Proceedings of the 2013 Winter Simulation Conference
R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, eds.

A SAAS-BASED AUTOMATED FRAMEWORK TO BUILD AND EXECUTE DISTRIBUTED SIMULATIONS FROM SYSML MODELS

Paolo Bocciarelli
Andrea D'Ambrogio
Andrea Giglio

Dept. of Enterprise Engineering
University of Rome Tor
Vergata Rome, ITALY

Daniele Gianni

Dept. of Science and Applied Technology
"Guglielmo Marconi" University
Rome, ITALY

ABSTRACT

The development of complex systems requires the use of quantitative analysis techniques to allow a design-time evaluation of the system behavior. In this context, distributed simulation (DS) techniques can be effectively introduced to assess whether or not the system satisfies the user requirements. Unfortunately, the development of a DS requires the availability of an IT infrastructure that could not comply with time-to-market requirements and budget constraints. In this respect, this work introduces *HLAcloud*, a model-driven and cloud-based framework to support both the implementation of a DS system from a SysML specification of the system under study and its execution over a public cloud infrastructure. The proposed approach, which exploits the HLA (High Level Architecture) DS standard, is founded on the use of model transformation techniques to generate both the Java/HLA source code of the DS system and the scripts required to deploy and execute the HLA federation onto the PlanetLab cloud-based infrastructure.

1 INTRODUCTION

Modeling and simulation represents a valuable time- and cost-effective alternative to carrying out experiments and measurements on existing systems and/or real-world phenomenons. In the systems engineering domain, the design and the implementation of complex, large-scale and heterogeneous systems claim for advanced quantitative analysis techniques to allow an early evaluation of the system behavior, in order to assess, before implementation, whether or not the system accomplishes the stakeholder requirements and constraints.

Due to the intrinsic distributed nature of such complex systems, which are often composed of several subsystems, distributed simulation (DS) may be effectively introduced to enact a design-time evaluation of the system behavior. Thus, the adoption of DS-based analysis techniques can be seen as a valuable strategy both to cut the cost of developing experimental prototypes and to mitigate the risk of time/cost overrun due to re-designing and re-engineering a system that does not provide the required quality.

In addition, the use of a DS-based approach may enact the reuse of already existing simulation components by taking advantage of the interoperability properties provided by the widely used HLA (High Level Architecture) DS standard, which makes possible federating newly developed components with reused ones.

Finally, the ever growing complexity of modern systems often requires computational capabilities that make impracticable the use of sequential simulation approaches.

Unfortunately, the use of DS in a system development lifecycle can be a challenging issue due to the fact that: i) the use of DS frameworks and standards requires significant and specific technical skills system engineers are usually not familiar with; ii) the development of a DS requires the availability of an appropriate IT infrastructure that could not comply with time-to-market requirements and budget constraints.

The first issue has been faced with in (Bocciarelli, D'Ambrogio, and Fabiani 2012), which illustrates a model-driven method to develop a DS of a system from its specification, by use of the HLA standard and the SysML specification language. This work is an extension of such a contribution, which has been revised and enhanced along two directions. First, the adopted workflow is now based on the IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP) (IEEE 2010d). Second, the existing model transformations have been modified and extended in order to ensure the compliance with the HLA-Evolved standard (IEEE 2010a) and to allow the deployment of the HLA federation onto a cloud-based infrastructure.

In this respect, in the last few years cloud computing is gaining attention as an effective approach to deploy and provide applications in a rapid, dynamic, scalable and virtualized way. The most relevant characteristic of cloud computing consists in its delivery paradigm, according to which the available computing resources are provided *on demand* in a scalable and dynamic way. More specifically, cloud computing is founded on an *everything-as-a-service* delivery model that enables the provision of applications (Software-as-a-Service, SaaS), development platforms (Platform-as-a-Service, PaaS) and computing infrastructures in terms of storing and processing capacity (Infrastructure-as-a-Service, IaaS) (Menascé and Ngo 2009).

In this context, cloud computing may ease the effective use of DS. Enterprises interested in introducing DS-based analysis approaches in their product lifecyle may exploit existing public cloud infrastructures for implementing such approaches, which relieve the need of purchasing, operating and maintaining a high-performance IT infrastructure at local site (Fujimoto, Malik, and Park 2010).

Indeed, this work illustrates *HLAcloud*, a model-driven framework that exploits the SaaS cloud paradigm to enact a Simulation-as-a-Service delivery paradigm (SIMaaS) (Tsai et al. 2011). The HLAcloud framework adopts the HLA standard for providing DS interoperability properties (IEEE 2010a, IEEE 2010b, IEEE 2010c).

HLAcloud makes use of a chain of model transformations both to develop the executable Java-HLA code of a distributed simulation from a SysML model and to deploy it in the cloud, without being required to own specific skills of DS standards and cloud computing environments.

The proposed methodology consists of a three-step transformation that takes as input the SysML model specifying the system architecture and yields as output the UML model of the corresponding simulation, which in turn is used to derive the HLA executable code and the script required to deploy the federation onto PlanetLab (PlanetLab 2013), a cloud infrastructure widely used as a research and development testbed.

Specifically, PlanetLab is an overlay network, built on top of the concepts of grid computing and distributed computing, which aims to support large scale research and development adopting the service-on-demand paradigm. Using PlanetLab as a testbed, experimental researchers are able to conduct computationally intensive experiments in a distributed and parallel way at a largely reduced cost. The aim of the testbed is to support researchers that want to develop new services, and clients that want to use them, allowing the overlay to become, in the long term, both a research testbed and a deployment platform.

The structure of PlanetLab can be described as a collection of machines distributed around the world. Most of the machines are hosted by research institutions, although some are located in co-location and routing centers. All of the machines are connected to the Internet. Each machine has at least one PlanetLab node which runs a common software package implementing PlanetLab services. A *slice* is a set of allocated resources distributed across PlanetLab nodes.

Finally, *HLAcloud* adopts two UML profiles, namely *SysML4HLA* and *HLAprofile*, which are used to annotate the several models specified along the system development lifecycle with the details required to drive the model transformations at the basis of the proposed approach.

The rest of this work is organized as follows: Section 2 reviews relevant contributions dealing with the topics addressed in this work. Section 3 illustrates the proposed *HLAcloud* framework, while Section 4 and Section 5 describe the UML profiles and the model transformations that are part of it, respectively. Finally, Section 6 gives an example *HLAcloud* application.

2 RELATED WORK

To the best of our knowledge, no contribution can be found that specifically deals with both the provisioning of simulation services by use of a cloud computing infrastructure and the model-driven automated implementation and deployment of DS code from system specifications, even though model-based approaches to the generation of simulation models from SysML models can be found in, e.g., (Schönherr and Rose 2009). In this respect, as stated in Section 1, this work extends and improves our previous contribution (Bocciarelli, D'Ambrogio, and Fabiani 2012), which introduced a model-driven method for the generation of executable HLA code from a SysML specification. The *HLAcloud* framework adopts a workflow that redesigns the one proposed in our past work according to the IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP) (IEEE 2010d). Moreover, the existing model-to-model and model-to-text transformations have been extended both to allow the integration with the PlanetLab cloud infrastructure and to exploit the enhancements promoted by the novel HLA-Evolved standard (IEEE 2010a).

The provision of simulation systems according to a SaaS paradigm has been addressed by various contributions, such as in (Marr, Storey, Biles, and Kleijnen 2000) and (Xie, Teo, Cai, and Turner 2005).

In (Marr, Storey, Biles, and Kleijnen 2000) a java-based architecture for carrying out web-based simulations with the execution of replicated experiments on networks of processors is proposed. In (Xie, Teo, Cai, and Turner 2005) an extension of the HLA standard is introduced, in order to support the provisioning of HLA-based distributed simulation services on the grid. More specifically, such a contribution introduces an architectural framework in which the RTI-federate interactions are mediated by a proxy and the RTI services are exposed as Grid services. Differently from such contribution, this work is based on the HLA-Evolved standard without requiring extensions to enact the cooperation of HLA-based federates available either in a cloud infrastructure or in the Web.

The specific adoption of a cloud infrastracture to provide distributed simulation services is a novel issue that has just recently started to be addressed, such as in (Fujimoto, Malik, and Park 2010) and (Tsai et al. 2011). In (Fujimoto, Malik, and Park 2010) the authors argue that cloud computing eases the use of distributed simulation techniques as it eliminates the need to purchase, operate and maintain the needed computational infrastructure at the local site. Moreover, the cloud computing paradigm allows to provide distributed simulation software as-a-service without dealing with the several complications of executing distributed simulation code, offering the potential to exploit this technology in an effective and effortless way. In (Tsai et al. 2011) a multi-tenancy architecture is introduced for providing simulation services according to a SaaS delivery paradigm. Specifically, such contribution focuses on the definition of an ontology and a set of tenant policies to support the creation of flexible simulation models able to satisfy the requirements of different tenants. Similarly to such contributions, this work aims to propose a framework to provide innovative simulation services built on top of a cloud infrastructure. Nevertheless, in this work model-driven techniques are used to generate federates from SysML specifications and to execute the related federation onto PlanetLab. Indeed, the so-obtained simulation services are specifically tailored to support system developers in studying the behavior of relevant systems at design time.

Finally, it should be underlined that, unlike all such aforementioned contributions, even though *HLAcloud* specifically leverages SysML and PlanetLab, other standards and technologies can be introduced by extending or modifying the proposed set of model transformations.

3 THE *HLAcloud* FRAMEWORK

The *HLAcloud* framework is inspired by the IEEE DSEEP, a standardized and recommended process for developing interoperable distributed simulation federations (IEEE 2010d). The framework is founded on model-driven principles and standards and exploits two UML profiles, namely *SysML4HLA* and *HLAProfile*, to support the generation of both the model and the implementation of the HLA-based distributed simulation.

The rationale of the framework is depicted in Figure 1. The system developer uses the *HLAcloud* framework to carry out two different actions. First, the framework automates both the development of

the HLA-compliant Java code implementing the federation that simulates the system under study and the deployment of federates ($F_1..F_n$) onto the nodes of the cloud-based infrastructure ($Node_1..Node_n$). It should be noted that the federation may include existing third-party federates. Second, once the federation has been deployed, the *HLAcloud* framework allows system developers to start the federation execution and collect the related results. The federation may include additional federates available on traditional LAN/WAN infrastructures ($EF_1..EF_k$).

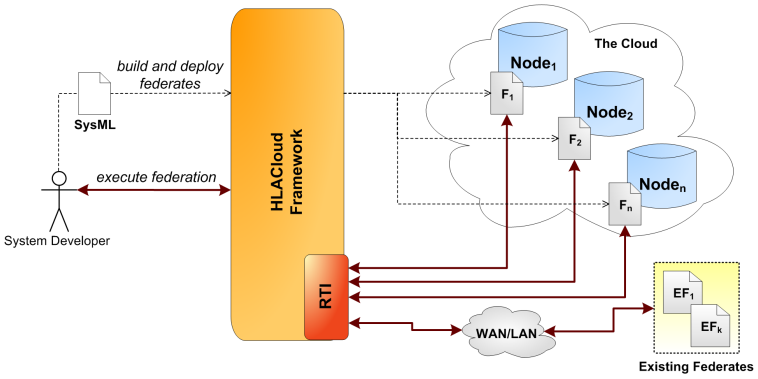

Figure 1: Framework rationale.

A detailed description of the federation development, deployment and execution workflow is given in Figure 2. The process is contextualized in a more general system development scenario, where simulation is introduced as the key technology in order to obtain an early evaluation of the system behavior.

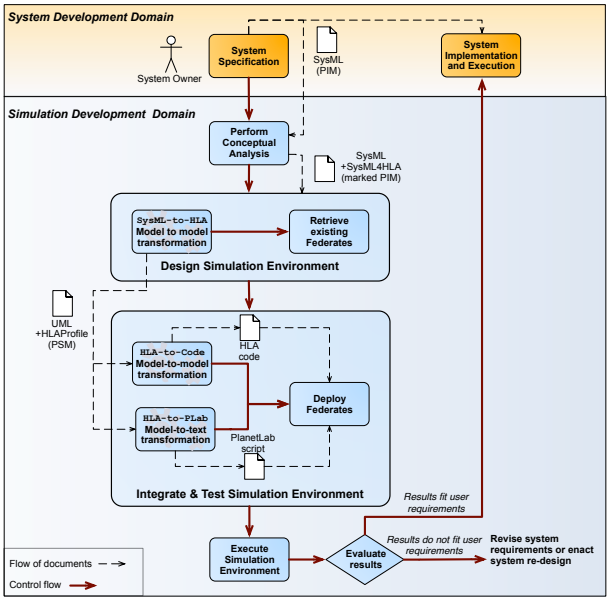

Figure 2: HLAcloud model-driven workflow.

According to Figure 2, the system under study is initially specified in terms of a SysML model (e.g., block definition diagrams, sequence diagrams, etc.). According to the model-driven terminology (OMG 2003), such a model constitutes the platform independent model (PIM) of the system. In the system development domain, the system engineer builds the system model without being concerned with details relevant to the simulation model.

The SysML model represents the input artifact of the distributed simulation development sub process. In this respect, simulation engineers carry out a conceptual analysis of the required simulation and use the *SysML4HLAProfile* to annotate the PIM in order to enrich such a model with information needed to derive the HLA simulation model. Specifically, the *HLA profile* allows to specify both how the system has to be partitioned in terms of federation/federates and how system model elements have to be mapped to HLA model elements, such as object class and interaction class. Then, the design simulation environment step is executed. This step takes the marked PIM and the *HLA profile* as input to the `SysML-to-HLA` model-to-model transformation, in order to automatically obtain an UML model, annotated with stereotypes provided by the *HLA profile*, which represents the HLA-based distributed simulation model. The latter, according to the model-driven terminology, constitutes the platform specific model (PSM). According to the DSEEP, the design simulation environment is also concerned with the discovery of existing federates suitable to be integrated in the distributed simulation. The integrate & test simulation environment step is then executed to implement the distributed simulation. The Java/HLA-based code of the simulation model is generated by use of the `HLA-to-Code` model-to-text transformation, while the scripts to configure the PlanetLab slice and to deploy the federates are obtained by use of the `HLA-to-PLab` model-to-text transformation.

Finally the distributed simulation is executed and the results are evaluated to determine whether or not the system behavior satisfies the user requirements and constraints. According to such an evaluation, the SysML specification is used to drive the implementation of the system, or a system re-design has to be planned.

The following sections briefly describe the UML profiles and the model transformations used throughout the simulation development process.

4 EXTENSION OF THE UML AND THE SysML MODELS

The workflow described in the previous section makes use of several SysML and UML models which are used both to represent the system under a system and software perspective and also to embody the details required to support the model transformations upon which the proposed framework has been built. To this purpose, such models have been extended by use of stereotypes provided by the *SysML4HLA* and *HLAProfile* UML profiles, introduced in (Bocciarelli, D'Ambrogio, and Fabiani 2012). The *SysML4HLA* profile is used to annotate a SysML-based system specification to support the automated generation of the HLA-based distributed simulation models. Differently, the *HLAProfile* is used to annotate an UML diagram in order to represent HLA-based implementation details. For the reader convenience such profiles are briefly recalled in the following subsections. A detailed description of *SysML4HLA* and *HLAProfile* can be found in (Bocciarelli, D'Ambrogio, and Fabiani 2012).

4.1 SysML4HLA

The *SysML4HLA* profile has been introduced in (Bocciarelli, D'Ambrogio, and Fabiani 2012), and has been here revised, to annotate SysML models with HLA-specific data, so as to drive the automated generation of the HLA code that implements the distributed simulation of the specified system. *SysML4HLA* provides a set of stereotypes (or metaclass extensions) that extend the `block` element of SysML, which in turn is an extension of UML `class` metaclass. The introduced stereotypes model the basic elements of an HLA simulation: federation, federates, object classes and interaction classes.

4.2 HLAProfile

The role of the *HLAProfile* is to allow the representation of concepts, domain elements and relationships required to represent a HLA-based simulation under an implementation-oriented perspective. The *HLAProfile* is inspired by the HLA metamodel proposed in (Topçu, Adak, and Oğuztüzün 2008) and includes several stereotypes grouped into a number of packages.

The *HLADatatypes* package includes the datatypes of the several attributes used to specify the stereotypes of the OMTKernel package, such as the `PSKind` attribute, to describe the publish/subscribe capabilities of a federate, or the `UpdateKind` attribute, to represent the policy for updating an instance of a class attribute.

The *OMTKernel* package includes the stereotypes defined in the HLA Object Model Template Specification (IEEE 2010c). The provided stereotypes model HLA concepts such as a federate or the object classes and interaction classes that are part of a federation. The provided stereotypes and the related associations are shown in Figure 3.

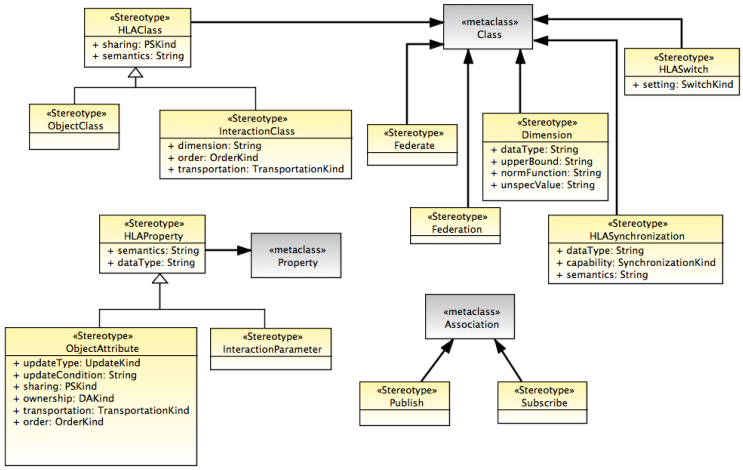

Figure 3: OMT Kernel package.

The *BMM (HLA Behavioral Metamodel)* package includes the stereotypes derived by the HLA Behavioral Metamodel (Topçu, Adak, and Oğuztüzün 2008), which provides the UML extensions needed to model the observable behaviour of the federation, as summarized in Table 1.

Table 1: BMM Stereotypes.

Stereotype	*Extends*	*Description*
`<<initialization>>`	UML::Message	Identifies messages exchanged to setup the distributed simulation infrastructure
`<<action>>`	UML::Message	Identifies a local action executed by a federate that does not require any interaction with other federates
`<<message>>`	UML::Message	Identifies a communication between two federates

5 MODEL TRANSFORMATIONS

This section outlines the rationale of the model transformations at the basis of the workflow depicted in Figure 2, namely the `SysML-to-HLA` model-to-model transformation, the `HLA-to-Code` model-to-text transformation and the `HLA-to-PLab` model-to-text transformation.

5.1 SysML-to-HLA model-to-model transformation

The `SysML-to-HLA` model transformation takes as input a SysML model representing the system that is going to be simulated and yields as output an UML model that represents the corresponding HLA-based distributed simulation. The input model is annotated with stereotypes provided by the *SysML4HLA* profile,

while the output model makes use of the *HLAProfile*. Specifically, the source model consists of the following SysML diagrams:

1. a ***Block Definition Diagram (BDD)***, which defines the static stucture of the system to be developed in terms of its physical components;
2. a set of ***Sequence Diagrams (SDs)***, which specify the interactions among the system components.

The target model is specified by the following UML diagrams:

1. a ***Class Diagram (CD)***, which defines the HLA-based structural model and shows the partitioning of the simulation model in terms of federates, along with the HLA resources (i.e., object classes and interaction classes) published and/or subscribed by federates;
2. a set of ***SDs***, which define the HLA-based behavioral model and show the interactions among federates and RTI.

The model transformation has been implemented by use of the QVT language (OMG 2008), which is provided by the Object Management Group (OMG) as the standard language for specifying model transformations that can be executed by available transformation engines (Eclipse Foundation b).

The mapping rules are summarized in Table 2, while the patterns for generating the publish/subscribe relationships are depicted in Figure 4.

Table 2: Mapping rules of SysML elements to UML elements.

SysML (with *SysML4HLA* profile)		UML (with *HLA* profile)	
Element (Stereotype)	*Diagram*	*Element (Stereotype)*	*Diagram*
Block (Federation)	BDD	Class (Federation)	CD
Block (Federate)	BDD	Class (Federate)	CD
Block (ObjectClass)	BDD	Class (ObjectClass)	CD
Signal (InteractionClass)	BDD	Class (InteractionClass)	CD
Block Attribute	BDD	Attribute (objectParameter)	CD
Signal property	BDD	Attribute (interactionParameter)	CD

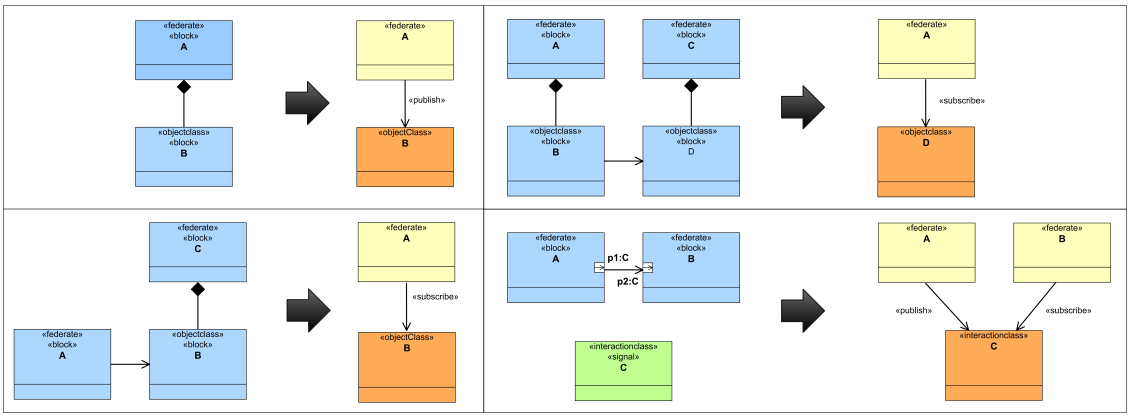

Figure 4: Publish/Subscribe relationships generation patterns

5.2 HLA-to-Code model-to-text transformation

The `HLA-to-Code` model-to-text transformation takes as input an UML model representing the HLA-based simulation system and yields as output the Java/HLA implementation code. The code generation makes use of Pitch pRTI (Pitch 2012) as the RTI implementation and Java as the language for implementing federates and ambassadors. The model-to-text transformation has been implemented by use of Acceleo (Eclipse Foundation a), the model-driven Eclipse plugin for code generation.

The implementation of the proposed transformation includes the following templates:

- *generateFederate*: generates a Java class for each UML class stereotyped as `<<federate>>` ;
- *generateObjIntClass*: generates the XML tags that serialize object classes and interaction classes, for each UML class stereotyped as `<<objectClass>>` or `<<interactionClass>>`. The resulting XML file represents the FOM (Federation Object Model), i.e., the description of objects, attributes and interactions shared by federates. Moreover, this template creates, within each class generated by the generateFederate template, the method for publishing and subscribing resources, according to the Publish/Subscribe relationships of the HLA-based structural model;
- *generateAmbassador*: generates a set of Java classes constituting the implementation of the required federate ambassadors;
- *completeFOM*: generates the FOM sections related to switches, synchronization points and dimensions, while the corresponding elements are specified by the HLA-based structural model.

5.3 HLA-to-PLab model-to-text transformation

The *HLA-to-PLab* transformation consists of the following five steps:

1. *retrieve active nodes*
2. *select a node for each federate*
3. *add selected nodes to the slice*
4. *deploy federates*
5. *execute the simulation*

There are several ways to carry out such sequence of steps, the main one based on the use of the PlanetLab Central API, a set of common abstractions provided by PlanetLab to ease the access to its resources. On the top of this API many tools or command shells have been developed, most of all based on the *Python* scripting language. Specifically, the *pShell* tool, a Linux shell like interface providing a few basic commands to interact with a Planetlab slice (Maniymaran 2007), has been used to implement the scripts required to execute the aforementioned steps. Listing 1 gives the *pShell* commands for retrieving active nodes, adding nodes, deploying files and executing a command, respectively.

```
1 ...
2 plist -a -o active_nodes.txt
3 ...
4 addnodes -f nodes_to_add.txt
5 ...
6 put filename nodeID
7 ...
8 cmd launchcommand nodeID
9 ...
```

Listing 1: `pShell` pShell script

The adoption of a model-driven approach allows to easily integrate additional cloud infrastructure (e.g., Amazon E2C or Google Cloud Platform) by adding/modifying the relevant model transformations.

6 EXAMPLE APPLICATION

This Section presents an example application of the proposed model-driven method to the development of an aircraft.

6.1 System Specification

The first step of the method includes the definition of the system model by use of the SysML notation. For the sake of brevity, this example only takes into account the diagrams needed to derive the HLA simulation model and, consequently, the distributed simulation code. Specifically, the SysML aircraft model includes the following diagrams:

- A BDD that models the aircraft components (e.g., the structural parts such as ailerons and the elevator, the autopilot and the powerplant) and their relationships;
- A set of SDs that describe the interactions between system components.

6.2 Conceptual analysis: the specification of the aircraft federation

At the second step, the SysML model is refined and annotated with the stereotypes provided by the *SysML4HLA* profile. This step adds to the SysML model the information needed to map each SysML domain element to the corresponding HLA domain element and makes the model ready to be automatically processed by the model transformations carried out in the next steps. As an example, Figure 5 and Figure 6 show the BDD and a SD that specifies the interactions between the `Autopilot` (composed by `ControlLogic`, `GPS` and `Gyroscope` components) and the `Structure` (composed by `Aileron`, `Elevator` and `PositionSensor` components) components, respectively.

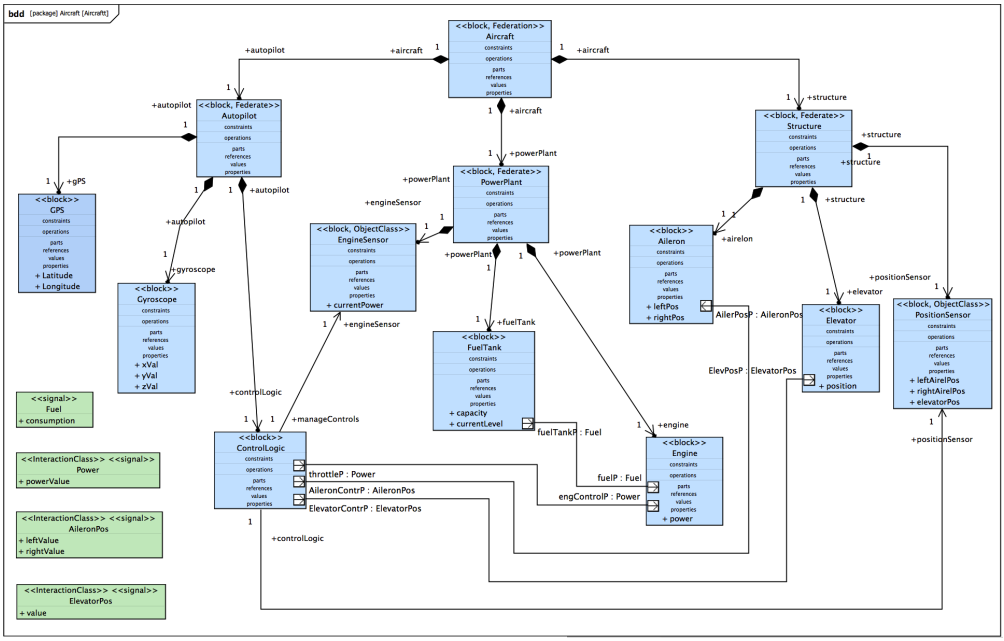

Figure 5: Source model: block definition diagram.

6.3 Design the federation

The SysML model annotated with the *SysML4HLA* profile is given as input to the `SysML-to-HLA` model-to-model transformation, to generate the HLA-based distributed simulation model, which is specified by

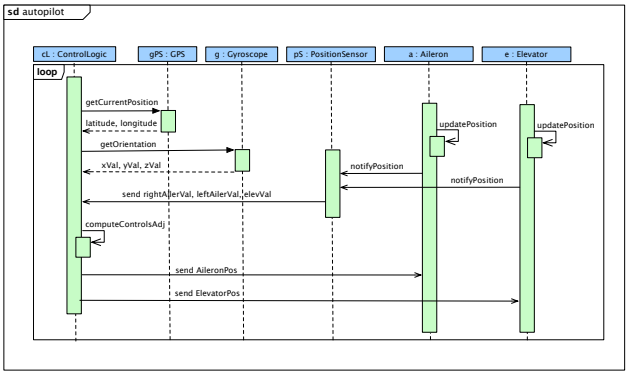

Figure 6: Source model: sequence diagram (interaction between Autopilot and Structure).

use of a CD (i.e., the HLA-based structural model) and a set of UML SDs (i.e., the HLA-based behavioral model). As an example, Figure 7 and Figure 8 depict the CD and the SD corresponding to the BDD and the SD shown in Figure 5 and Figure 6, respectively. Once the federation structure has been specified, the federation design step also includes the optional execution of a discovery that aims to retrieve already available federates to be integrated in the federation. The present example assumes the federation is fully implemented from scratch.

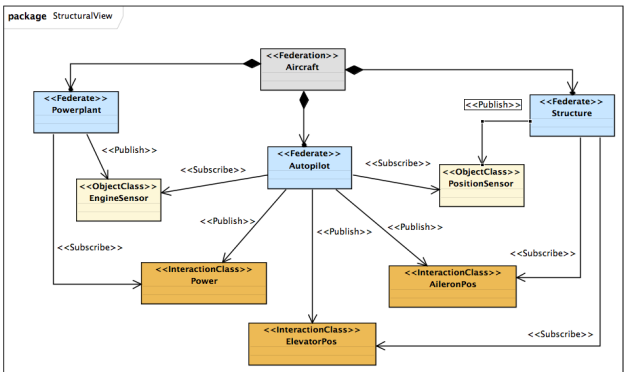

Figure 7: Target model: structural view.

6.4 Integrate the federation

Once the federation model has been specified, the `HLA-to-code` and the `HLA-to-PLab` model-to-text transformations are executed. Such transformations take as input the HLA-based distributed simulation model produced in the previous step and yield as output both the Java/HLA code which implements the HLA-based distributed simulation and the script to configure the PlanetLab slice and deploy such federates. It should be noted that the proposed method does not generate the complete code that implements the HLA-based distributed simulation. More specifically, the `HLA-to-code` transformation allows to generate a template of the Java classes that contain the class structure (i.e., constructor, method and attribute declarations, exception management, etc.) and most of the HLA-related code (i.e., data types definition, RTI interaction methods, etc.), while the code implementing the federate simulation logic has to be added manually.

The final step of the method deals with the simulation execution and the results evaluation. For the sake of conciseness, this step is not discussed, being the example application mainly focused on the use of the proposed framework to build and deploy the HLA-based distributed simulation.

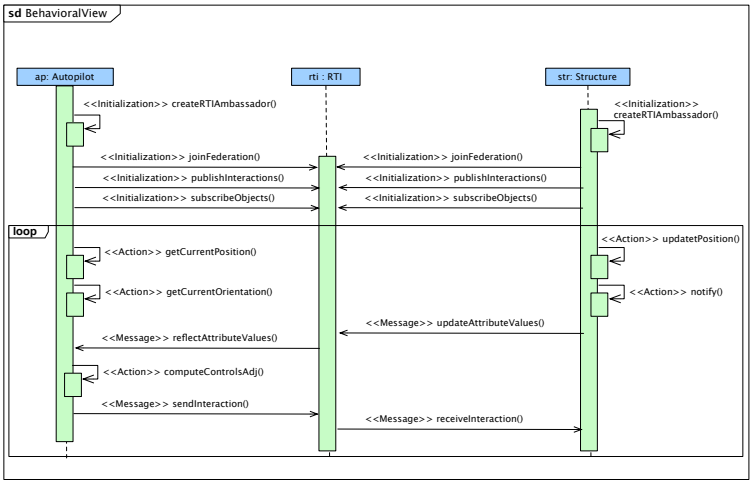

Figure 8: Target model: behavioral view (interaction among Autopilot, Structure and RTI).

7 CONCLUSIONS

DS-based solutions provide an effective approach for the design-time analysis of complex systems, but can be time- and effort-consuming.

In this respect, this paper has introduced the *HLAcloud* framework, which provides a model-driven approach for the automated generation of Java/HLA code from SysML models, thus reducing the DS development effort.

The proposed framework makes use of a set of model transformations that support the entire HLA-based federation development and execution process, from the specification of the SysML model of the system under study down to the generation, cloud-based deployment and execution of the corresponding HLA-based DS implementation. The HLACloud framework has been illustrated by use of an example application to the SysML-based development of an aircraft. The example has been kept simple to show the effectiveness of the HLACloud framework without dealing with unnecessary and lengthy details. Applications to real world systems to validate the proposed framework in different domains are currently in progress.

REFERENCES

Bocciarelli, P., A. D'Ambrogio, and G. Fabiani. 2012. "A Model-driven Approach to Build HLA-based Distributed Simulations from SysML Models". In *Proceedings of the 2nd International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, SIMULTECH '12, 49–60.

Eclipse Foundation. "Acceleo". Website: http://www.acceleo.org/pages/home/en.

Eclipse Foundation. "QVT Transformation Engine". Website: http://www.eclipse.org/m2m.

Fujimoto, R. M., A. W. Malik, and A. Park. 2010. "Parallel and distributed simulation in the cloud". *SCS M&S Magazine* 3:1–10.

IEEE 2010a. "IEEE 1516 - Standard for Modeling and Simulation High Level Architecture - Framework and Rules".

IEEE 2010b. "IEEE 1516.1 - Standard for Modeling and Simulation High Level Architecture - Federate Interface Specification".

IEEE 2010c. "IEEE 1516.2- Standard for Modeling and Simulation High Level Architecture - Object Model Template (OMT) Specification".

IEEE 2010d. "IEEE 1730 - Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP)".

Maniymaran, B. 2007. Available at http://cgi.cs.mcgill.ca/~anrl/projects/pShell/docs/pShell-1.7.pdf.

Marr, C., C. Storey, W. E. Biles, and J. P. C. Kleijnen. 2000. "A Java-based simulation manager for web-based simulation". In *Proceedings of the 2000 Winter Simulation Conference*, edited by J. A. Joines and R. R. Barton, WSC '00, 1815–1822. San Diego, CA, USA: Society for Computer Simulation International.

Menascé, D. A., and P. Ngo. 2009. "Understanding Cloud Computing: Experimentation and Capacity Planning". In *Proceedings of the 2009 Computer Measurement Group Conference*.

OMG 2003. "MDA Guide, version 1.0.1.".

OMG 2008. "Meta Object Facility (MOF) 2.0 Query/View/Transformation, version 1.0.".

Pitch 2012. "pRTIe". Website: http://www.pitch.se/.

PlanetLab 2013. Website: http://www.planet-lab.org/.

Schönherr, O., and O. Rose. 2009. "First steps towards a general SysML model for discrete processes in production systems". In *Proceedings of the 2009 Winter Simulation Conference*, edited by A. Dunkin, R. G. Ingalls, E. Yücesan, M. D. Rossetti, R. Hill, and B. Johansson, WSC '09, 1711–1718. USA: Winter Simulation Conference.

Topçu, O., M. Adak, and H. Oğuztüzün. 2008, July. "A metamodel for federation architectures". *ACM Trans. Model. Comput. Simul.* 18 (3): 10:1–10:29.

Tsai, W., L. Wu, B. Xiaoying, and J. Elston. 2011. "P4-SimSaaS: Policy specification for multi-tendency simulation Software-as-a-Service model". In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Jain, R. R. Creasey Jr., J. Himmelspach, K. P. White, and M. C. Fu, WSC '11, 3067–3081. USA: Winter Simulation Conference.

Xie, Y., Y.-M. Teo, W. Cai, and S. Turner. 2005. "Service provisioning for HLA-based distributed simulation on the grid". In *Proceedings of the Workshop on Principles of Advanced and Distributed Simulation, PADS 2005*, 282–291.

AUTHOR BIOGRAPHIES

PAOLO BOCCIARELLI is a postdoc researcher at the Enterprise Engineering Department of the University of Rome Tor Vergata (Italy). His research interests include software and systems engineering and business process management, specifically with regards to the areas of modeling and simulation and model-driven development. His email address is paolo.bocciarelli@uniroma2.it.

ANDREA D'AMBROGIO is associate professor at the Enterprise Engineering Department of the University of Roma TorVergata (Italy). His research interests are in the fields of model-driven software engineering, performance engineering, distributed and web-based simulation. His email address is dambro@uniroma2.it.

DANIELE GIANNI is a requirements engineering consultant at EUMETSAT and assistant professor at the "Gugliemo Marconi" University (Italy). His research interests are in the area of systems engineering, including requirements engineering, model-driven engineering and simulation. His email address is d.gianni@unimarconi.it.

ANDREA GIGLIO is a Ph.D student at the Enterprise Engineering Department of the University of Rome "Tor Vergata" (Italy). His research interests include model-driven software engineering and business process management. His email address is andrea.giglio@uniroma2.it.