A heterogeneous model integration and interoperation approach in distributed collaborative simulation environment

Miaomiao Fan*, Jianliang Zhang and Yushun Fan

Department of Automation, Tsinghua University, P.O. Box 100084, Beijing, China
Fax: +86-010-62789650
E-mail: fanmm06@mails.tsinghua.edu.cn
E-mail: jl-zhang07@mails.tsinghua.edu.cn
E-mail: fanyus@tsinghua.edu.cn
*Corresponding author

Abstract: The integration and interoperation among heterogeneous distributed models is an important technique to realise distributed collaborative simulation for complex product development. HLA and web service technology are employed to facilitate the integration in this paper. First, the HLA-enabling method based on HLA agent model is proposed to make models created through commercial software conforming to HLA standard. In order to make up the drawbacks of HLA based simulation in interoperability on the WAN, a collaborative simulation framework by combining HLA with web service merits is proposed and described in details. Finally, a prototype system for heterogeneous models integration in forging manipulator system is implemented, where a method of developing ABAQUS engine agent is present, which could solve the difficulty of wrapping ABAQUS software. The case study verifies the high fidelity and effectiveness of our integration and interoperation approach.

Keywords: high level architecture; HLA; integration; interoperation; multidisciplinary collaborative simulation; web service.


Biographical notes: Miaomiao Fan is a PhD candidate at Tsinghua University. She received her undergraduate degree in Information Engineering from Northwestern Polytechnical University, Xi’an. Her research interests are in the area of distributed modelling and simulation techniques, HLA, SOA.

Jianliang Zhang is a PhD candidate at Tsinghua University. He received his undergraduate degree in Automation from Beijing Institute of Technology, Beijing. His research interests are in the area of distributed modelling and simulation techniques, HLA, SOA.
1 Introduction

The design and development of a complex product is usually involved in multidisciplinary field knowledge, including mechanics, kinetics, kinematics and so on. As a single domain simulation cannot analyse the multidisciplinary behaviour of a complex product comprehensively and accurately, it is indispensable to adopt multidisciplinary collaborative simulation, which is able to integrate models in different disciplines and simulation tools together, and facilitate better communication and cooperation among different development teams as well. As the different disciplinary models are usually created by different simulation software, which do not support interoperability between each other, we pay much attention to how to integrate these heterogeneous models and make them interoperable with each other.

Over the past few years, the high level architecture (HLA), has been a component integration standard for large-scale distributed simulation systems. The HLA was initiated by the US Defense Modeling and Simulation Office (DMSO) in 1996. Unlike some of its predecessor simulation interoperability standards, the HLA make improvement on separating the data model and the architecture’s semantics from the functions or methods for exchanging information. It defines the rules and interface specification to support reusability and interoperability among simulation object models (SOMs) or federates, and reusability and interoperability are the two most important features of the HLA (Allen et al., 1998).

Although, HLA shows great advantage in supporting reusability and interoperability, lots of practical applications indicate that currently HLA/RTI products were actually insufficient in these two features (Blais, 2005), as mentioned below:

1 About interoperability: It is difficult to use HLA/RTI on wide area network (WAN) because of security issues such as firewalls. The communication between federates could not carry through on WAN normally. With the increase of simulation scale and complexity, geographically distributed simulation on WAN comes to be the trend of necessary.

2 About reusability: A simulation, or a federation in HLAs terminology, consists of a set of logically related simulators, called federates. Federates communicate with each other through the RTI. In this way, problems of simulation codes reuse are worked out in some manner. But it is hard to assemble simulation applications dynamically, because of high coupling between federate codes and local RTI components (LRC).
The HLA, as an advanced simulation architecture, has to be continually developed and evolved, has to use other domain related technology for reference, thus, it can become more perfect and show a wider application prospect in military and civil field.

Service oriented architecture (SOA), another milestone in distributed computing, is characterised by loosing coupling and facilitating systematic reusability and interoperability. SOA takes the existing software components residing on the network and allows them to be published, invoked and discovered by each other. Some of the first SOAs were based on CORBA or DCOM technologies. Today, SOA based on web service and XML/SOAP technology is in focus for civilian enterprise integration. Web services are a new breed of web application. They are self-contained, self describing, modular applications that can be published, located, and invoked across the web. Web services perform functions, which can be anything from simple requests to complicated business processes. As an emerging pattern of web application, a web service is an effective mechanism of information integration. The project of extensible modelling and simulation framework (XMSF) initiated by DMSO had pointed out in 2002 project summary report (Brutzman et al., 2002), that “work in web services appears to be an appropriate basis for organising and composing the many necessary capabilities of web/XML and internet/networking needed for M&S applications.”

As web technology shows great promise for achieving interoperability and reusability of diverse applications, it makes much sense to make up the drawback of HLA by introducing web technology into simulation field. In this paper, we also make some effort on the combination of HLA and web service technology.

This paper is organised as follows: The next section gives a review on HLA-based simulation and web service technology; in Section 3, some key techniques are introduced in details; In Section 4, a prototype system for forging manipulator is developed according to the method proposed in this paper; a conclusion is given in Section 5.

2 Literature review

2.1 HLA based collaborative simulation among heterogeneous simulators

HLA is initiated to provide an open, flexible and adaptable architecture. It enables interoperation between diverse simulations and reusability of legacy models. HLA consists of three parts: HLA rules, interface specification (IEEE Standard, 2001) and object model template (OMT). HLA attempts to specify a standard interface between simulations and separates these interfaces from the implementation of any specific simulation. HLA represents the software system architecture and run time infrastructure (RTI) is its implementation. Guideline for designing and building HLA federation is initiated as federation development and execution process (FEDEP). Within HLA, federations are defined as a group of federates forming a community. The formation of federations is based on a number of simulation components called federates that exchange information in the form of objects or interactions.

HLA adopts object model (OM), which contains the required interaction data and relevant information in the federation execution, to describe federation and each federate. In the HLA OMT, HLA defines two types OMs. One is the federation object model (FOM), describing the possible messages among federates of a federation; the other is the simulation object model (SOM), representing the capabilities of a federate to interact...
A heterogeneous model integration and interoperation approach

with others (Shen et al., 2002). Both of these two OMs aim to facilitate interoperability of simulation system and reusability of simulation components.

HLA takes concern on how to construct a federation assuming that the federates are existent rather than how to construct a federate. It mainly focuses on how to realise the federation integration. RTI, the service procedure according to HLA, realises all the service function of HLA interfaces specification. RTI is portability and interoperability. It can be perceived as a soft bus of simulation.

As HLA facilitate the integration of diverse simulation and the management of simulation execution, nowadays it is widely applied in military and civil field, such as defence, aerospace, manufacturing, complex product development and so on.

Figure 1 shows the architecture of HLA-based collaborative simulation in complex product development environment. The federation consists of five members; each federation member is called a federate. If the federate is designed conforming to HLA standard, it can be directly plugged into RTI soft bus, e.g., the running management and data collector. For the other three models created by commercial software, which do not provide the interfaces with HLA, they must be made some transformation with the help of an adaptor so as to reuse the existing models. The original model in combination with the adaptor is called a federate. Only federate can join the federation through calling RTI service. Thus, model transformation is a key technique to integrate diverse simulations on the RTI platform.

![Figure 1](https://via.placeholder.com/150)

**Figure 1** The architecture of HLA based collaborative simulation (see online version for colours)

2.2 Web service

Web service adopts open standard protocol specification and has the advantage of platform-independent, high integration ability and interoperability, loose coupling and good encapsulation. It brings revolutionary change for software issue way, software reuse, communication across the firewall as well as application programme integration, business integration and so on.

Web services are XML-based modules with abilities of self-description and self-comprehension, which can be invoked through standard web protocol (Cerami, 2002). Services are defined using a web service definition language (WSDL) and published in directories based on the UDDI meta-service. One of the main advantages is
that web services are supported by a wide range of operating systems and development/deployment environments.

At the moment the web services area mainly consist of four technologies in combination that provide an implementation of an SOA.

1. HTTP as the primary network protocol
2. SOAP/XML for the payload format
3. UDDI for service registry
4. WSDL to describe the service interfaces.

A web service consumer interacts via XML based messages on a transport protocol with a web service registry or a web service provider. The dynamic binding to the consumed web services is done using the information from a web service registry. A web service provider is the provider of the web service, publishes information about the web service to the registry and offers a programmatic access the consumer can bind to. The web service registry contains information about location and offered web services from all web service providers.

2.3 XMSF

With the development of web technology, the emerging and evolvement of middleware technology based on web – web services and XML technology, which provide common data description and standard interact protocol for web based M&S. The concept of web based simulation first emerged in the middle of 1990s, when Fishwick (1996) proposed a web based simulation framework prototype, subsequently, web based simulation becomes a hot topic in simulation field. The advocator of web services in simulation is XMSF.

The XMSF is defined as a composable set of standards, profiles and recommended practices for web-based modelling and simulation (M&S), to realise the integration of heterogeneous simulation systems such as the HLA based simulation and traditional command control communication computer and intelligence (C4I) systems. Through the standard web services, the XMSF not only can improve the interoperability of the spectrum of DoD models and simulations including constructive, virtual, and live systems, but also can integrate legacy simulation frameworks, tactical systems and the increasingly important distance learning technologies (Brutzman et al., 2002).

XMSF proves to be successful in methodology and technology. From the system layer, the DoD M&S already owns many different application systems, and each can provide high compute capabilities respectively. If the commercial web technologies can be adopted as a common communication platform and ubiquitous-delivery framework, the pre-existing M&S systems can be utilised sufficiently to develop large scale software systems. Similarly, providing web interoperability to general M&S applications can provide broad new classes of capability for commercial, educational and scientific applications. From the technology layer, to bridge the gap that arises when different simulations are supported by diverse infrastructures, XMSF applies web services and grid computing approach in the technical level while at the conceptual level advocating the techniques from model driven architecture (MDA). Web services technology is the technical backbone for integration of simulation systems while MDA helps to realise
meaningful integration. HLA is inherited by XMSF and cooperates with web services and grid computing to provide interoperability between highly distributed systems and models with different granularity.

3 Some key techniques

3.1 HLA-enabling method based on HLA agent model

Nowadays, during the simulation fields, most of the analysis models are created through typical commercial software, such as Matlab, ADAMS. All of these simulation tools are weak in interoperability and do not support HLA standard and specification. In order to inherit and reuse the single domain model information and the modeling knowledge of experts, we propose a HLA-enabling method based on the HLA agent model to transform original model, making them compatible with HLA standard. Under this situation, the whole transformed model can be directly integrated into HLA-based collaborative simulation platforms as a federate.

The HLA agent model, a procedure object compliant with HLA standard, is a buffer link in nature between disciplinary simulation model and RTI simulation bus. During the simulation process, disciplinary simulation model is still running in the corresponding simulation software, through the outer programming interfaces of which, the HLA agent model operates on disciplinary model, and meanwhile takes charge of communication with RTI. With support of the HLA agent model, disciplinary model could realise data filtering according to publish/subscribe mechanism, information output according to updating object class attributes and sending interaction class parameters, acquisition of kinds of information according to executing calling and callback function of RTI service.

The principle of the HLA agent model is as Figure 2.

Figure 2 Principle of the HLA agent model

According to the analysis of federate function structure, we can see that, in a whole federate member, it mainly consists of two part procedure objects, one is in connection with domain simulation software and its simulation model and simulation software, and the other is related to information interaction with RTI. The former has high coupling with simulation software and simulation models, while the latter’s function is relatively
permanent, regardless of simulation software. In this way, we can develop the HLA agent model through the following approaches: defining the operation interfaces on the control of model simulating, wrapping the technical details of different simulation software and corresponding disciplinary models and realising the interface function of communication with RTI.

The framework of the HLA agent model is shown in Figure 3.

**Figure 3** Framework of the HLA agent model (see online version for colours)

The HLA agent model mainly contains:

Simulation engine agent. It is used to operate on disciplinary model as well as specific software utilised to construct the model. It wraps the function including operation on model input and output, modification of model parameters, controlling of simulation process.

Federate object class. Corresponding to high level model, it inherits the function of simulation engine agent and RTI agent and is used to realise data interface mapping and interaction maintenance.

RTI agent. It defines a two-direction interface and realises interoperation among different models through calling RTI service and RTI callback. This mechanism is implemented by RTI through two ambassador paradigms: RTIAmbassador and FederateAmbassador.

The principle of the HLA agent model is with the guidance of high level modelling result, different disciplinary federate object class realise the operation and maintenance on disciplinary model through simulation engine agent, where the interaction data during simulation runtime communicates with each other through RTI agent and thus multidisciplinary collaborative simulation is implemented.
3.2 Simulation engine agent

Simulation engine agent is relevant to specific simulation software and its implementation needs the secondary development interfaces provided by simulation software, in this section, we present the scheme of simulation engine agent.

In order to identify the function and boundary of simulation engine, we define simulation engine agent $\text{SimEngineAgent} = \langle \text{Model}, \text{Input}, \text{Output}, \text{Parameter}, \text{Operation}, \text{Configuration} \rangle$, where:

- **Input and output** refer to the input and output data of simulation component and the input and output data of the operation method, which are the sets of ports data type.
- **Parameter** refers to the parameters to be modified or the interim variables to be saved in the model.
- **Operation** is the function called when accessing the simulation component. It can be further represented as $\text{Operation} = \langle \text{SetInput, GetOutput, ModifyParameter, SaveVariable, StartSimulation, SuspendSimulation, ResumeSimulation, TerminateSimulation, SimTimeAdvance} \rangle$, in which the specific function includes setting input, getting output, modifying parameter, saving variable, start/suspend/resume/terminate simulation process.
- **Configuration** refers to setting workspace path, work path, path of model and temporary data saved, etc.

In order to fulfil the dynamic message exchange based on HLA among heterogeneous models created by commercial software, that is setting the output variable of one model to the input port of another model, at first these simulation models have to be divided into different federates, then make sure of the object class and its attributes each federate could publish and subscribe. To achieve the variables mapping effectively, we must utilise the the following APIs function of simulation software, thus, we can develop the simulation engine agent:

- acquire the output variable of model from the model workspace
- set the attributes values reflected as the new input variable in the model workspace
- start the simulation software and initialise the model
- execute relevant operation after simulation completion and close the software
- fulfil the step advancement of model simulation running.

3.3 RTI agent

In order to make federates interoperate through RTI, RTI defines a two-directed interface based on agent model, as shown in Figure 4. From the standpoint of federate developer, the agent, as procedure object, communicate with other federates according to calling its method. From the standpoint of RTI, the agent is used for federates calling in the form of library. During the running, a federate calls the RTI library function, and meanwhile it has to provide an agent object for RTI calling.
3.4 A framework of collaborative simulation on the internet

According to the encapsulation through HLA agent model, we can transform the disciplinary model created by commercial software into a federate conforming to HLA specification, then the federate is able to join the federation in a plug and play way. The federates belonging to the same federation via FOM, the common data exchange model, interact with each other. Thus, the reuse and interoperation of existing disciplinary model are achieved. However, the communication is only limited in the LAN, for the WAN communication between federates, it fails. With the increase of simulation scale, more and more simulations reside on the internet in a highly distributed environment. It becomes the necessity to establish a simulation platform supporting WAN simulation. We also make some effort on solving this problem. Web service technology has better integration and interoperability on the internet, in this section, a framework based on HLA in combination with web service is proposed, which tries to realise the communication among diverse simulations residing on the internet. The specific structure of the framework is shown in Figure 5.

For the LAN simulation, we still adopt the method mentioned in the former sections. Through the model transformation, disciplinary model could be wrapped as a federate so as to join the federation execution via RTI call and callback function.

For the WAN simulation, the legacy models or simulations should be first wrapped as a web service. Of course, if it is already a web service, this step can skip over. After the encapsulation through web service adaptor, the service could join the RTI with the help of a local mediator. The mediator performs two kind of function: one is calling web service, the other is calling RTI service.

The service encapsulation module and the mediator module are present in details as follows.
The service encapsulation module (Wang and Zhang, 2008):

1. A web server. It is the container for the HTTP requests, responses and SOAP engine.
2. SOAP engine. It provides a mechanism for sending and receiving SOAP messages, XML serialisation and de-serialisation.
3. Web service adaptor. It is the interface between legacy system and SOAP engine. It receives data from legacy system and serialises it in XML stream format. It de-serialises the XML stream and pushes the data to legacy system.
4. Legacy system. It consists of software modules and simulation system that should be reused in collaborative simulation.

The mediator module:

1. Message transforming and mapping. It consists of an engine which finishes the transformation of SOAP message and object through XML serialisation and de-serialisation, then realises the mapping between the object and FOM data.
2. HLA adaptor. It helps the simulation application join the federation execution according to the FOM data.

A framework like this no only facilitates the integration of diverse simulations on the internet, but also improves the reusability of legacy models or simulation systems. However, it is inevitably accompanied by other drawbacks. The combination between HLA with web service results in the complex structure of our framework and the difficult implementation.

**Figure 5** The framework of collaborative simulation based on HLA and web service (see online version for colours)
4 Prototype system for forging manipulator

The forging manipulator system is one of the key equipments in the manufacturing industry, which is characterised by complex non-linearity, strong coupling, multiple variables and multiple degrees of freedom. An appropriate simulation system for forging manipulator is able to reduce the production cost and resources consumption significantly, as a result, the developed countries pay much attention to the forging manipulator and have been carrying on related researches on M&S. However, the simulation on forging manipulator are limited in single disciplinary field (Kiefer and Shah, 1990; Ou and Balendra, 1998; Aksakal et al., 1997; Ren et al., 2007), thus, can not describe the complex multi-disciplinary coupling behaviour accurately. Then it is indispensable to develop the collaborative simulation platform to integrate single disciplinary research, and it is also a trend in simulation of manufacturing systems (Duvivier et al., 2003). In this paper, the collaborative simulation techniques are employed to implement prototype system in this section, where the approach to develop simulation engine agent of ABAQUS is given and a case study on forging and manipulating process is carried on to verify our method.

4.1 Simulation engine agent of ABAQUS

The forging and manipulating process of forging manipulator system is involved in different domain models, including control, hydraulics, mechanics, deformation of workpiece and so on in which the deformation subsystem is established by commercial FEA softwares. However, the research on developing simulation engine agent is mainly focused on the redevelopment of the commercial software Matlab/Simulink, which has already achieved much progress. By contrast, there is little work on the equally important FEA softwares.

ABAQUS is one of the most advanced general nonlinear FEA softwares. It is competent for the static and dynamic analysis of complex structures, tackling huge problem as well as emulating the highly nonlinear influence between structure and material. As a result, it is preferred to work on the techniques for developing ABAQUS simulation engine agent.

The approaches to redeveloping ABAQUS (Zhong et al., 2006) are as follows:

a) create new models through user subroutine to control the computing process and result
b) change default settings of ABAQUS through the environment initial file
c) fulfill pre-process modelling and post processing analysis of computing result through kernel script interface
d) create new GUI to interact with user through GUI script.

The script interface of ABAQUS is a programming library based on object, embedded with object-oriented programming language python, and provides a set of APIs, which are used to control ABAQUS/CAE and realise modelling and post process. ABAQUS extends Python with approximately 500 additional objects, and there are many relationships among these objects. In general terms, the ABAQUS OM is divided into the session, the Mdb, and the Odb objects, as shown in Figure 6 (ABAQUS Inc., 2008).
Every object has its corresponding data members and methods. After the object is created, one can use provided method to process the data member.

Here, we adopt the kernel script interfaces embedded with object-oriented programming language python to fulfil the important operations of ABAQUS engine agent.

### 4.1.1 Some operation fulfillment

SetInput(parameter) \{Modify InputFile\}, where the modification of .inp file should conform to the format and specification of .inp file, and the time parameter also can be altered by such an operation.

GetOutput(parameter) \{Exec python script\}, that is to run a python script file wrote by user self to access the data in result database .odb file. Data computed by ABAQUS is stored in the result database in object-oriented form. To access the result data, the path where the data is stored must be found in the first place, and then the data can be accessed by referring to the corresponding variable or through calling the methods of the data object.

The realisation of these two operations above demands in-depth communication between the developer and the modeller. The developer must know sufficiently well about the .inp file or the original .cae model and be able to identify which hierarchy the variable belongs to (refer to Figure 6).

For example, the following command refers to the sequence of field data contained in a FieldOutput object.

```python
odb.steps['10 hz vibration'].frames[3].fieldOutputs['U'].values[47]
```

To make the Odb commands available to the script, the user first needs to import the odbAccess module using the following statements:

```python
from odbAccess import*
from abaqusConstants import*
```

Task submit, suspend, resume, terminate etc, all of which can be realised by executing ABAQUS internal command. You can execute ABAQUS command through this statement \{system(cmd);\} in VC++.
Task submit: call abaqus job= job-name
Task suspend: abaqus suspend job=job-name
Task resume: abaqus resume job=job-name
Task terminate: abaqus terminate job=job-name

The paradigm of ABAQUS engine agent is shown as Figure 7. The client application program is used as the schedule engine to call the ABAQUS engine agent to complete the standard API operation. ABAQUS engine agent is to call the background ABAQUS engine through the execution of ABAQUS internal command. ABAQUS engine is to complete the operation on data in workspace through calling the user subroutine.

**Figure 7** ABAQUS engine agent paradigm (see online version for colours)

4.2 Case study

The horizontal passive submissive action of forging manipulator is of great importance for relieving reaction force caused by workpiece deforming. Otherwise, it will result in the key components of manipulator being overloaded or even worse accidents such as the tipping over of the equipment. As mentioned above, the forging and manipulating process is actually multidisciplinary, where exists the coupling between reaction force caused by workpiece deforming and clamps movement. When the reaction force reaches the given valve value of the cushion dashpot, it unlocks, and clamps moves backward. In the meantime, manipulator performs horizontal passive cushioning movement. The movement of clamps is dependent on reaction force, and simultaneously the reaction force is influenced by the movement of clamps.

Based on the interaction, the federation model consists of several domains principally, including the mechanical multi-body dynamics model, the control model and the hydraulic servo model of the manipulator, the deformation model of workpiece. Here, the deformation model is created using ABAQUS while the rest of domain models are created using the same software Matlab/Simulink, and they are finally connected to create a complete model. As a result, the interaction is mainly between Matlab/Simulink model and the ABAQUS model. The interaction relation model is shown in Figure 8.
With the aforementioned model transformation method, we wrap the Matlab model and the Abaqus model into FOMs, which conform to HLA standard and specification. More specifically, the wrapping of ABAQUS model adopts the proposed simulation engine encapsulation technique in this paper. After the transformation, each disciplinary model could join federation directly on our HLA/RTI platform as a federate.

Figure 9 shows the simulation result of the force load that the forging workpiece acted on clamps in this process (the effect of forging process is assumed to be one second).

Compared to the peak value of horizontal load in the single discipline experiment, the value on this point of multidiscipline experiment decreases significantly, and it results from the passive submissive action of manipulator. Such an action could reduce the load largely acted on the manipulator, and therefore guarantee the protection of the manipulator. Collaborative simulation experiment reflects on the spot’s real working condition, cushion dashpot valves are forced to unlock twice so as to reduce the horizontal load on manipulator.

It can be seen that this experiment verifies the high fidelity and validity of the multidisciplinary collaborative simulation technology proposed in this paper.
5 Conclusions

The multidisciplinary collaborative simulation technology is widely adopted for complex product design and development. HLA and web service technology are effective way to achieve the integration and interoperation among heterogeneous distributed simulations. As the models created by commercial simulation software do not support HLA standard, a HLA-enabling method based on HLA agent model is proposed in this paper. It is helpful to improve the reusability and interoperability of disciplinary model. More specifically, the development of simulation engine agent which is a vital part of HLA agent model is presented in detail; a framework of collaborative simulation based on HLA and web service is proposed to extend the simulation area from LAN to WAN; the application of the proposed method in forging manipulator system development is also implemented. In our future work, we will establish a platform on which highly distributed simulations across the internet is to be implemented.

Acknowledgements

This research was supported by National Basic Research Development Program (973 Program) of China, ID2006CB705407 and National R&D High-Tech Plan (863 Program) of China, ID 2007AA04Z150.

References


