Using Workflows And Web Services To Manage Simulation Studies

Judicaël Ribault, Gabriel Wainer
Department of Systems and Computer Engineering
Carleton University Centre for Visualization and Simulation (V-Sim)
Ottawa, ON K1S-5B6 Canada
jribault@gmail.com, Gabriel.Wainer@sce.carleton.ca

Keywords: workflow, replayability, web service

Abstract

In the last few years, computer simulation has suffered from a credibility crisis. One main issue is the lack of replayability for the simulations. We propose a new methodology for executing simulation studies in the cloud (accessible through RESTful Web Services) and orchestrated by workflows. We will show how replayability could be improved by sharing the same simulation environment, simulation files, simulation inputs and also the same experiment workflows. As a proof of concept, we have implemented our methodology to manage CD++ simulation studies using RISE, Taverna, and myExperiment.

1 INTRODUCTION

In the last few years, computer simulation has suffered from a credibility crisis. Credibility involves a strong validation, verification and accreditation (VV&A) process of the simulation model. But credibility also involves *replayability* of the simulation. Several studies [Pawlikowski, 2003; Pawlikowski *et al.*, 2002] questioned credibility in modeling and simulation. The authors surveyed over 2200 publications on telecommunication networks, and their conclusion was that "[...] the majority of recently published results of simulation studies do not satisfy the basic criteria of credibility", including replayability. In [Kurkowski, 2006], the authors analyzed several papers from the ACM MobiHoc symposium between 2000 and 2005, and they have found that less than 15% of the simulation results were replayable.

There are several reasons why replayability is an issue. One of them is that installing a simulation environment can be complex, as it implies dealing with dependencies (the correct libraries and compiler) and configuring the simulation environment precisely and this is even more complex for distributed simulations, as it may also imply to deal with hardware or software optimizations used in the original study. Then one must reproduce the simulation in the same way as the original, which implies that one must have the same simulation files, inputs and also the same experimentation process.

In recent years, there has been some efforts [Al-Zoubi and Wainer, 2010b; Rybacki *et al.*, 2010, 2011], that introduced the idea of using workflows to deal with some of these issues. Workflow was first designed to improve the business process. A product workflow is a set of steps required for developing a product until it gets into market [Weske, 2007]. 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.

Here, we show how replayability could be improved by sharing the same simulation environment, simulation inputs and experiment workflows, and how these elements can be shared using Web Services, in order to make them easily accessible from any device connected to the Internet. This new methodology allows executing simulation studies in the cloud (accessible through RESTful Web Services [Richardson and Ruby, 2007] and orchestrated by workflows). The idea is based on:

- to make existing simulators accessible through RESTful Web Services
- to build experiment workflows based on a process that deals with the RESTful simulation environment,
- sharing the workflow with other practitioners.

To replay an experiment, we only need to get the workflow, and run it. The workflow automatically executes all the experiment steps once again, using the same simulation environment.

As a proof of concept, we have implemented our methodology to manage CD++ simulation studies [Wainer, 2002] using RISE [Al-Zoubi and Wainer, 2010a], a middleware that provide RESTful Web Services [Richardson and Ruby, 2007], Taverna [Hull *et al.*, 2006] a workflow management system, and myExperiment [Goble and De Roure, 2007] a Web site to share and reuse workflows.

Section 2 provides necessary background and state of the art in this field. Section 3 describes the architecture of the solution, while Section 4 presents the use of Taverna workflows and RISE to manage CD++ simulations. Finally, section 5 discusses the potential opportunities offered by this methodology.

2 BACKGROUND

In this Section, we present a brief state of the art of workflows management system that can use and integrate Web Services, and the RESTful Interoperability Simulation Environment (RISE) middleware we used for this research.

2.1 Workflows of services

We need to orchestrate various services to manage and formalize the simulation process in the Cloud. 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 modeling. Due to our needs, we identify that a workflow management system such Taverna would be a better alternative than BPEL to illustrate the feasability of our approach.

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 to distributed resources. In [Curcin and Ghanem, 2008], four of the most popular scientific systems were reviewed. We decided to use Taverna [Hull *et al.*, 2006] instead of Kepler [Altintas *et al.*, 2004; Ludäscher *et al.*, 2006], Triana [Churches *et al.*, 2006] or Worms [Rybacki *et al.*, 2011] because Taverna eases and focus on the integration of RESTful Web Service and provides a deep integration of myExperiment services.

Taverna [Hull *et al.*, 2006] is an application that eases the use and integration of the growing number of tools and databases available on the Web, especially Web Services. It allows users who are not necessarily expert programmers to design, execute and share workflows of Web Services. These high-level workflows can integrate many different resources into a single analysis. A Taverna workflow can contain several services that can be useful to manage simulation studies:

- A service to develop a service in Java directly inside a Taverna workflow.
- 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 query a RESTful Web Service.
- A service to access local tools within a workflow. Thus, we can use local tools within a Taverna workflow.

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. The Taverna RESTful service can take in input various data, and it returns a status code and a response. For example, figure 1 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 (for example 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.

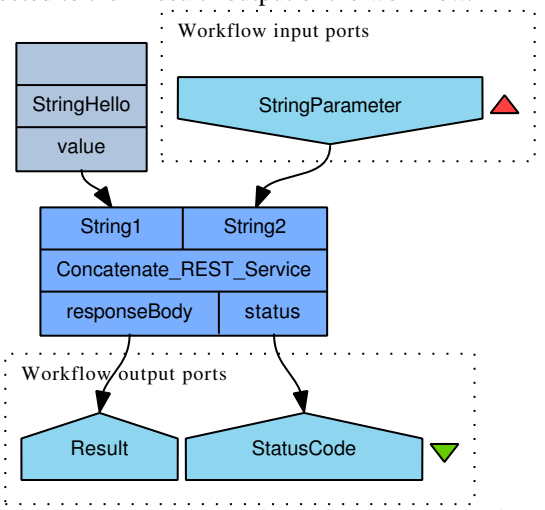

Figure 1: Taverna workflow with a REST Service.

Workflows are ideal for automating simulation tasks, but all the parameters needed as inputs might not always easily identifiable in advance. In those cases, it is desirable to interact with the users to understand what they want to do. 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, several workflows can be combined together to obtain more complex workflows. A complex workflow can then be saved and shared on myExperiment.

myExperiment [Goble and De Roure, 2007] is an online social networking environment to find, share and reuse workflows. myExperiment allows anybody who finds workflows relevant to their research to reuse and repurpose them to their specific requirement. At the same time, developers who build their own workflows can submit them to myExperiment to contribute to the scientific community. myExperiment also provide a social aspect to the sharing of workflows, allowing developers to get feedback from the community of users who reuse their work. Since its release in 2007, myexperiment has over 5000 registered users and contains more than 2000 workflows.

2.2 Middleware for simulation services

There are now a large number of research and commercial applications that target specific modeling and simulation environments [Zacharewicz *et al.*, 2008]. In our case we use RISE. In [Al-Zoubi and Wainer, 2010a] the authors discussed

the advantages and disadvantages of many of them, including the High Level Architecture (HLA) [Kuhl *et al.*, 1999], CORBA, SOAP-based Web-services, etc. As discussed there, most of these distributed simulation middlewares still lack of plug-and-play interoperability, dynamicity, and composition scalability. Instead, RESTful Web-services [Richardson and Ruby, 2007] imitates the Web interoperability style. Based on these ideas, we designed the first existing RESTful Interoperability Simulation Environment (RISE) middleware.

The main goal of RISE it to provide simulation interoperability and mashup regardless of their formalism, theory or implementation. Access to RISE is done through Web resources (URIs) and XML messages using HTTP channels: GET (to read a resource), PUT (to create new resource or update existing data), POST (to append new data to a resource), and DELETE (to remove a resource). RISE allows modelers to run any number of experiment instances, whose settings and resources (URIs) are persistent and repeatable (unless deliberately removed or updated). An interface between RISE and CD++ [Wainer, 2002] allows running distributed simulations.

RISE is the first (and only) REST-based simulation middleware available to date. Most distributed simulation systems have been provided using SOAP-based WS and other approaches (see [Wainer *et al.*, 2008] for a more comprehensive list). The RISE API looks like a classic website URL such as: "http://www.site.com/cdpp/sim". The following services are provided:

1. ../cdpp/sim/workspaces/{userworkspace} holds all simulation services for a given user.
2. ../cdpp/sim/workspaces/{userworkspace}/{servicetype} holds all frameworks for a given user and service type (i.e., a simulation engine like CD++).
3. ..orkpaces/{userworkspace}/{servicetype}/{framework} allows interacting with a framework (including the model's source code, the simulation's input variables, and sub-models interconnections).
4. ..erworkspace}/{servicetype}/{framework}/simulation allows interacting with the simulator.
5. ..erworkspace}/{servicetype}/{framework}/results holds the simulation outputs.
6. ..erworkspace}/{servicetype}/{framework}/debug holds the model-debugging files.

All these URLs respond to HTTP methods, for example, to submit files to a framework we use the POST method in the URI defined by Line 3. PUT is used to create a framework or update simulation configuration settings. DELETE is used to remove a framework. GET is used to retrieve the framework's state.

The main motivation behind basing workflow component on RESTful services is that implementation can be hidden in resources, which are represented via URI and are accessible using HTTP channels.

The next section present a method to address the problem of the *replayability* of simulation studies by using RESTful Web Services and orchestrated by workflows.

3 WORKFLOWS WITH RISE

A discrete-event simulation usually requires a simulation environment, simulation files (models, data initialization, ...), input parameters, and an experimentation process. The replayability of a discrete-event simulation study involves the reuse of all these elements. The reuse of simulation files can be easily solved by using a repository; it is much more complicated to reuse the simulation environment and the experimentation process.

Reusing the simulation environment involves the local installation of the simulation environment used in the original simulation study, which can be complex because we must take into account all the dependencies, the configuration, and the optimization of the simulation environment with precision. The reuse of the experimentation process involves the accurate reproduction of all the steps that led to the execution of the simulation. This task can also be complex since it requires a formal description of all the steps of the experimentation.

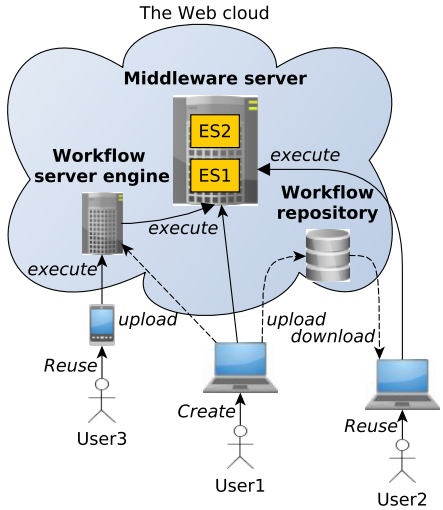

Figure 2: Overview of the proposed contribution.

Figure 2 shows a conceptual view of the proposed architecture. ES1 and ES2 represent two different simulation environments in the Web cloud. RISE can be used to make those (and other) simulators available through RESTful Web Services. RISE can provide RESTful services to several simulation environments, and additional services to the simulation environments (such as archiving the simulation files and results to enhance the replayability).

Figure 2 also shows that accessibility to the simulation environment can be done through several platforms: either directly through the local execution of the workflow (us-

ing a personal computer) or through a "workflow server engine". The execution of the simulation in the cloud allows not installing the simulation environment locally, sharing and reusing the same simulation environment among all practitioners and improving repeatability. The use of the RESTful protocol to reach and manage the simulation environments improves, through a simple protocol (GET, PUT, POST, and DELETE), the accessibility, the interoperability, and the mashup of applications.

Figure 2 also shows three users. User1 created a workflow and shared it on a "workflow repository". User1 also made the workflow executable from the cloud by uploading it on a "workflow server engine". User1 can also execute the workflow locally: the workflow engine invokes the Web Services provided by the "middleware server" to control the simulation environment "ES1" and "ES2". User2 downloaded the workflow uploaded by User1 and reused it. User3 used lightweight client to log onto the "workflow server engine", and execute the workflow previously uploaded by user1.

Workflows can help to automate these steps. The automation will allow repeating the simulation process without requiring human interaction. This can be useful to run several times the same simulation by changing a few settings automatically, such as the random seed or the input parameters. In addition, a workflow is more convenient for designing and sharing than a script. Indeed, a workflow using the RESTful protocol to control the remote simulation does not require any installation or configuration of the local simulation environment. The execution of the workflow allows repeating the simulation process completely, from any place and software environment.

Workflows can also guide the user during the simulation process. While most of the simulation environments provide a graphical interface to a dedicated simulation domain (biology, networking, etc.), a workflow is dedicated to a specific simulation study and allows to take into account its specificity.

In [Al-Zoubi and Wainer, 2010b], a solution based on the BPM formalism was proposed. This formalism is used in business process but e-science has other expectations needs, such as the necessity to track the provenance of the workflow execution results and to share feedbacks and comments. Likewise, in [Rybacki *et al.*, 2010], the authors propose a workflow solution to improve the replayability of their simulation environment. It first implies to install the simulation environment while we try to avoid it. In [Rybacki *et al.*, 2011], the authors propose an interesting workflow system dedicated to M&S. Their solution does not take into account the execution of workflow via Web Service, without having to install any simulation environment. Our solution facilitates the comparison of simulation studies because practitioners share the same simulation environment. Our solution also eases the sharing of workflows among practitioners, because RISE is environment-independent.

4 USING TAVERNA WORKFLOWS

We built a proof-of-concept implementation of our methodology using RISE, the CD++ simulation environment, Taverna, and myExperiment.

RISE makes available the CD++ simulation environment using RESTful Web Services as explained in [Wainer *et al.*, 2008]. We reused the exact same RISE server and CD++ simulation environment, without any need of modification.

The life cycle of a simulation with CD++ involves the installation of the simulation environment, the creation of the model, the compilation of the model, the execution of the simulation, and the retrieval of simulation results to be further analyzed. The use of RISE allows one to avoid the installation of the simulation environment, using an installation behind the RISE middleware, accessible through the Web.

To manage a CD++ simulation with RISE using Taverna workflow, we need to create or import existing Taverna workflows from myExperiment into a new Taverna workflow. This new Taverna workflow will contains the following workflows (available on myExperiment) as presented on figure 3.

- select a workspace among those present on the RISE server (RISE_ChooseUserWorkspace).
- select a simulation environment among those present in the user workspace (RISE_ChooseServiceType).
- select a framework among those present in the user workspace and corresponding to the selected simulation type (RISE_ChooseFramework).
- enter a name for the new simulation experiment (RISE_AskFrameworkName).
- select the XML configuration file on the user personal computer (RISE_SelectXmlModelFile).
- select the zip file containing all the files of the simulation: model, inputs, etc. (RISE_SelectZipModelFile).
- create a framework on RISE (RISE_CreateFramework).
- send the zip file on RISE (RISE_SendZipModelFile).
- execute the simulation on RISE (RISE_ExecuteSimulation).
- get the simulation results (RISE_GetResults).

This parent workflow (figure 3) requires no input to be executed. It performs, via an interaction with the user, the required steps to run a new CD++ simulation on RISE.

A Taverna workflow is data-flow oriented, which means that as soon as all the inputs are provided, a workflow component is executed. In our workflow (figure 3), we can see that the last nested-workflow *RISE_GetResults* receive in inputs the outputs of the workflows *RISE_ChooseUserWorkspace*, *RISE_ChooseServiceType*, and *RISE_AskFrameworkName*. As soon as we answered those 3 questions, the workflow RISE_GetResults will be executed. Thus, it is necessary to

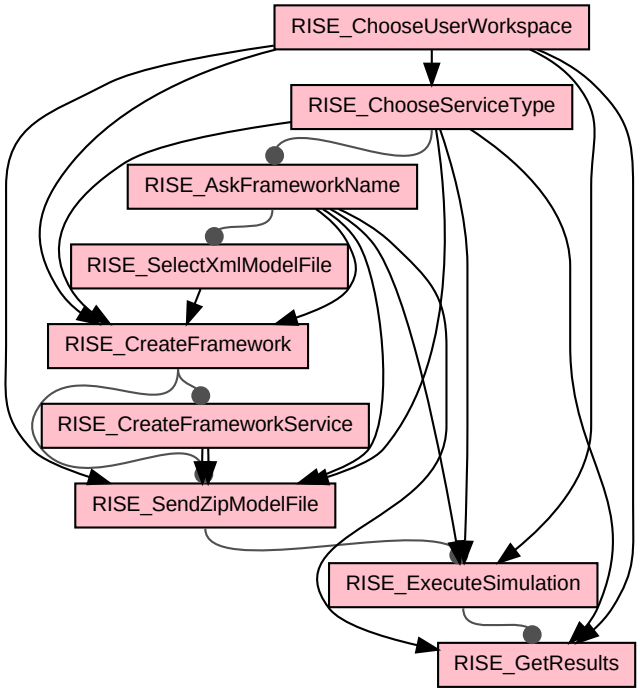

Figure 3: Complete workflow to manage new CD++ simulation using RISE WS.

schedule workflows, otherwise they will run simultaneously. The collection of the results (workflow RISE_GetResults) can only be done after the execution of the simulation (workflow RISE_ExecuteSimulation). As well the execution of the simulation can only be done after uploading th simulation files onto the RISE middleware (workflow RISE_SendZipModelFile). Finally, the upload can only be done after the complete creation of the simulation experiment into the user workspace (workflow RISE_CreateFramework). This dependance is represented by a line and a black circle on figure 3.

This workflow can be found on myExperiment and it is possible for the owner of the workflow to upload a new version of the workflow, as well as to update the information about the workflow. It is also possible for users to import the workflow, in order to reuse it, as it, or as a base for a new workflow, or as a nested workflow part of a bigger workflow. It is also possible to tag, rate and comment the workflow.

To replay a study, users only have to get the corresponding workflow and run it using Taverna. There is also the possibility for the developer of the workflow to setup a Taverna Server and upload the workflow on it. The execution of the experiment workflow of the figure 3 is done as follow:

- The "RISE_ChooseUserWorkspace" nested workflow invokes the RISE RESTful service to get the list of workspaces, and asks the user to select one workspace.
- The "RISE_ChooseServiceType" nested workflow invokes the RISE RESTful service to get the list of simulation services available for the selected workspace and asks the user for one simulation service type.
- The "RISE_AskFrameworkName" nested workflow asks to enter a name for the simulation experiment.
- The "RISE_SelectXmlModelFile" nested workflow asks for a XML configuration file on the local computer.
- The "RISE_CreateFramework" nested workflow invokes the RISE RESTful service to create a new experiment on the server. Taverna receives an HTTP Basic Challenge and asks the user to authenticate.
- The "RISE_SelectZipModelFile" nested workflow asks for the model files.
- The "RISE_SendZipModelFile" nested workflow invokes the RISE RESTful service to upload and compile the model into the experiment.
- The "RISE_ExecuteSimulation" nested workflow invokes the RISE RESTful service to execute the simulation and invokes periodically the RISE RESTful service to know the simulation status in order to be aware of the simulation termination. Once the simulation is terminated, the "RISE_ExecuteSimulation" stop and the experiment workflow continue.
- The "RISE_GetResults" nested workflow invokes the RISE RESTful service to download the simulation results.

5 CONCLUSION

We have defined and built a workflow system using Web Services to perform remote simulations. We have illustrated the benefits of using the RISE middleware to provide RESTful service to CD++. We showed how to create a simple but complete Taverna workflow to illustrate the concept.

However, workflows can be much more complex and must be able to handle all steps of a simulation study. In a case of emergency planning, an emergency crew would benefits from having an automated workflow that run new simulations remotely, based on real-time data and processing the simulation results to facilitate their reading directly from the emergency site. Moreover, a workflow ensuring the replayability of a simulation should not interact with the user; it should run automatically. Conversely, a workflow to teach or to run simulations quickly must assume that the user has no knowledge, and should help them during each step; from the creation of the simulation to the analysis of the results. The workflow must handle simulation errors and report them to the user. This can be done by displaying error messages or by sending e-mails to the user. The resolution of errors can also be automated.

This way of managing simulation studies should allow to:

- facilitate the learning of the simulation software by offering learning workflows to guide the user.
- facilitate access to the simulation by reducing the infrastructure costs to a minimum: it only requires access to

the Internet to execute a workflow containing only remote service, such as RISE.

- increase the credibility of a study, providing automated workflows that will guarantee the replayability.
- mix data from simulations and data from the real world to make mashups, and use simulation as a prediction and decision-making tool.

ACKNOWLEDGEMENT

This research was supported by a fellowship from the Direction Générale de l'Armement and the INRIA Sophia-Antipolis and partially funded by NSERC.

REFERENCES

K. Al-Zoubi and G. Wainer. Distributed simulation using restful interoperability simulation environment (rise) middleware. *Intelligence-Based Systems Engineering*, pages 129–157, 2010.

K. Al-Zoubi and G. Wainer. Managing simulation workflow patterns using dynamic service-oriented compositions. In *Winter Simulation Conference (WSC), Proceedings of the 2010*, pages 765–777. IEEE, 2010.

I Altintas, 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.

David Churches, 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.

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

Ewa Deelman, 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.

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.

Duncan Hull, Katy Wolstencroft, Robert Stevens, Carole Goble, Mathew R Pocock, Peter Li, and Tom Oinn. Taverna: a tool for building and running workflows of services. *Nucleic acids research*, 34(Web Server issue):W729–32, July 2006.

F. Kuhl, R. Weatherly, and J. Dahmann. *Creating computer simulation systems: an introduction to the high level architecture*. Prentice Hall PTR, 1999.

S.H. Kurkowski. Credible mobile and ad hoc network simulation-based studies, 2006.

B. Ludäscher, I Altintas, Chad Berkley, D. Higgins, E Jaeger, M. Jones, E.A. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.

K. Pawlikowski, H.D.J. Jeong, and J.S.R. Lee. On credibility of simulation studies of telecommunication networks. *Communications Magazine, IEEE*, 40(1):132–139, 2002.

K. Pawlikowski. Do not trust all simulation studies of telecommunication networks. In *Information Networking*, pages 899–908. Springer, 2003.

Leonard Richardson and Sam Ruby. *Restful web services*. O'Reilly, first edition, 2007.

Stefan Rybacki, Jan Himmelspach, Enrico Seib, and Adelinde M. Uhrmacher. Using workflows in ms software. In *Proceedings of the 2010 Winter simulation Conference*, pages 535–545, 2010.

Stefan Rybacki, 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.

Wei Tan, Paolo Missier, Ravi Madduri, and Ian Foster. Building scientific workflow with taverna and bpel: A comparative study in cagrid. In *Service-Oriented Computing–ICSOC 2008 Workshops*, pages 118–129. Springer, 2009.

G.A. Wainer, R. Madhoun, and K. Al-Zoubi. Distributed simulation of devs and cell-devs models in cd++ using web-services. *Simulation Modelling Practice and Theory*, 16(9):1266–1292, 2008.

G. Wainer. Cd++: a toolkit to define discrete-event models. *Software, Practice and Experience*, 32(3):1261–1306, 2002.

M. Weske. *Business process management: concepts, languages, architectures*. Springer-Verlag New York Inc, 2007.

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.

G. Zacharewicz, C. Frydman, and N. Giambiasi. G-devs/hla environment for distributed simulations of workflows. *Simulation*, 84(5):197–213, 2008.