of Dynamic Systems. SCS Tr, 17(3) 120-134.

C.A. Goble and D.C. De Roure. myExperiment: social networking for workflow-using e-scientists. In Proceedings of the 2nd workshop on Workflows in support of large-scale science, pages 1–2. ACM, 2007.

Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M. R., Li, P., & Oinn, T. (2006). *Taverna: a tool for building and running workflows of services*. Nucleic acids research, 34(suppl 2), W729-W732.

IEEE std 1516.2-2000. IEEE Standard for Modeling and Simulation (M&S) High Level Architec-ture (HLA) - Federate Interface Specification, In-stitute of Electrical and Electronic Engineers.

IEEE std 1516.3-2003. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federation Development and Execution Process (FEDEP) The Institute of Electrical and Electronic Engineer.

IEEE std 1516-2010, 2010. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) -- Framework and Rules, New York: Institute of Electrical and Electronic Engineers

Dahmann Kuhl, F.,, J., & Weatherly, R. (2000). Creating computer simulation systems: an introduction to the high level architecture. Prentice Hall PTR.

OMG, "Business Process Model and Notation (BPMN) version 2.0" document number: formal/2011.

Ribault, J., & Wainer, G. (2012, March). Using workflows and web services to manage simulation studies (WIP). Proceedings of the 2012 Symposium on TMS/DEVS Integrative M&S Symposium (p. 50).

Richardson, L., & Ruby, S. (2007). RESTful web services. O'Reilly Media, Incorporated.

Rybacki Stefan, Jan Himmelspach, Fiete Haack, and Adelinde M Uhrmacher. Worms- a framework to support workflows in m&s. In S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, editors, Proceedings of the 2011 Winter Simulation Conference, Piscataway, New Jersey, 2011. Institute of Electrical and Electronics Engineers, Inc.

Missier Tan, W.,, P., Madduri, R., & Foster, I. (2009). *Building scientific workflow with taverna and bpel: A comparative study in cagrid.* In Service-Oriented Computing–ICSOC 2008 Workshops (pp. 118-129). Springer Berlin/Heidelberg.

Tu, Zhiying; Gregory Zacharewicz; David Chen Building a high-level architecture federated interoperable framework from legacy information systems, International Journal of Computer Integrated Manufacturing, Stephen Newman, 2012, pp. 1-20, DOI : 10.1080/0951192X.2011.646306

Weske, M. (2012). Business Process Management Archi-tectures. Business Process Management, 333-371.

Workflow Management Coalition. 1999. Terminology & Glossary. WfMC-TC-1011, 3.0.

Workflow Management Coalition. 2005. Workflow Process Definition Interface -- XML Process Definition Language (XPDL). WFMC-TC-1025, Oct.

Jia Yu and Rajkumar Buyya. A taxonomy of workflow management systems for grid computing. Journal of Grid Computing, 3(3-4):171–200, January 2006.

Zacharewicz G., Frydman C., Giambiasi N. (2008) G-DEVS/HLA Environment for Distributed Simulations of Workflows. Simulation 84(5): 197-213

Zacharewicz G., Deschamps J.C., François J., (2011). Distributed platform for advanced freight transportation systems, Computers in Industry 62, 6 597-612.

Zeigler, B. P., Praehofer, H., & Kim, T. G. (2000). Theory of modeling and simulation: Integrating discrete event and continuous complex dynamic systems. Ac. Pr.

*Highlights (for review)

Workflow Interoperability with G-DEVS/HLA

Distributed Simulation of Workflow

Combining Service Calls and Simulation models

Adding time constraints in Web Service Orchestration

Application to a Transport Simulation Platform