Title: Time-Based Orchestration Of Workflow, Interoperability With G-Devs/Hla

Author: Judicaël Ribault Gregory Zacharewicz

PII: S1877-7503(14)00098-2
DOI: http://dx.doi.org/doi:10.1016/j.jocs.2014.07.002
Reference: JOCS 295

To appear in:

Received date: 24-2-2014
Revised date: 11-6-2014
Accepted date: 23-7-2014

Please cite this article as: J. Ribault, G. Zacharewicz, Time-Based Orchestration Of Workflow, Interoperability With G-Devs/Hla, *Journal of Computational Science* (2014), http://dx.doi.org/10.1016/j.jocs.2014.07.002

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 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 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.

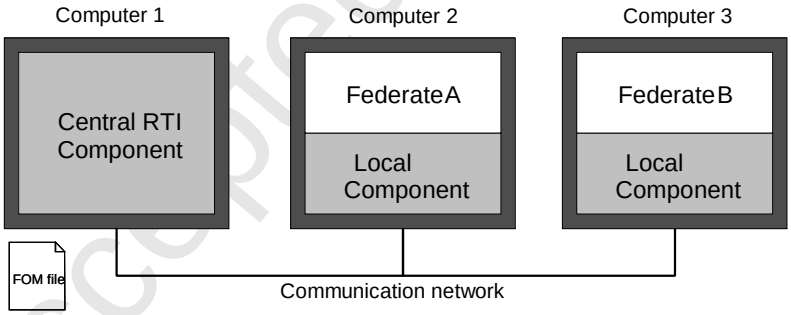

Figure 1. HLA implementation components

A federate can, through the services proposed by the RTI, "Publish" and "Subscribe" to a class of shared data. "Publish" allows diffusing 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 validating 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 [Wolstencroft et al., 2013] 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.

Any workflow management system that enables the interoperability between services could be used. Depending of the objectives, we will focus on the execution time, debugging options or ease of use of the workflow management system. We decided to use Taverna to demonstrate the feasibility of our methodology because Taverna eases the interoperability with other services and the data flow modelling compare to other workflow management system we studied.

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 workflows. These workflows can integrate many different resources in a single experiment.

A Taverna workflow can contain several 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 service to have a Taverna workflow nested within another hierarchically. In that way, simple workflows can be used as a basis for more complex ones
- A service to access local tools within a workflow. Thus, we can use local tools within a Taverna workflow.

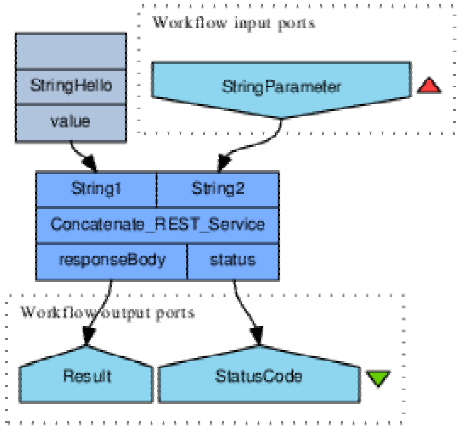

Figure 2. Taverna workflow with a REST Service.

In Taverna, a service can take input and produce output. The workflow input can be part of the workflow or can be given prior to the execution of the workflow. For example, the Taverna RESTful service takes in input various data, and it returns a status code and a response. Figure 2 shows a simple workflow containing one REST service ("Concatenate REST Service") taking two inputs: a string containing "Hello" (top left) and a string parameter to be given at runtime.

This workflow will simply concatenate to the string “Hello” the value passed at runtime parameter (for example “World”). The invocation of the REST service provides two outputs: a “status” and a “responseBody”. The status (404 if the service was not found, or 200 if everything went well) is connected to the “StatusCode” output of the workflow. The responseBody (for example “Hello World”) is connected to the “Result” output of the workflow.

In contrast, a WSDL service will find automatically, thanks to the WSDL file, the number and type of input and output. Figure 3 represents a Taverna workflow with a WSDL service in green in the middle of the figure. The service is available in Taverna after the addition of the URL of the WSDL file (such as http://xxx.xxx.xxx.xxx:8080/WS-PRODIGE/services/Identification?wsdl). Taverna offers the possibility to automatically format the input and output based on the type of parameters required by the Web service. In this example, the Web service "identificationChauffeur" that allows a driver to identify within the PRODIGE application takes as input a data type 'identificationChauffeur_input' that encapsulates ‘id’, ‘imei’, and ‘pwd’ input. The Web service "identificationChauffeur" produces as output a data type 'identificationChauffeur_return' that encapsulates various data such as firstName, lastName, login, etc.

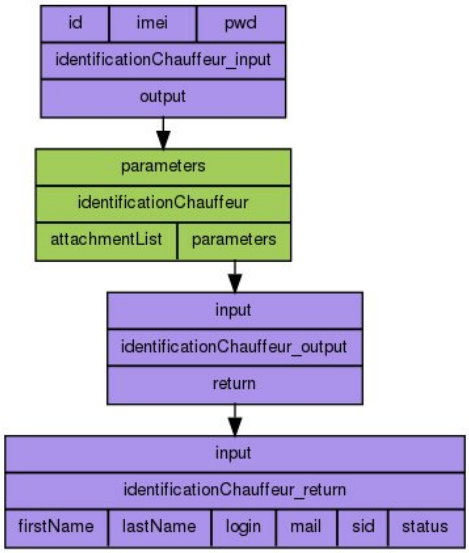

Figure 3: Taverna WSDL service.

Workflows 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. Taverna offers several user interfaces with this purpose:

- Ask: opens a box so the user can enter text.
- Choose, Select: lets the user select a value among a list of values.
- Select File: lets the user select a file in the system.

- Tell, Warn: gives a message to the user.

A Taverna workflow can contain nested workflows. Thus, it is possible to create a parent workflow that contains several workflows. Several workflows can be combined together to obtain more complex workflows that do not need the external inputs and are fully automated.

2.6 Coupling Web Services and Simulation Existing works

Taverna is used to draw and run a sequence of service calls. Nevertheless this tool (as most service WF builder/runner) does not allow defining time synchronisations and constraints (that still exists in real situation due to reaction delay of software, material or human), so it does not support time synchronization required to compose real services calls and simulated ones in a common sequence. For instance, too early or too late access to a data base can be a problem, i.e. with using obsolete values or too recent values. This problem can arise when parallel processes are executed. To address this issue, several options have been considered in previous works to combine simulation with service calls.

Main approaches use a global synchronisation component. For instance, a Run Time Infrastructure tool (RTI) has been used to run an HLA federation of service calls in [Al-Zoubi and Wainer, 2010a]. This option requires the use of a systematic and direct connection of all components to the RTI. Also, [Zacharewicz et al., 2011] and [Tu et al., 2012] have used G-DEVS models and HLA RTI to simulate the behaviour of several components by mixing HLA and web service calls. The distributed simulation principle of [Tu et al., 2012] was based on the original pessimistic algorithm described in [Chandy and Misra, 1979], adding more recent advances on lookahead described in [Zacharewicz et al., 2008]. These set-ups, with RTI as mediator of all messages exchanged, can be interesting from the interoperability point of view but they can cause overheads and slow down the communication as discussed in [Al-Zoubi and Wainer, 2010b]. In addition, none approach is truly dedicated to integrate human, material and software simulated behaviour with service calls.

3 CONTRIBUTION

From the state of the art, it appears that the efficient combination of service calls and simulated components is not fully achieved. We propose in this article to use a WF of services handler as a global interoperability backbone among heterogeneous services call. In this system, the call can be either done to existing web services or to virtual WF components. In that last case, we run a G-DEVS/HLA engine to simulate the behaviour of the resource. In detail, we found that general state machines of G-DEVS and WFs of service handlers that proceeds down different branches until done like an acyclic graphs are complementary to describe general WFs behaviour can be cyclic. Thus, the WF engine could benefit from using local G-DEVS (simulating local cyclic behaviour in the sequence) while keeping the top to

bottom execution of the main WF manager. Interoperability between WF engines and applications is achieved using web services for non-temporal execution and HLA for G-DEVS components for time stamped information exchanged.

3.1 Global Workflow Orchestration Architecture

The Figure 4 presents the proposed global orchestration architecture that is based on the WF architecture introduced by the WfMC [WfMC 99] and [WFMC 05]. We detail in the following this specific architecture tailored to "timed" WF.

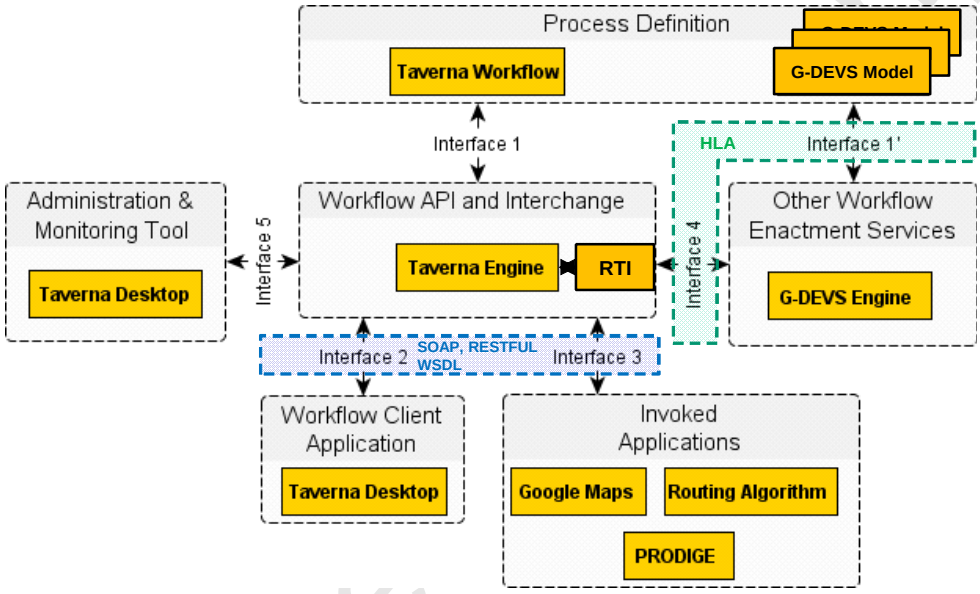

Figure 4. Workflow global orchestration architecture.

In this work, Taverna is used as the macro process orchestrator to express and run WFs and G-DEVS/HLA as the local behaviour simulator of sub process or task. As a result, Taverna WF implements the backbone WF that organizes and sequences all tasks and enables interoperability between different web services. HLA insures the interoperability and time synchronisation between Taverna and G-DEVS. In detail, Taverna WF process definition is executed by the Taverna Engine (Interface 1). G-DEVS models are triggered by Taverna trough the RTI (Interface 4) and executed by the G-DEVS Engine (Interface 1') as another WF enactment services. Bask to Taverna engine (Interface 4), HLA/RTI guaranty again to interpret events from G-DEVS WF. Taverna enables the interoperability with other services using RESTful or SOAP/WSDL Web services protocols through Interface 3. Interface 2 allows users to interact with WF using Taverna Desktop. Interface 5 can track WF execution step by step using the Taverna Desktop application.

3.2 HLA based Architecture for G-DEVS and Taverna Workflow

This work is reusing the concepts proposed in [Zacharewicz et al., 2008] that were consisting in connecting distributed

G-DEVS WF models thanks to a HLA linking. The goal was to establish distributed simulation for G-DEVS Models but also to open G-DEVS to interoperability with other software components. Also the connector developed in [Tu et al., 2012] for the open source Portico RTI [Portico, 2009] has been reused to facilitate the connection between RTI and the calls to web services format used in Taverna.

In the proposed architecture the use of the HLA permits interoperability and time synchronisation between G-DEVS, the Taverna tool and potentially other software components. The Figure 5 is a HLA/RTI centric view; it focuses on the proposed WF simulation architecture part. Regarding Figure 4, it zooms on Taverna, RTI and G-DEVS interface 1' and 4. The HLA RTI is the information scheduler and the clock keeper of the simulation. More details are provided in the next section. Regarding HLA technical components, two categories of federate have been distinguished.

On one side the G-DEVS simulator component is integrated in specific federate (Figure 5 federate 1,2). For that, we reused works from [Zacharewicz et al., 2008]. On the other side, Taverna that was not defined in the objective of using timed information for simulation and was not HLA compliant has been extended by a java wrapper to become a federate (Figure 5 federate 3).

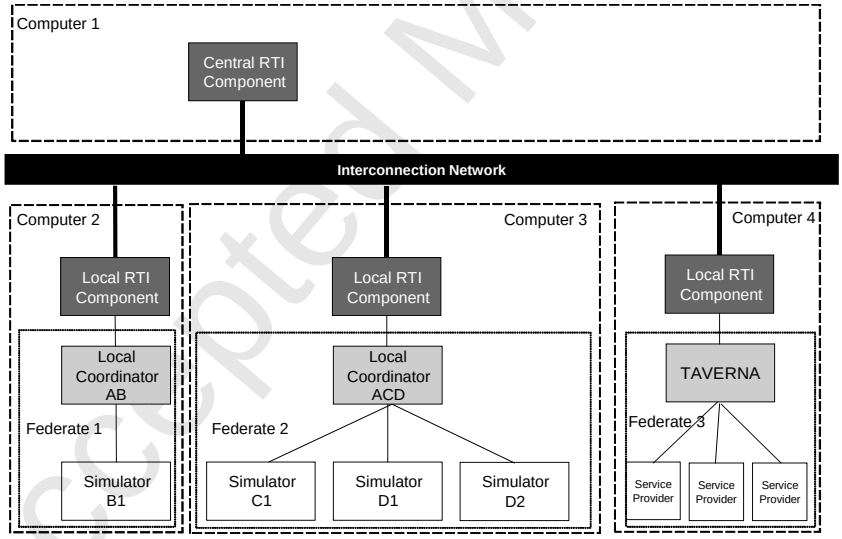

Figure 5. Workflow Simulation main components view

3.3 G-DEVS/HLA, Clock and Message Sorting

The interoperability with web service providers is assumed by the Taverna engine that links the data. But to run mixed real/virtual WF scenario, we need to combine Taverna with G-DEVS/HLA to obtain a time based message scheduler.

Technically, we bridged Taverna and G-DEVS engine using HLA standard implementation. But contrarily to previous approaches, the use of the RTI is not systematic; it is solely solicited when the WF is communicating with a simulated

component. On demand from the RTI, the G-DEVS model is playing the behaviour of the simulated component with time spent achieving an action in order to reproduce real time reaction delay.

We reused the G-DEVS component extended in [Zacharewicz 2008]. It adds a federate ambassador component to implement to RTI communication functions and to compute the Lookahead (minimum treatment delay) of each local G-DEVS model to inform the RTI about the local model Lower Bound on Time Stamps (LBTS) [IEEE, 2000]. The G-DEVS federates are both time regulating and time constrained.

The main contribution for simulation was to add primitives to to Taverna messages in order to publish / subscribe HLA objects, to stamp them with HLA time stamp and to define minimum treatment duration in each workflow step to be communicated to the RTI. This time amount is quantum regarding other time value in G-DEVS models. Thanks to this information taken by the RTI as the Taverna LBTS, the distributed simulation can be run without deadlocks. The Taverna federate is both time regulating and time constrained.

At run time, according to the HLA Time Management specification [IEEE 2010], the RTI collects simulation messages, sorts them and triggers federates right in time. The RTI is ordering the message exchange between Taverna and G-DEVS regarding their occurrence time. It stores the information before releasing them to Taverna or G-DEVS according to the global scenario definition played in Taverna and time constraints. It can be considered as the script clock and block/releaser of the simulation. The approach is running a conservative algorithm based on [Chandy and Misra, 1979] (with strict time constraints) specialized to G-DEVS/HLA proposed in [Zacharewicz et al., 2008].

We detail here and in figure 6 the communication steps of the architecture. All federates start by communicating communicate their LBTS. Then Taverna imitates the process by sending query to web applications or forwarding a query message to a G-DEVS model (Figure 6 Step 1) through the RTI. Messages received at RTI from Taverna possess time stamp information to be used to insert the message at the right place in the queue (Figure 6 Step 2). Then depending on the state of the global clock it will sort the next message and direct it to the proper receiver (using message publish and subscribe mechanism). The RTI status can be either treating a message or waiting for inputs. In particular if a federate is late for reporting to RTI about its current state and its next message to be locally treated, this pending information temporary blocks the RTI execution, so no message is ignored in this conservative approach. Then simulation is unblocked as soon as RTI gets the information from all federates to process the next earlier message (it shows the interest providing a value of LBTS to the RTI). The receiver of RTI message can be a G-DEVS or a web 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. We note that G-DEVS can receive triggering messages either from the server as a service query to simulated component (Figure 6 Step 3) or from another G-DEVS model that sends an

output message as a simulation result of a local behaviour. The message back from simulation is addressed by the RTI to Taverna (Figure 6 Step 4) that transforms it to service call and then triggers the web service server (Figure 6 Step 5). Finally, in the case RTI has no event to release, it goes to waiting inputs state.

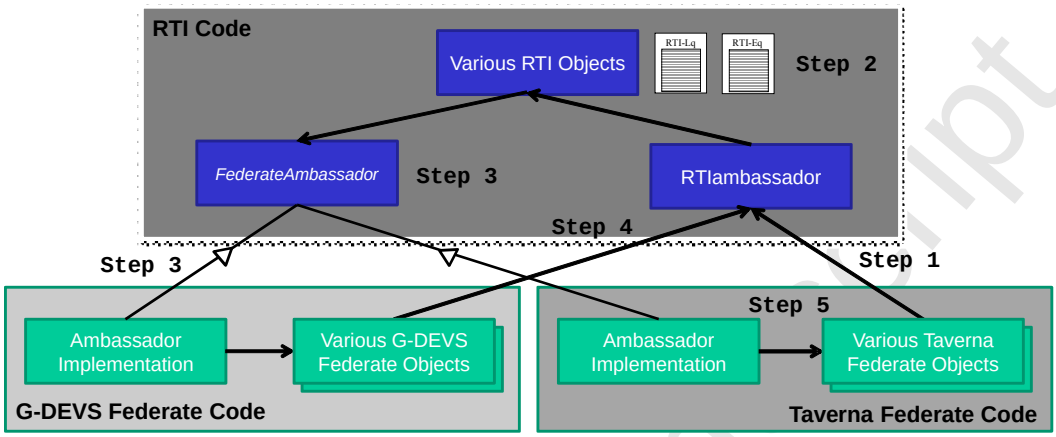

Figure 6. Code components GDEVS/TAVERNA in HLA Architecture

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 7 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 instantiates and ask the GDEVS clock model to be woken-up at 8:45.
2. The Taverna WF model 2 instantiates 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

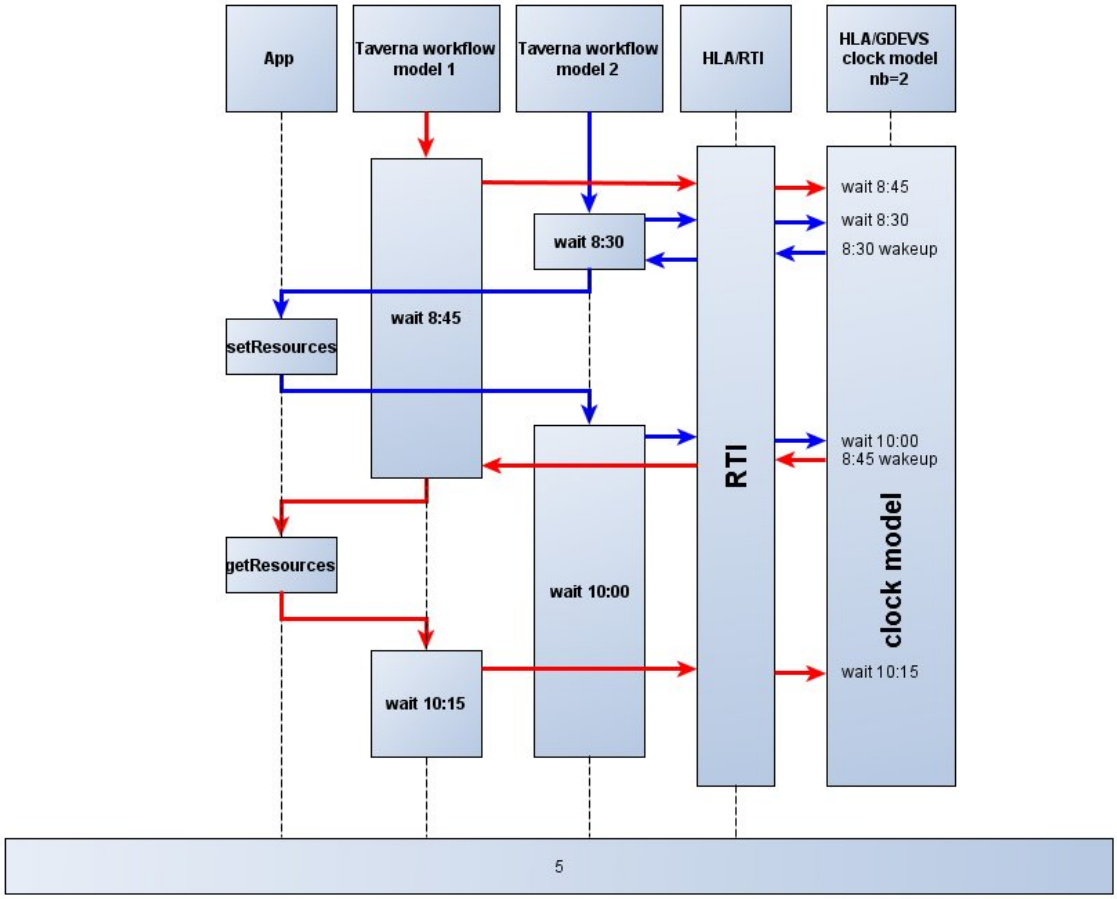

Figure 7. Interoperability sequence diagram.

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 has been applied to one use case of product transportation. It has been implemented as a part of PRODIGE platform project [Zacharewicz et al., 2011], where the main components of a transport and logistic system that interact in a WF have been specified using G-DEVS models (section 4.2). The goal was to study the hardware and human behaviour and to simulate their dialog with the platform. For instance smartphone behaviour has been formalized. The behaviour of the smartphone was taking 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.

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 using 2 main axes of development. The first one is 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. The second one is the optimization of the tours and transported volume, 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 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. 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 Expectations

It is important to define, prior to the experimentation, some objectives to quantify the success of the contribution. Through this experimentation, we expect to validate several points.

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

- Feasibility: Is it possible to use a workflow to define a scenario, using the workflow layer to enable interoperability between enterprise services, and orchestrate them using a DEVS engine?
- Reduce time to market: Is this approach satisfying to reduce the time to market of new services?
- Ease of use: Is this approach suitable to be handled by non-expert such as SME manager?

We don't take into account the scalability and the execution time. Our real case scenario don't imply too many services (few trucks, 1 manager, few clients), thus scalability wasn't our priority. The execution time is short enough (few minutes) to be negligible compared to a real truck driving and give exploitable results in our case-study.

4.4 Experimentation Framework

We have implemented the architecture and concept described in the previous sections. Figure 8 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.

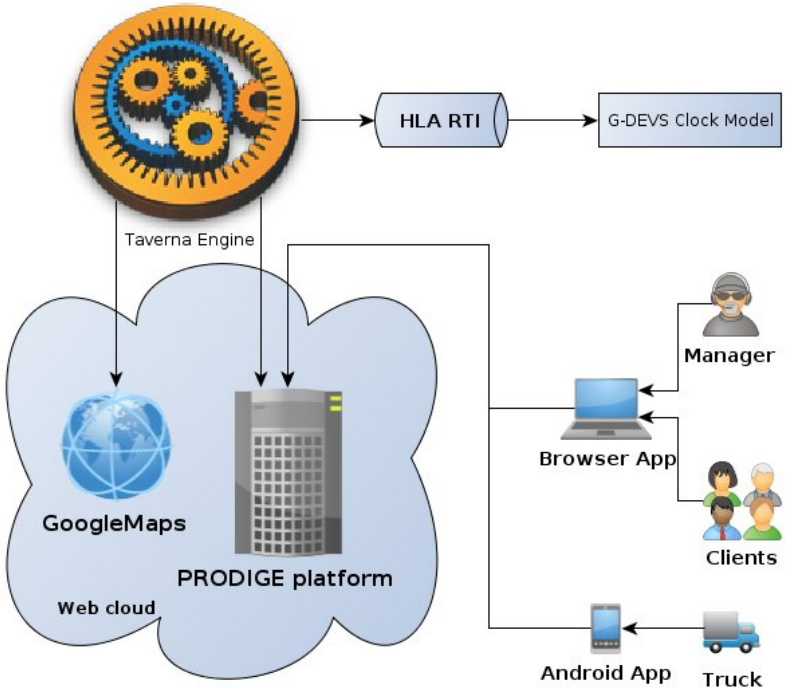

Figure 8. PRODIGE and simulation framework architecture.

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 9.

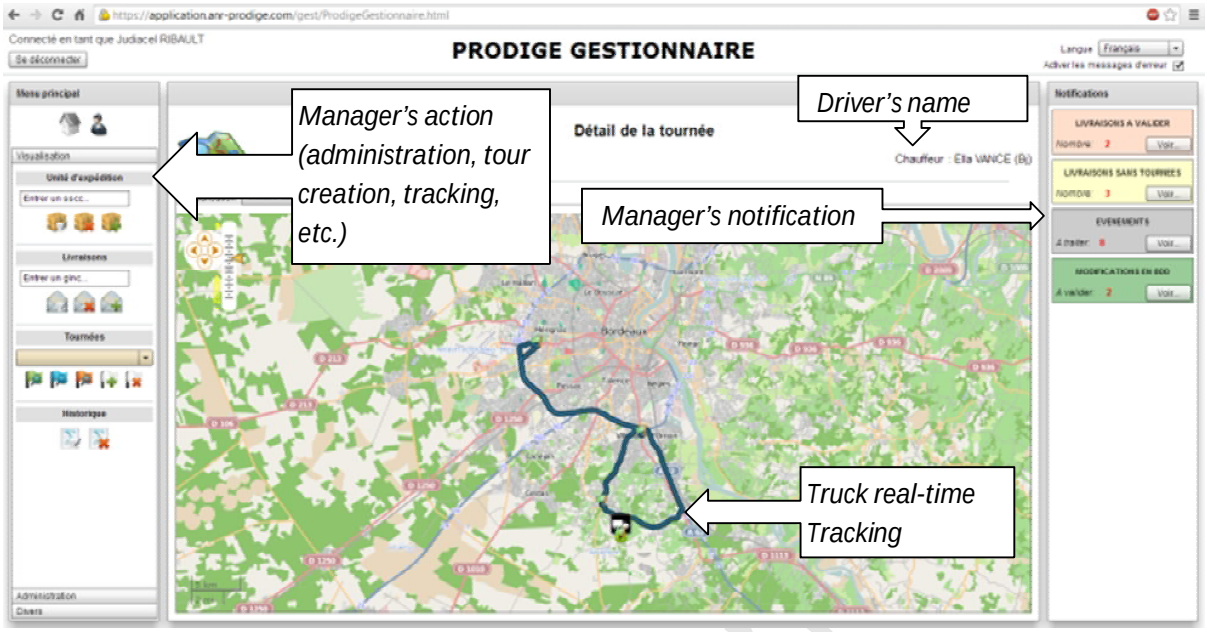

Figure 9. PRODIGE Web application

4.5 Experimentation Results

We made several run of the experimentation with different scenarios. Our first runs were helpful to validate the PRODIGE platform, identify bugs and help correcting them. Thus, it clearly reduces the time to market and cost compare to a solution involving real trucks.

Then, we use the experimentation framework to test and optimize the routing algorithm which was part of a PhD thesis associates to the PRODIGE project. Again, it clearly reduces the time to market to virtually build specific scenario to test specific routing problem. All of these experimentations were built above the same Taverna Workflow and G-DEVS orchestration, but with different inputs: client's locations, number of delivery, number of truck available.

Finally, we use the workflow as a demonstration tool. It allow to show the execution process step-by-step and graphically. We could also add some user interface into the workflow to interact with users. Thus, non-expert can use the workflow for answering simple questions, such as: How many trucks are available today? How many drivers? Is there a problem during the delivery? In addition, a small documentation describes the whole experimentation framework installation, setup and execution. For instance, Master students during practical classes at the university were able to quickly run the workflow and modify some branching themselves, which answers to our ease of use objective.

5 CONCLUSION

This work has permitted to introduce a new architecture for interoperability between simulations and WF of web services including time constraints. It started by recalls on existing works on distributed simulation standard HLA and G-DEVS formalism for the description of the logistic platform components. Then, it introduced the Taverna tool as the interoperability link to connect the web services on one side to the simulation components on the other side. Then it described the G-DEVS/HLA architecture that has been proposed to serve as the clock ordering component in the system since Taverna and more generally web services do not consider time synchronization as core concern.

The main objective of this paper was to demonstrate the interoperability between simulated components and enterprise services (including applications, web services and human resources). The approach was driven by the objective to replace rapidly real components by simulated ones due to their non-availability. For instance, during the specification step or when some restrictions occur on components, the simulated components permit to replace them and yet test the global workflow.

In order to combine simulated components and the rest of the WF components, an orchestration tool was required. In that objective, we have proposed on one side to use Taverna to deal with service interoperability and, on the other side, G-DEVS/HLA Clock architecture to handle time management. These two combined aspects represent the main contribution of the paper. This architecture has been validated on a logistics and transportation platform. In particular, this project was mixing terrain devices difficult to solicit and ongoing development platform. Simulation has permitted to reduce the time to market by replacing unavailable components during implementation.

However, this architecture is nonspecific to this domain and can be reused to handle any kind of services. Future works will consist in transposing the concept tested on Taverna to other Workflow management systems. This will result in several customized architecture dedicated to performance, usability, scalability depending on the workflow management system chosen and goals specified.

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), IEEE.

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

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

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

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.

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

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

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

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).

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

Katherine Wolstencroft, Robert Haines, Donal Fellows, Alan Williams, David Withers, Stuart Owen, Stian Soiland-Reyes, Ian Dunlop, Aleksandra Nenadic, Paul Fisher, Jiten Bhagat, Khalid Belhajjame, Finn Bacall, Alex Hardisty, Abraham Nieva de la Hidalga, Maria P. Balcazar Vargas, Shoaib Sufi, and Carole Goble (2013). *The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud.* Nucleic acids research, 41(W1), W557-W561.

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. *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 (2011), *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 (2011). *Worms- a framework to support workflows in m&s.* In Proceedings of the 2011 Winter Simulation Conference.

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.

poRTIco (2009), Developer Documentation. available from: http://porticoproject.org [accessed 2 June 2014]

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

Weske, M. (2012). *Business Process Management Architectures*. 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 (2006, January). *A taxonomy of workflow management systems for grid computing.* Journal of Grid Computing, 3(3-4):171–200

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.

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

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.

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.