Simulation Modelling Practice and Theory 55 (2015) 27–45

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Distributed simulation of DEVS and Cell-DEVS models using the RISE middleware

Khaldoon Al-Zoubi *, Gabriel Wainer

Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada

ARTICLE INFO

Article history:
Received 11 November 2014
Received in revised form 28 February 2015
Accepted 30 March 2015
Available online 17 April 2015

Keywords:
Interoperability
Distributed simulation
Web services
REST
SOAP
Middleware
DEVS
Cell-DEVS
CD++

ABSTRACT

With the expansion of the Web, the desire toward global cooperation in the distributed simulation technology has also been on the rise. However, since current distributed simulation interoperability methods are coupled with system implementations, they place constraints on enhancing interoperability and synchronization algorithms. To enhance simulation interoperability on the Web, we implemented the RISE (RESTful Interoperability Simulation Environment) middleware, the first existing simulation middleware to be based on RESTful Web-services (WS). RISE is a general middleware that serves as a container to hold different simulation environments without being specific to a certain environment. RISE can hold heterogeneous simulations, and it exposes them as services via the Web. One of such services is called Distributed CD++ (DCD++) simulation system, an extension of the CD++ core engine that allows executing DEVS and Cell-DEVS models. Here, we introduce a proof-of-concept design and implementation of DCD++ using the distributed simulation using the RISE environment. We show how the RESTful WS interoperability style in RISE has improved the design, implementation and the performance of the DCD++ simulator. We also discuss a substantial performance improvement of the implementation of the RISE-based DCD++ presented here, showing many advantages of the RESTful WS presented here: improved interoperability, a seamless method to be connected into a cloud computing environment, and performance improvement when compared to our SOAP-based DCD++ in a similar testing environment.

1. Introduction

Modeling and simulation (M&S) is used extensively in studying complex systems. As simulated systems become increasingly sophisticated, the simulation software becomes larger and more complex. In these cases, the resources provided by a single-processor machine often become insufficient to execute these systems. Distributed simulation can expand from a single building to global networks usually interoperating heterogeneous processors (and software) [15]. With the expansion of the Internet, the desire toward global cooperation in the distributed simulation technology has also been on the rise as indicated by a number of surveys such as [9,29]. A focal point of distributed simulation software has been on how to achieve model reuse via interoperation of different simulation components. Other benefits include [32] connecting geographically distributed simulation components (without relocating people/equipment to other locations), interoperating different vendor simulation components (allowing reuse of M&S solutions), and information hiding—including the protection of

* Corresponding author.
E-mail addresses: kazoubi@connect.carleton.ca (K. Al-Zoubi), gwainer@sce.carleton.ca (G. Wainer).

http://dx.doi.org/10.1016/j.simpat.2015.03.010
1569-190X/

intellectual property rights, and simulating larger problems via exploiting more available distributed computer resources (e.g. memory).

Web-enabled simulation is increasingly becoming the norm. These simulation systems can be divided into the following categories: (1) Web *access* enabled simulations, and (2) Web-based *distributed* simulations. The *Web access enabled simulation* systems provide users access to simulation systems running on a single machine [18,27] or on multiple machines geographically distributed [38]. However, the distributed simulation, in this case, is not *synchronized* via the Web, but according to those specific architectures. The second category is the *Web-based distributed simulation*. In such systems, the simulation partitions are synchronized partially or fully using Web services. Web-based distributed simulation interoperability methods are usually based *on SOAP-based WS* (e.g. [22,28,30,37]) or an *HLA* [20] *with SOAP WS* interface (e.g. [10,19,39,40]). However, such distributed simulation frameworks still have constraints in the structural rules that are placed on the interoperability design methods. In particular, the way they exchange, structure, and use information is tied to programming, making it difficult to decouple systems implementations and design. In practice, such constraints are difficult to overcome when interoperating existing heterogeneous simulation systems to synchronize same distributed simulation run. This is because each system interface is highly coupled to its own software design and programming, hence it is complex to homogenize different systems interfaces in order to be able to synchronize a distributed simulation.

To provide better solutions to overcome such interoperability constraints, we have proposed the RESTful Interoperability Simulation Environment (RISE) middleware [2], which is the first existing RESTful WS simulation environment. The objective was to achieve improved interoperability, as explained in [2], using RESTful WS. The use of RESTful WS is on the rise, and it has become the standard interoperability method on industrial cloud computing environments such as those in IBM [17] and Cisco [12]. RISE middleware has been built as a general container of different simulation environments. RISE exposes simulations as services (as URIs), allowing them to be seamlessly interoperated with other services (simulation and other) on the Web and cloud computing environments. The Distributed CD++ simulator (DCD++) presented here is one of such services exposed by RISE. Therefore, the presented DCD++ here is one of the service types exposed via the RISE middleware. However, RISE can expose more software services in addition to DCD++ like other simulators, visualization etc. [21,35].

In fact, exposing simulation services via the RISE middleware interoperability methods have been proven on different fronts. For example, as described in [35], RISE exposed DCD++ simulation services via the Amazon elastic compute cloud (Amazon EC2) [5]. In [35] we also proposed methods for model composition using workflows via RISE. In [21] we proposed a hybrid simulation and visualization approach where a dedicated mobile application runs on an Android Smartphone and the RISE-based DCD++ simulation (whose details are presented here) runs the simulation while hosted on the Cloud.

As discussed above, RISE is a general a Web-based interoperability container (i.e. system-of-systems); hence, systems need to be plugged into RISE before being able to use its services. We extended simulator called CD++ [31] and plugged into RISE so that it can perform distributed simulation on the Web, called Distributed CD++ (DCD++) [3,4]. It is worth to note that RISE-based DCD++ is the only distributed simulation system to use RESTful WS interoperability style, hence being part of a Cloud. Further, performing RISE-based DCD++ distributed simulation comes as the first natural step toward practical DEVS standardization [34], allowing different DEVS implementation to interoperate in order to reuse each other models and resources.

In the following sections, we introduce the RISE-based DCD++ simulation design, algorithms, implementation and performance. In Section 2, we present background information and summarizes current related work with a comparison to the presented RISE-based DCD++ simulation. Section 3 presents the RISE-based DCD++ simulation design and synchronization algorithms. Section 4 presents the RISE-based DCD++ implementation and the middleware implementation with a focus on the relevant parts to the DCD++. Section 5 compares the performance of the RISE-based DCD++ (presented here) to the SOAP-based DCD++ [30].

2. Background

This section provides an overview of the RISE middleware and the DEVS formalism, and it discusses related work.

2.1. Overview of the RISE middleware interoperability

RISE (RESTful Interoperability Simulation Environment) [2] is the first existing simulation middleware to be based on the RESTful Web-services [25]. In the Representational State Transfer (REST) [14] which is used to describe the WWW architectural elements, resources hold representations (states) where these representations are transferable between resources. For example, the client (e.g. Web browser) uses the GET HTTP method to transfer the HTML representation from the resource (e.g. Web site) to the client. Therefore, RISE is an interoperability container that can hold different simulation services regardless of their differences as shown Fig. 1. The presented DCD++ here is one of those services. However, RISE can expose more software services in addition to DCD++ like other simulators, visualization etc. (Fig. 1). It is worth to note that Fig. 1 only shows one partition of the DCD++ simulation, but DCD++ is often performed between several partitions.

To summarize RISE principles [2], RISE spreads services over a number of resources (resource-oriented), and resources exchange synchronization information in form of XML messages (message-oriented) via predefined constant virtual channels (uniform interface). These resources are exposed as URI templates whose instances can be created/destroyed at runtime