The Journal of Systems and Software 86 (2013) 2520–2541

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

S-IDE: A tool framework for optimizing deployment architecture of High Level Architecture based simulation systems

Turgay Çelik[a,*], Bedir Tekinerdogan[b]

[a] Department of Computer Engineering, Hacettepe University, Ankara, Turkey
[b] Department of Computer Engineering, Bilkent University, Ankara, Turkey

ARTICLE INFO

Article history:
Received 30 June 2012
Received in revised form 27 February 2013
Accepted 1 March 2013
Available online 20 March 2013

Keywords:
Deployment model optimization
Metamodel based tool development
Distributed simulation
High Level Architecture (HLA)
FEDEP
Software architecture
Model transformations
Metamodeling

ABSTRACT

One of the important problems in High Level Architecture (HLA) based distributed simulation systems is the allocation of the different simulation modules to the available physical resources. Usually, the deployment of the simulation modules to the physical resources can be done in many different ways, and each deployment alternative will have a different impact on the performance. Although different algorithmic solutions have been provided to optimize the allocation with respect to the performance, the problem has not been explicitly tackled from an architecture design perspective. Moreover, for optimizing the deployment of the simulation system, tool support is largely missing. In this paper we propose a method for automatically deriving deployment alternatives for HLA based distributed simulation systems. The method extends the IEEE Recommended Practice for High Level Architecture Federation Development and Execution Process by providing an approach for optimizing the allocation at the design level. The method is realized by the tool framework, *S-IDE* (Simulation-IDE) that we have developed to provide an integrated development environment for deriving a feasible deployment alternative based on the simulation system and the available physical resources at the design phase. The method and the tool support have been validated using a case study for the development of a traffic simulation system.

1. Introduction

Simulation systems are used to simulate real world concepts in different domains such as manufacturing, performance analysis, decision support, virtual exercises and entertainment. There are different reasons for using simulation systems including analysis and testing, cost reduction in development, training, etc. Due to the complexity of the simulated domain very often the simulation is executed across multiple nodes and likewise several different simulators are integrated within a single distributed simulation environment. The reason for distributing the simulation is usually for reducing the execution time of the simulation, enabling geographic distribution of simulation parts, and enabling large simulations with a large number of users (Fujimoto, 1999).

Developing distributed simulation systems is not easy because different simulators might run on different platforms, adopt different data types, use different communication mechanisms, etc. Hence, an important challenge in distributed simulation systems is the integration, reusability and interoperability of the various simulators. To reduce the effort for developing distributed simulations, several standard simulation infrastructures have been introduced including Distributed Interactive Simulation (DIS) (IEEE, 1998), High Level Architecture (HLA) (Kuhl et al., 1999; IEEE, 2010a), and Test and Training Enabling Architecture (TENA) (Noseworthy, 2008). Among these, HLA is an important IEEE and NATO standard specifies a general purpose simulation architecture for realizing interoperable and reusable distributed computer simulation systems (Kuhl et al., 1999; IEEE, 2010a).

One of the important problems in HLA based distributed simulation systems is the allocation of the different simulation modules to the available physical resources. Each deployment alternative represents a different allocation of modules to physical resources and this can be done in many different ways. Further, each deployment alternative will have a different impact on the performance. This problem can be categorized as a *task allocation problem* that has been widely addressed in the literature (Stone, 1977; Lo, 1988; Pirim, 2006; Mehrabi et al., 2009). To solve the task allocation problem different algorithmic solutions have been proposed. Hereby, the algorithms take as input several optimization parameters such as execution cost, communication cost, memory requirement and I/O cost. Based on these input parameters the task allocation algorithms aim to derive feasible allocation of tasks to processors (Stone, 1977; Lo, 1988). The evaluation of the deployment alternative is usually based on expert judgment and postponed to the implementation phase. One cannot always rely on expert judgment because finding experts that have both a broad and specialized

* Corresponding author. Tel.: +90 505 476 8307.
E-mail address: turgaycelik@gmail.com (T. Çelik).

0164-1212/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2013.03.013

knowledge on the corresponding domains is not easy. Further, human expert judgments can be feasible for small to medium systems but are inadequate for large and complex systems. Moreover, postponing the evaluation of the deployment alternative to the implementation phase, might lead to non-feasible implementations which may require unnecessary iteration of the design and the related project lifecycle artifacts such as detailed design, implementation, test artifacts, documentation, etc. On its turn this will lead to delays in the project schedule and increased cost due to the unnecessary rework of the lifecycle artifacts.

The need for early analysis and optimization of the deployment alternatives has also been addressed by the IEEE Recommended Practice for High Level Architecture Federation Development and Execution Process FEDEP (IEEE, 2003). FEDEP describes recommended tasks for evaluating alternative design options and estimating the simulation performance in design phase but deliberately does not provide a detailed process and implementation for the indicated tasks.

To cope with the above problems and address the needs as addressed by FEDEP, we propose a method and our tool framework *S-IDE* (*Simulation-IDE*) that supports the early analysis of deployment alternatives and the automatic generation of the deployment alternatives for HLA based distributed simulation systems. *S-IDE* tool framework consists of several tools based on metamodels that we have developed including Federation Data Exchange Metamodel, Simulation Modules and Publish–Subscribe Relations Metamodel, Physical Resources Metamodel, Simulation Execution Configuration Metamodel, and Deployment Metamodel. Based on the design models developed with these tools, the necessary parameter values for the task allocation algorithms are defined, which are then used for automatic generation of a feasible deployment alternative. In addition, the tool framework can be used for design level analysis including, the impact of adding new simulations modules to the system, suitability of the selected physical resources for the given simulation design, and the impact of the change of publish–subscribe relations. To illustrate the usage of the method and S-IDE we have used a realistic case study concerning the development of a traffic simulation.

The remainder of the paper is organized as follows. In Section 2 we provide the background on HLA and Model Driven Engineering (MDE). Section 3 defines the problem statement based on a case study that will be used in subsequent sections. Section 4 presents the method for evaluating alternative design options briefly. Section 5 describes the metamodels that S-IDE tool framework is built on. Section 6 presents the model transformations step by step for deriving feasible deployment alternatives. Section 7 provides realization of S-IDE tool framework and using S-IDE to derive a feasible deployment alternative for the case study. Section 8 provides the evaluation of the tool. Section 9 provides the discussion. Section 10 describes the related work and finally we conclude the paper in Section 11.

2. Preliminaries

In this section we describe the background for understanding and supporting the approach that we present in this paper. In Section 2.1 we present a brief definition of the High Level Architecture (HLA), followed by a short overview of Model-Driven Engineering (MDE) in Section 2.2.

2.1. High Level Architecture (HLA)

As stated before, HLA is an IEEE standard that supports development of reusable and interoperable distributed simulation systems (Kuhl et al., 1999; IEEE, 2010a,b,c). To support the development of HLA compliant simulation systems, the "Federation Development and Execution Process – FEDEP" has been defined as a part of HLA standard (IEEE, 2003).

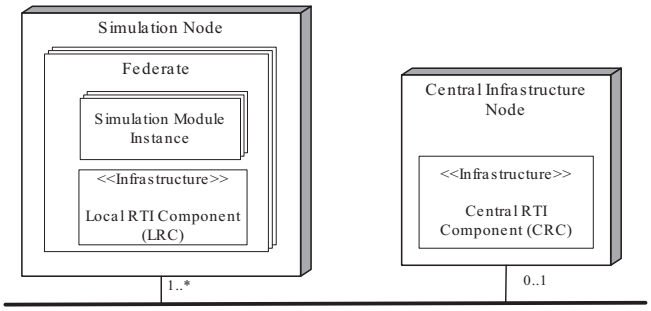

Fig. 1. Reference architecture for the high level architecture.

Based on a domain analysis to HLA standard we could derive the reference architecture of HLA based simulation systems which is shown in Fig. 1. A typical simulation system is deployed on a number of *Simulation Nodes*. Each *Simulation Node* includes one or more *Federates* which are processes that together form the simulation execution. Each member includes a number of *Simulation Module Instances* and *Local RTI Component* (*LRC*). *Simulation Module Instances* represent objects for simulating entities or events in the simulation. RTI represents the runtime infrastructure that realizes the HLA standard (IEEE, 2010a). *LRC* enables bi-directional interaction between federates for data exchange and collaborative execution of the simulation.

The simulation may also include an optional *Central Infrastructure Node* that contains *Central RTI Component* (CRC) which is responsible for managing the simulation lifecycle, timing, synchronization, and discovery concerns. Although this component is not mandatory, as a convention, major RTI implementations provide *CRC* definitions. In case *CRC* is missing, the services need to be supported by the *LRCs.* As such both the *LRC* and *CRC* provide similar services. In Fig. 1 this is indicated through the stereotype «Infrastructure».

The *CRC* and *LRC* implementations together provide services for federation management, declaration management, object management, ownership management, time management, and data distribution management (IEEE, 2010b).

The basic interaction model that is adopted in the HLA conforms to the Publish/Subscribe pattern (Eugster et al., 2003). In the Publish/Subscribe pattern the producer and consumer applications (members) are decoupled. This increases the reusability and interoperability, which are key concerns in simulation systems. The Publish/Subscribe interaction is realized by the «Infrastructure» components in the reference architecture in Fig. 1. Federates in the simulation execution can publish and subscribe data exchange model elements through the services provided by the «Infrastructure» components. HLA standard defines the *Object Model Template* (*OMT*) that can be used to define different data exchange models which are called *Federate Object Model* (*FOM*) and *Simulation Object Model* (*SOM*) (IEEE, 2010c).

2.2. Model Driven Engineering (MDE)

In traditional, non-model-driven software development the link between the code and higher level design models is not formal but intentional. Required changes are usually addressed manually using the given modeling language. Because of the manual adaptation the maintenance effort is not optimal and as such sooner or later the design models become inconsistent with the code since changes are, in practice, defined at the code level. One of the key motivations for introducing model-driven engineering (MDE) is the