A model-driven approach to multidisciplinary collaborative simulation for virtual product development

Heming Zhang, Hongwei Wang, David Chen, Gregory Zacharewicz

To appear in Advanced
Engineering Informatics
24 (2010) 167–179

A model-driven approach to multidisciplinary collaborative simulation for virtual product development

Heming Zhang [a,*], Hongwei Wang [b], David Chen [c], Gregory Zacharewicz [c]

[a] State CIMS Engineering Research Centre, Tsinghua University, Beijing 100084, PR China
[b] Engineering Design Centre, Cambridge University Engineering Department, Cambridge CB2 1PZ, UK
[c] LAPS-IMS, University Bordeaux 1, 351 Cours de la libération, 33405 Talence cedex, France

Abstract:

The design and development of complex artifacts and systems is shifting towards a distributed and collaborative paradigm. The simulation environments for such a paradigm, therefore, need to take into account the cooperation between design teams, i.e. supporting multidisciplinary simulation in a distributed environment. However, current simulation tools cannot fulfil this requirement as they have been developed to solve the specific problems from different disciplines. Although it's already possible to per- form multidisciplinary simulations by using several tools together, it is very difficult to implement it when these tools are distributed on the Internet. A solution which can support the integration of distributed simulation models at run-time is presented, involving a computational infrastructure and a high- level modelling approach. Specifically, the infrastructure is constructed by a novel combination of two distributed computing techniques to implement the synchronization of distributed models, as well as to ensure the interoperability at run-time. In addition, a model-driven approach is developed to bridge the high-level model of a simulation system and the infrastructure which implements this model. The solution is evaluated by making a comparison with other approaches, as well as by developing a prototype tool. It's shown in the evaluation that (1) it is viable to develop multidisciplinary simulations in a distributed environment using this solution; (2) the model-driven approach allows designers to focus only on the high-level structure of a design without getting concerned with the details of the infrastructure.

1. Introduction

Effective evaluation of design concepts at an early design stage can help to achieve shorter lead time and reduced costs. With the advent of inexpensive high-speed computing, it has become feasible to verify a design based on a virtual prototype, using modelling and simulation (M&S) technology [1]. Computer-aided engineering (CAE) tools are now widely used in a wide range of engineering disciplines, and M&S is further recognized as the primary means of design validation and verification [2,3]. Nowadays, development activities of complex artifacts and systems are increasingly undertaken by Multidisciplinary Design Teams (MDTs) in a virtual and collaborative environment. Therefore, the M&S environments also need to support the collaborative development of MDTs in a distributed environment, to allow developers only focus on their portion of work, as well as to enable the integration of models created using different tools or languages.

The challenges of fulfilling this requirement include the diversity of the simulation problems encountered during a design process, the using of different simulation tools, as well as the efficient interactions between models at run-time. Current approaches are to some extent insufficient in terms of these challenges. The motivation of this research is to develop a solution which can support the cooperation of MDTs and the integration of simulation models distributed on the Internet. Generally, a number of advantages can be obtained from such a solution as follows:

It supports the cooperative work of multiple users from MDTs distributed on the Internet, enabling cross-organization collaboration;
A variety of simulation applications can be created effortlessly by users with little expertise on distributed simulation;
A new simulation iteration can be started with minimal effort after changes are made to the simulation models;
Simulation models can be re-used by simulation applications in the future.

2. Related work

In this section, we review related work on developing collaborative simulation environments. Specifically, approaches based on

* Corresponding author.
E-mail address: hmz@mail.tsinghua.edu.cn (H. Zhang).

the integration of simulation tools are reviewed to find a better means for system implementation. Distributed computing techniques and distributed simulation standards are also surveyed to explore how an effective infrastructure can be constructed.

2.1. Approaches for building collaborative simulation environments

There are two main means to implementing collaborative simulation in terms of how simulation tools interoperate with each other, namely a monolithic approach and a modular approach. The monolithic approach uses a unified software environment, i.e. developing a whole simulation system using one single tool. It has the advantage of the consistent representation of the subsystems and the accurate solving of system equations. Requirements for choosing appropriate modeling and simulation environments were specified in [4], e.g. multi-domain scope, modular modeling and software re-use, reliability and efficiency of numerical integration, etc. Various modelling methods (e.g. port-based, object-oriented, diagram-based, etc) and simulation environments have been developed to implement this approach, e.g. VTB (diagram-based modeling) [5], 20-sim (port-based modeling) [6], NEWMOS (equation-based modeling) [7], DYMOLA (object-oriented modeling) [4], AMESim [8] and the composable simulation environment [1]. These tools speed up the resolving of simulation problems by encapsulating technical details of simulation models as building blocks, and make substantial contribution to the application of M&S in product development. However, these tools can only be used to solve a limited scope of engineering simulations. To solve this problem, a straightforward method is to divide a complex problem as a number of smaller and less complex problems which can be solved by tools currently available. The method leveraging the advantages of several tools is called the modular approach.

The modular approach, with a high level of modularity, allows using specialized software for each subsystem. Such an approach can generally be implemented by developing a block in a tool to communicate with, and access the simulation process of, another tool. For instance, the interfaces between ADAMS and SIMULINK have been used to develop multidisciplinary simulation [9]. The deficiencies of this approach are also obvious: (1) it is only appropriate for limited types of simulations; (2) only the popular simulation tools provide interfaces for each other. Another solution for the modular approach, also called Co-simulation [10], is proposed to develop an integrated environment which supports using multiple tools together and implements the run-time interactions between the models created using these tools. Research effort has been made towards developing Co-simulation and applying it to engineering problems [10–11]. It's indicated that the simulation development should evolve towards applications operating at the component or subsystem level, rather than just at system blocks [3]. Some research has also been undertaken to employ distributed computing technologies to integrate several simulation models during a simulation process.

2.2. Solutions based on distributed computing technologies

Over the last two decades, there are substantial bodies of research concerned with a more integrated use of information technology (IT) in the design process in many engineering sectors [12]. This tendency also provides impetus for the development of a new-generation engineering infrastructure and product development systems which will be distributed and collaborative [13–14]. In terms of the running of multidisciplinary simulation in a distributed environment, three aspects of work have been studied to implement the distributed connection: (1) basic networking protocol; (2) distributed computing technology; and (3) distributed simulation standards.

The first aspect emphasizes the realization of connection while not considering the decoupling of distributed communication and simulation running. For instance, Nakhimovski implemented Co-simulation by writing a TCP/IP interface [10]. Shen et al. developed a cooperative assembly environment which supports distributed simulation through TCP/IP communication [15–16]. The second aspect makes use of advances in the distributed computing domain to build up an infrastructure [17–18]. Although these approaches have good distributed computing capability, they are weak in synchronizing a number of simulation models. The last aspect is motivated by this pitfall, and uses distributed simulation standards in the derived solutions [19].

To support various simulation applications, using specialized distributed simulation standard can be more beneficial. High-Level Architecture (HLA) needs to be mentioned in terms of distributed simulation standard. It was first initiated for tactical simulation, and afterward was accepted as an IEEE standard for distributed simulation [20–21]. HLA can support a variety of simulation applications, e.g. continuous time, discrete event, hybrid time and even human-in-the-loop simulation. However, the application of HLA-based simulation platform in [19] is still restricted by the inherent drawbacks of HLA. First, HLA is hard to understand for users without knowledge about distributed simulation. Second, development of HLA-based simulation involves comprehensive coding work, and the developed codes are tightly coupled with simulation models. Third, implementation of HLA-based application strongly depends on the Run-Time Infrastructure (RTI, the software implementation of HLA), and the interoperability between different RTI products is not good enough [22].

Although it is difficult for HLA to fulfil these objectives, the advantages of HLA in supporting distributed simulation are still very manifest. Recently, Web Services technology has begun to be employed in the development of engineering software tools. Rosselo et al. identified a component framework for re-using proprietary CAE environments based on Web Services [23]. Johansson proposed a framework to manage simulation models by representing simulation models at a high-level abstraction and encapsulating them as Web Services [24]. To support collaborative engineering, Schubert et al. proposed to bridge the gap in the Virtual Organisation (VO) frameworks identified from the participant's perspective [25]. Dong et al. used Web Services to encapsulate and integrate distributed manufacturing resources [26]. In the review of Bakis et al., they highlighted the role of XML and Web Services in the development of product data sharing environments [27]. Web Services and HLA have been incorporated to supplement each other so that more capability can be achieved [22].

In summary, distributed computing technologies and simulation standards provide very promising solutions for engineering applications. However, none of them has been designed just for this purpose. Therefore, the adaptation and integration of these technologies is necessary.

3. Developing multidisciplinary simulation in a distributed environment

3.1. An example of multidisciplinary simulation

To give a simple example of multidisciplinary simulation, the titling process of an antenna will be illustrated in this section. Originally, this example was used to demonstrate the development of multidisciplinary simulation using the programming interfaces between MSC.ADAMS [28] and SIMULINK [29]. To simulate this tilting process, two models need to be created, namely the mechanical model and the control model. Based on this example, we can acquire the knowledge about developing multidisciplinary

simulations for more complex designs with three or more subsystems.

As shown in Fig. 1, development of such a simulation starts with the design requirements. The system design is further divided as the designs of several subsystems each of which represents a specific discipline, and can be evaluated by creating simulation models. The simulation results can be analyzed by experts to give feedback to a design process to guide decision-makings. The mechanical model has three components connected to the ground by a revolute joint, namely azimuth rotor, azimuth reduction gear, and azimuth plate. Two fixed joints are created to constraint the motion between the antenna support and the plate, as well as the motion between the antenna support and the elevation bearings. An antenna is connected to bearings by a revolute joint. The interactions between the two models creates a closed loop in which the control inputs from SIMULINK affect the MSC.ADAMS simulation, whereas the MSC.ADAMS outputs affect the control input levels.

Although this simulation can be performed based on the interfaces between SIMULINK and MSC.ADAMS, it is still very necessary to develop an approach which allows the distributed simulation models to be integrated at run-time. First, it is difficult to run this simulation if we add another model, e.g. an electronic model, to this example. Second, it is hard to access the run-time interaction data as the simulation is automatically executed by SIMULINK. With this approach, designers can obtain more insights into the run-time behaviour of a design by analyzing the detailed interactions between the subsystems. As simulation evolves, more information can be communicated to the design process so that design concepts can be improved by addressing the problems identified during the simulation process.

3.2. Developing a solution for multidisciplinary simulation

Several requirements need to be taken into account to perform the illustrated simulation accurately. First, the development of such a simulation, from the identification of subsystems to the implementation of each subsystem, can be undertaken by designers from MDTs in a distributed environment. Second, the two models can run separately and be distributed on the Internet whilst the run-time interactions can be guaranteed. Third, the two models need to be updated separately during each design iteration, as well as to be re-used in new simulations. Therefore, a computational infrastructure needs to be created to implement the distributed interactions, as well as to synchronize the two models. High-Level Architecture (HLA) can be viewed as a potential infrastructure.

HLA was initiated for military training, to promote the interoperability between diverse simulators and to improve the reusability of legacy models. Essentially it aims to implement the accurate run-time interactions between any two subsystems (federates) in a big simulation system (a federation). It consists of three parts in general: HLA rules, interface specification and Object Model Template (OMT). Specifically, the first part defines a set of rules to guarantee the accuracy of distributed simulation, for both a federation and an individual federate. The second part involves a set of interfaces that need to be implemented for HLA-based simulation regardless of what technology will be employed. OMT specifies how a concrete simulation problem can be modeled to form a HLA federation, facilitating the re-use of simulations. Data defined in OMT are categorized as two further types, namely a Federation Object Model (FOM) and a Simulation Object Model (SOM). The former defines the possible messages among federates of a federation while the latter defines the capability of a federate to interact with others [20–22].

Mechanisms for managing distributed simulation, offered by HLA, make it a promising candidate for developing a multidisciplinary simulation platform [19]. In our opinion, the key advantage of HLA is a set of time management strategies which are really effective for the simulations developed during the product development processes. However, we argue that a HLA-based platform can not meet the requirements identified in the proposed distributed collaborative simulation. First, users of such a platform need to have the knowledge about how HLA works. Second, it requires re-collecting and re-adapting all the simulation models during simulation iterations. Essentially this is due to the inherent problems of the platform; that is, HLA codes and simulation models are highly coupled. Departing from this point, we propose a method to improve the design, by separating simulation codes from HLA codes. The separated simulation codes can be deployed on the Internet and be integrated at run-time. In our reviewed work, it was indicated that Web Services technology is capable of integrating distributed computing resources effectively and efficiently. Therefore, we proposed a solution by developing an integrated framework based on Web Services and HLA.

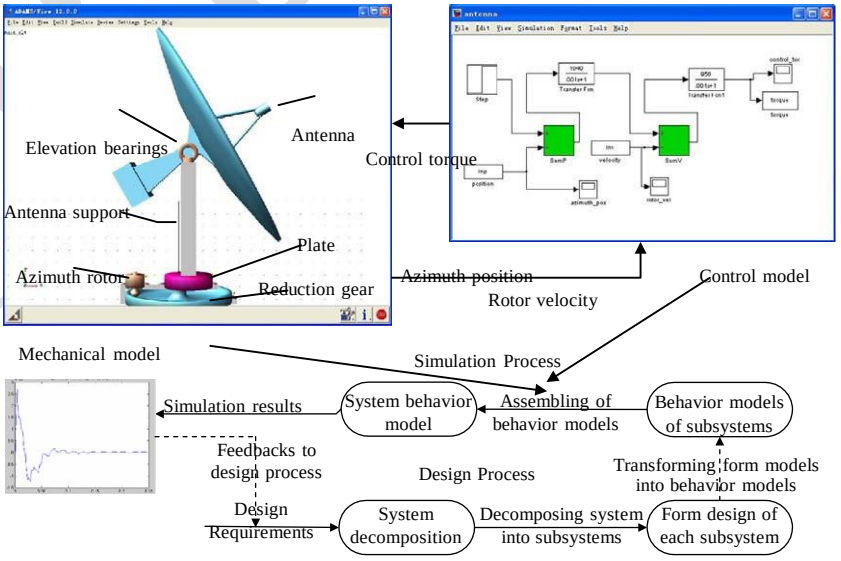

Fig. 1. An example of multidisciplinary simulation and its development process.

3.3. An integrated system framework based on Web Services and HLA

Based on HLA and Web Services, an integrated system framework has been developed, as shown in Fig. 2. There are mainly two parts in this framework, representing HLA and Web Services respectively. The HLA side is identified to manage the advancement of the simulation process and to guarantee accurate interactions. As it imposes lots of burden on network traffic to manage a HLA federation, the required communication is performed in Local Area Network (LAN) by the Run-Time Infrastructure (RTI). HLA agents are the software components developed based on the libraries provided by specific RTI products, which could bridge the HLA side and the Web Service side.

Web Services technology is utilized in this framework to improve the interoperability of the simulation system. Simulation models encapsulated as Web Services can be accessed regardless of their computing platform and their implementation languages. In the Web Services side, the operation of each model during a simulation process is abstracted as a behaviour model that consists of a number of computer routines. These routines serve various purposes, e.g. advancing the simulation process, updating model data, etc. Any inputs from outside a specific service are obtained via the communication model which exchanges data with HLA agents.

Several advantages can be highlighted in the integrated framework. First, simulation models do not need to be re-collect and re-adapted during simulation iterations as only interfaces of the encapsulated services are needed in the HLA codes. Second, the simulation application can be extended to WAN so that simulation models can be deployed on the Internet. Third, the source code of each simulation model can be kept confidential yet integrated at run-time, which is especially suitable for the situation of collaboration where models can not be released. Fourth, designers only need to focus on their part of design tasks as each subsystem is developed separately. More detailed comparison between the proposed integrated framework with other solutions will be given in Section 6.

4. A model-driven approach for the integrated framework

The proposed framework can address the problem of performing simulations in a distributed environment by encapsulating models as Web Services and managing the simulation process within an HLA federation. However, this solution can only resolve the problem at the infrastructure level. Designers without the knowledge of HLA and Web Services will find it difficult to implement the simulation, i.e. a method needs to be developed to interface designers with the infrastructure. In this section, we will discuss the development of a model-driven approach for this purpose.

4.1. Elements involved in a multidisciplinary simulation

The system perspective of a multidisciplinary simulation is shown in Fig. 3, with a number of elements interacting with each other. There are four elements in general, namely simulation models, infrastructure, high-level description, and work of preparation. Simulation models are essentially a depository of computer models which are used to evaluate design concepts from the perspective of a specific discipline. Infrastructure refers to the techniques utilized in the proposed integrated framework, allowing multiple simulation models to run together in a distributed environment. High-level description needs to be specified for each simulation which is performed by separating a system into several subsystems. Specifically, a hierarchy tree indicates how a system is decomposed; and a coupling graph is derived from the interactions between models each of which is developed for a specific subsystem.

''Work of preparation'' defines a series of tasks that should be completed before the simulation can be started. First, designers should work together to identify the requirements of a simulation problem, and construct a high-level representation for it. Then, simulation models of different disciplines need to be created and transformed based on the requirements. At last, software engineers start to develop codes to make the infrastructure work so that a distributed simulation can be started. The relationships between these elements are highlighted in the figure, e.g. simulation models are connected to the Web Services infrastructure as they are encapsulated using Web Services.

Ideally, developers of multidisciplinary simulations should not be concerned with the infrastructure of a simulation environment. In addition, developers should still be able to create or re-use models in the same way when the infrastructure is changed. To fulfil these requirements, an approach needs to be developed to allow designers to work on system models in a platform-independent manner. During the run-time of a simulation, the system models should be mapped to specific codes which can be used to drive the infrastructure. This concept is similar to the Model-Driven Architecture (MDA) and we therefore develop

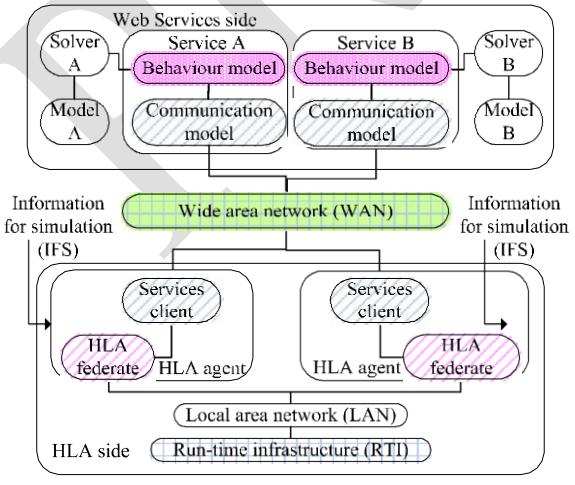

Fig. 2. An integrated framework for multidisciplinary simulation.

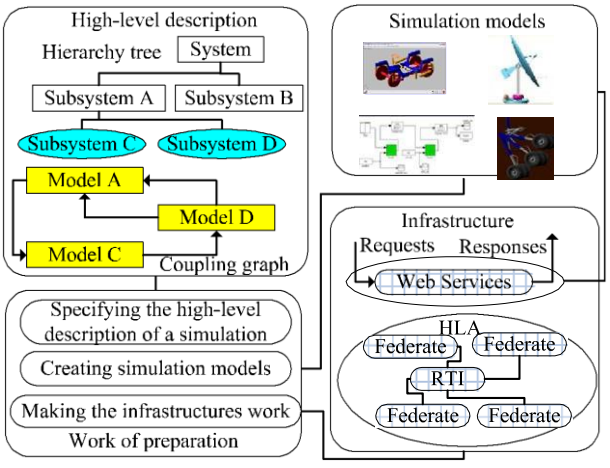

Fig. 3. Elements involved in a multidisciplinary simulation.

a model-driven approach to bridge the high-level model and the infrastructure.

4.2. A model-driven approach

As discussed above, specific information concerning the infrastructure needs to be acquired before a multidisciplinary simulation can be performed. This requires a new mechanism to bridge the high-level modeling and the underlying infrastructure. MDA of the Object Management Group (OMG) is a good framework in terms of separating system model from platform technology [30]. MDA has been applied in industry to address the interoperability between computer systems [31]. In our solution, some concepts from MDA, e.g. the Platform-Independent Model (PIM) and the Platform Specific Model (PSM), are used. However, we do not use the unified modeling language (UML) as our modeling language as we developed a multi-view modeling paradigm to support the collaborative work of designers. Apart from the advantage of better collaboration, we argue that the proposed multi-view modeling paradigm is more straightforward and easy to understand for designers and simulation engineers.

As shown in Fig. 4, the MDA-based approach consists of four parts: user's operations, a multi-view modeling paradigm, model transformation, and infrastructure. Specifically, users' operations involve a set of operations on the high-level model, e.g. models creation, model re-use, model definition and model deployment. The high-level model can be viewed, updated, validated, and shared by users geographically distributed as the representation of the model can be understood by the computers. The system model is further transformed to generate the object models for HLA and Web Services, until all the information for the simulation has been acquired. The model representation is platform-independent, allowing models to be re-used even if the underlying platforms will be changed later (e.g. we use JavaBeans instead of Web Services).

4.3. A multi-view modeling paradigm

A multi-view modeling paradigm is essentially a process of acquiring necessary information for a simulation. This process involves several views each of which represent different stages of the development, as well as different roles of the developers. It aims to supporting the gradual refinement of the system model whilst allowing the users to only focus on a specific view. As shown in Fig. 5, there are four views identified to represent different contents of the high-level model according to the information acquired at different stages:

Decomposition view describes the process of decomposing a system into subsystems. The output of this view is the hierarchy tree of a simulation system.

In realization view, each subsystem needs to be specified about how it is implemented, either by re-using legacy models or by creating a new model.

Composition view represents the aggregation of subsystems where the inputs and outputs of each subsystem are specified. The output of this view is a coupling graph.

Deployment view continues to add information to the system model, describing how each subsystem is deployed as Web Services or HLA federates.

The four views are essentially divided from a holistic process, representing different perspectives of the high-level model. The advantages for such a division are many-fold. First, collaborative

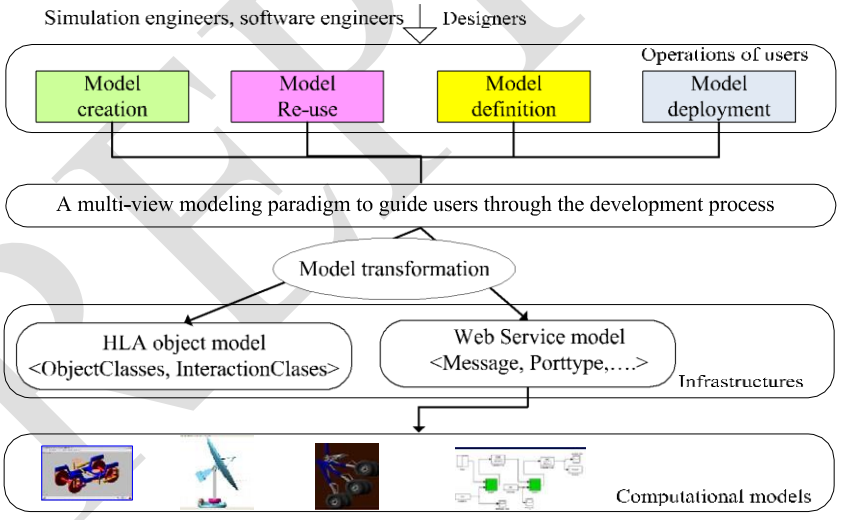

Fig. 4. A model-driven approach for the integrated framework.

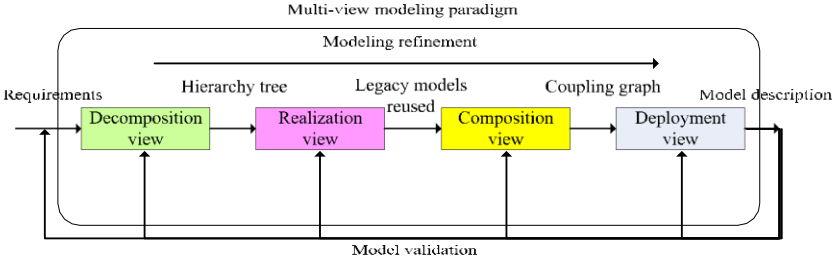

Fig. 5. A multi-view modeling paradigm.

work of users is supported by allowing them to focus only on their own portion of work. Second, model validation can be performed within a specific view so that problems can be easily identified and correlated to a specific view. Last but not least, once the information of a specific view needs to be modified, the information acquired in other views will not be affected.

4.4. MDA/HLA methodology

The proposed MDA-based approach aims to bridge the high-level modeling and the infrastructure by transforming high-level model information into information necessary for the infrastructure. Although some of the transformation work can be done by the computer, a process still needs to be identified to guide the resolving of complex simulations. HLA has a standard development process called the Federation development and execution process (FEDEP) which is not designed to be compatible with MDA or the Model-Driven Integration (MDI). This section presents the outline of a methodology for the development of complex simulation applications where automatic generation can not fulfil the requirements [32–33]. The methodology aims to support the combined use of FEDEP and MDA. It is based on the life cycle which is proposed to standardize the steps to implement simulation from a conceptual model, as depicted in Fig. 6. The need of interoperability between models and simulation tools is also considered in this methodology.

Phase 1: The objectives of the federation need to be defined in the first step. The common goal of all federations created by this methodology is to define a federation of interoperating models. In addition, as described in the second step of FEDEP, a conceptual model is required. In our case, this model contains various domain models represented as entities and actions that represent external information exchanging.

Phase 2: In the second step, the mapping of domain models into HLA federates is realized. In detail, the way models handle received information and how they send information to the federation is addressed, these mechanisms can conform to the synchronization

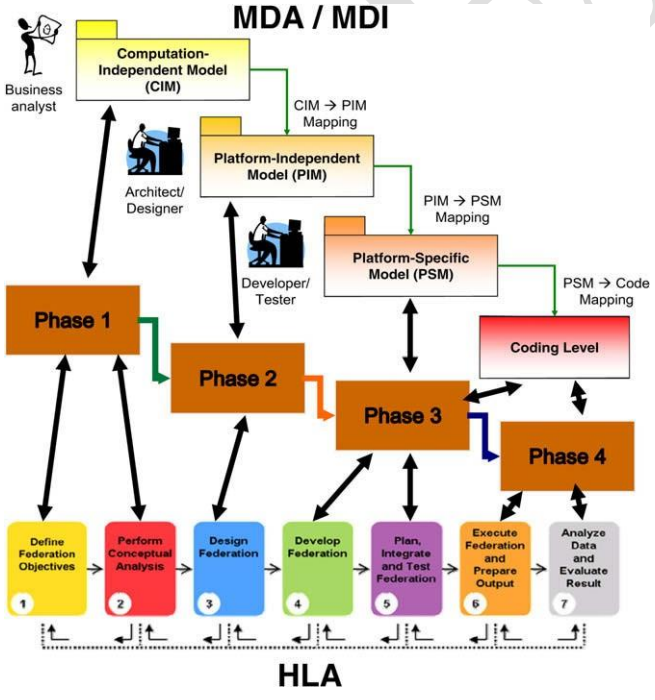

Fig. 6. HLA/MDA integration methodology.

algorithm proposed in [33]. We pay here special attention to the re-use of already existing domain models. In addition, we address in this step what information are to be exchanged, in others terms, what is the structure of the distributed ontology. This level is to consider the problem at the MDA/PIM level.

Phase 3: In the third step, the methodology maps domain interoperating connections between models into HLA interactions and objects. Then, these data are structured to generate the associated FOM. The strategy concerning the confidentiality of data is also explicitly addressed in this step. In addition, to respect time causality, interactions among federates are defined with a 'Time Stamped Order'. They are emitted with a timestamp related to local logical time of the supplier federate so that the Run-Time Infrastructure (RTI) can handle the interactions based on the sequence of the logical time.

Phase 4: The results obtained by simulation are used for the validation of the models by testing and analyzing; in case it does not fulfil the specification, the methodology must allow doing feedback correction as described in the last step of FEDEP.

At the end, the domain model federates generated by this methodology can be re-used and interfaced with heterogeneous HLA-compliant models. For instance, 'client models' federates can be upstream connected to federates and 'subcontractors' federates can be downstream connected.

5. Model representation and transformation

5.1. A model representation schema

Model representation is a technique to store high-level model information, as well as to present this information to both human users and computers in a structured way. In our solution, the model representation schema is implemented using the eXtensible Markup Language (XML). The benefits of using XML have been illustrated in literature, see for example [23–24,27]. The particular advantage of using XML in our solution is that we can transmit the contents of the schema through Web Services, which is especially effective for finding reusable models. A segment is extracted from the model representation schema of the antenna model, as shown in Fig. 7.

This segment describes a simulation which encompasses two models, namely the control model and the dynamics model. The tree structure of a XML document can be used to represent the hierarchy tree of a simulation application. The elements of <inputs> and <outputs> represent the interactions between models, which can be read by computers to construct the coupling graph. The <service> element and <hla> element are highlighted in dashed circles, representing the information for invoking services and initializing HLA federation respectively. Such a schema in our approach is generated automatically by computers once after the multi-view modeling process is completed. Information contained in the representation schema can then be utilized by computers to drive the infrastructure. For instance, the information will be loaded by the transformation engine, a routine which implements the model transformation mechanisms for HLA and Web Services.

5.2. Model transformation mechanisms for HLA and Web Services

A methodology is introduced above to bridge HLA and MDA so that a standard process of developing complex simulations using our approach can be identified. In our approach, computers can help to generate the object models useful to the infrastructure. For instance, the FOM and SOM information required by HLA can be generated based on the multi-view model. FOM represents a

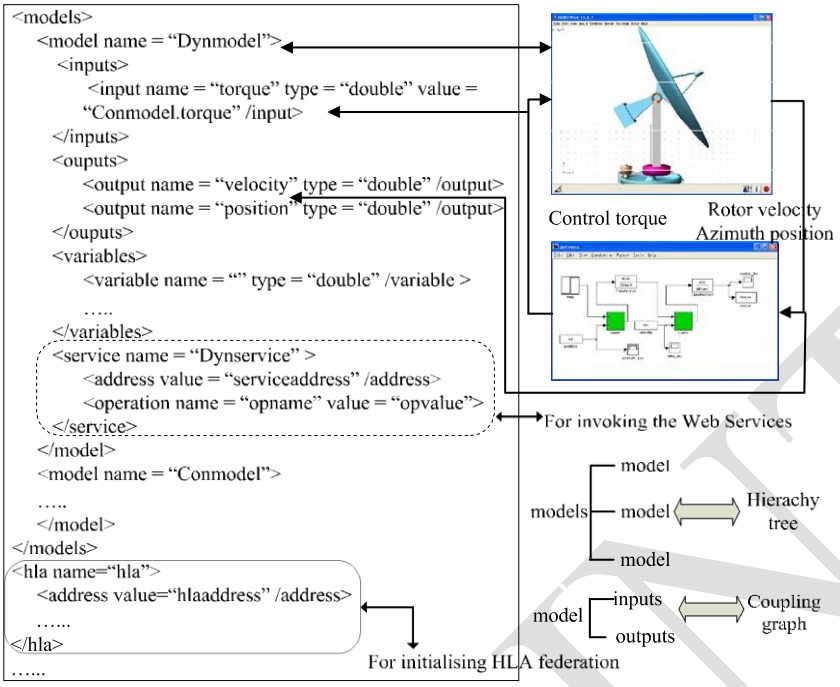

Fig. 7. A segment of the model representation schema.

set of objects (including ObjectClass and InteractionClass) that can be shared within a federation while SOM describes the ability of a federate to share data with counterparts. Specifically, the SOM of a federate encompasses the objects that it subscribes from others, as well as the objects it publishes to be used by others. Furthermore, the deployment of Web Services requires the information about how a service interacts with its client, as well as what data need to be transferred. At the run-time of a simulation, a mapping and transferring of information will be performed between the representation schema and object models of the infrastructure, as shown in Fig. 8.

The representation schema of the antenna simulation is shown on the left side of the figure while elements required by, and identified in the integrated framework are shown on the right. The dashed line with arrow means that an element is transformed into another element. The solid line with arrow indicates information flow between two elements. As shown in the figure, the elements of <outputs>, <inputs>, <variables>, <services> in the representation schema can be used to generate the necessary object models for HLA and Web Services. The representation schema is obtained from a high-level modeling process, the generation of object models for HLA and Web Services is then driven by the high-level mod-

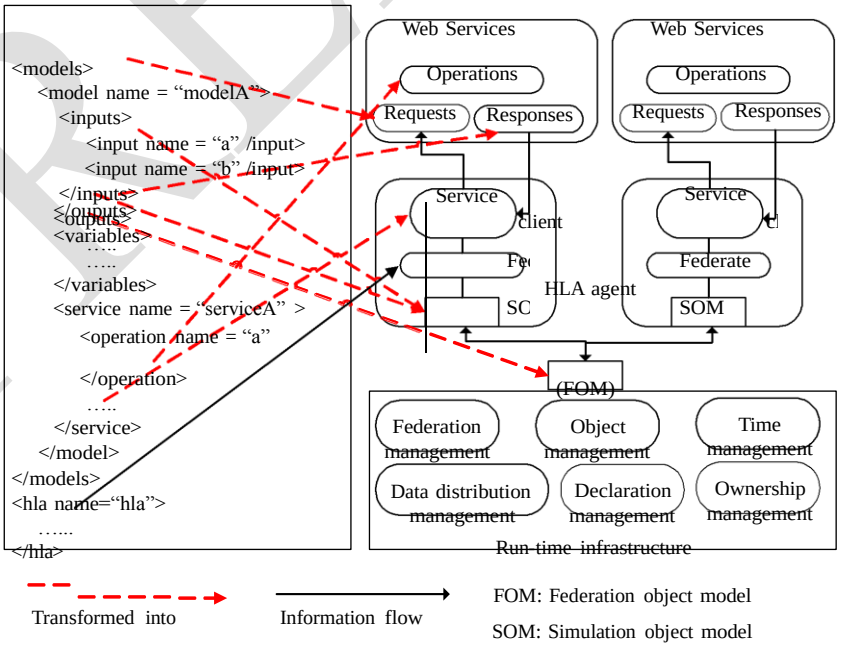

Fig. 8. Model transformation mechanisms for HLA and Web Services.

els. In this way, the advantages of HLA and Web Services are retained whilst designers only need to focus on the high-level models of a simulation.

5.3. Code generation and system deployment

Development of the mechanisms mentioned above aims to make computers to complete as much work as possible during the preparation of a simulation. However, the automatic generation of object models can only provide the static information required to start a simulation. In our research, we find that the further generation of codes can reduce the work load of developers. For instance, each HLA federate has the same behaviour during the running of a simulation. We therefore develop a generic federate class which can be used as a template to generate customized individual federates. Specifically, the customization involves specifying a set of parameters regarding objects and interactions exchanged within a HLA federation, local simulation time and advancement step, as well as the commands to start or stop simulations. It's simpler to generate codes for Web Services as such codes mainly deal with run-time interactions. In our implementation, a standard Java class is generated with a set of functions to initialize a model, advance the simulation, exchange simulation data, etc. Web Services created with other techniques can also be developed by following this method. For those tasks which can not be performed by the generated codes, software engineers can make further development as the supplement of these codes. In this way, designers only need to develop codes to control a simulation tool and obtain simulation data.

6. Evaluation of the solution

To evaluate our approach, a comparison was made between different methods for constructing distributed simulations. A prototype was developed based on the model-driven approach proposed above, leading users through the solution of a simulation problem. To test the accuracy of a simulation, the antenna simulation described above was run using the prototype, and the results obtained were compared with those acquired by performing the same simulation based on interfaces between simulation tools. A discussion is given based on the comparison and the case study.

6.1. A comparison of different approaches

Although collaborative design has been studied for a long time, there is only a little research on performing simulations in a distributed and collaborative manner. Our motivation is to develop an approach to perform multidisciplinary simulation by integrating a number of distributed simulation models, as well as to provide a virtual environment where users only need to focus on a specific aspect of the whole problem. We therefore made a comparison between previous approaches for distributed collaborative simulation to evaluate how the proposed approach can fulfil the requirements, as well as to highlight key factors influencing the development of such a simulation environment.

The criteria we selected for the comparison are interoperability, efficiency of simulation development, types and scales of simulation supported, degree of encapsulation, and the complexity of developing such an environment. Approaches compared are those reviewed in the related works section. The comparison is shown in Table 1. ''Interoperability'' defines the degree to which individual models can interact with each other. The approach based on interfaces between simulation tools is not good regarding this criterion as interoperation can only take place between models created with tools that have interfaces. HLA-based simulation only allows federates in a LAN to interact with each other, meaning that its interoperability is constrained to a single LAN.

Solving an engineering problem generally involves many iterations where parameters of specific models in the simulation need to be updated. Therefore, ''efficiency of simulation development'' refers to how easily a simulation can be created or updated. An interface-based approach requires a copy of every model to be manually to be collected into a single computer for each iteration, so its efficiency is low. Approaches with a medium efficiency require the codes controlling the simulation to be rewritten if models are updated, e.g. HLA-based approach, VR-based approach, and the integrated approach. Web Service-based approach is efficient as it requires only the services to be changed. The model-driven approach addresses this problem by using a high-level model which is independent of the infrastructure, which is therefore very efficient.

''Types'' and ''scales of simulation supported'' evaluate how an approach supports various simulation applications. Approaches using HLA support a variety of types of simulation, e.g. continuous time, discrete time, human-in-the-loop simulation, etc. The VR-based and Web Services based approaches can only deal with some kinds of simulations. Regarding the scales of simulation, approaches based on several models can share the computational burden between multiple computers on the network. However, as a simulation scales up, the simulation modelling overhead will definitely increase; so we argue that among the approaches in the comparison, only those using HLA can hopefully have better performance.

In our opinion, collaborative development depends strongly on two factors: (1) the effectiveness of the communication between developers; and (2) the degree to which a single developer can focus on a specific part of the whole design, without being concerned with others' work. ''degree of encapsulation'' corresponds to the second aspect, since we believe that an effective division of tasks can improve collaboration. An interface-based approach requires designers to know how other models are developed, and therefore imposes many loads on developers. In HLA-based and VR-based approaches, models are coupled with infrastructure, so more effort is needed to change a model. The model-driven and Web Service-based approaches have a clear division of tasks, helping designers to perform collaborative work.

The ''complexity of development'' refers to the difficulty of implementing an approach. In our opinion, the interface-based ap-

Table 1
A comparison of different approaches.

Approaches	Interoperability	Efficiency of simulation development	Types of simulation supported	Scales of simulation supported	Degree of encapsulation	Complexity of development
Interface-based simulation [9]	Restricted to some tools	Low	Restricted to some tools	Medium	High	Low
HLA-based approach [19]	Local area network	Medium	Many	Large	Medium	High
VR-based approach [15–16]	Cross platforms	Medium	Some	Medium	Medium	High
Approach using Web Services [23–24]	Cross platforms	High	Some	Medium	Low	Medium
Integrated approach [22]	Cross platforms	Medium	Many	Large	Medium	High
Model-driven approach (this paper)	Cross platforms	High	Many	Large	Low	High

proach requires no infrastructure to be constructed, and is therefore the least complicated. On the other hand, approaches involving HLA or VR are complex in general as significant infrastructure must be put in place. A Web Services based approach is of moderate complexity as only few codes for managing a simulation need to be developed.

Besides the criteria used in the comparison, other criteria, e.g. accuracy of simulation and loads imposed on the network connection, can also be applied to evaluate distributed simulation approaches. As the interface-based approach makes use of interfaces provided by vendors, we assume it has the best accuracy. In the following sections, the accuracy of the model-driven approach will be compared with the interface-based approach, and a comprehensive analysis will be given.

6.2. Prototype implementation

A software prototype was developed to test the proposed model-driven approach. It is designed to support designers through a simulation process, from the high-level modeling to the execution of a simulation. Some objectives are identified during the development of this prototype. First, it should support multiple users for creating a high-level description of a simulation. Second, it should transform a representation obtained from the multi-view modeling process into object models for HLA and Web Services, and should also generate necessary codes for them. Third, it should provide interfaces for users to control a simulation process. The prototype is implemented as a Web-based application which can support geographically distributed users. It is implemented in Java, being integrated with the middleware to implement HLA and Web Services. The prototype can be deployed on any computing platform. A set of Graphical User Interfaces (GUIs) were developed to lead designers through the development process which consists of three stages. In this section, screenshots are taken from the case study of the antenna simulation.

In Stage 1, a high-level model of the simulation needs to be created. Each member of the development team logs on to the prototype and completes a multi-view modeling process, as shown in Fig. 9. The system is first decomposed into two subsystems in the decomposition view by the simulation engineers. Then, two designers work in the realization view to specify how each subsystem should be implemented. The prototype software allows designers to re-use models from a set of existing models. In our case, the existing fuzzy control model is not suitable for the antenna simulation, so the two subsystems will be implemented by creating two new models. After that, the two designers work in the composition view, specifying the interactions between the two subsystems. Finally, a software engineer starts to work in the deployment view, defining how each subsystem model should be deployed. The prototype tool provides an interface for sending messages to team members to facilitate their communication during the development process.

In Stage 2, preparation work for the simulation should be completed. First, the high-level model of the antenna simulation obtained in Stage 1 is transferred in the format of the representation schema described in Section 5.1. A transformation engine then transforms the representation schema into object models for HLA and Web Services as shown in Fig. 10, and also generates necessary codes. After that, the software engineer needs to complete all the tasks to make the infrastructure work, based on the models and

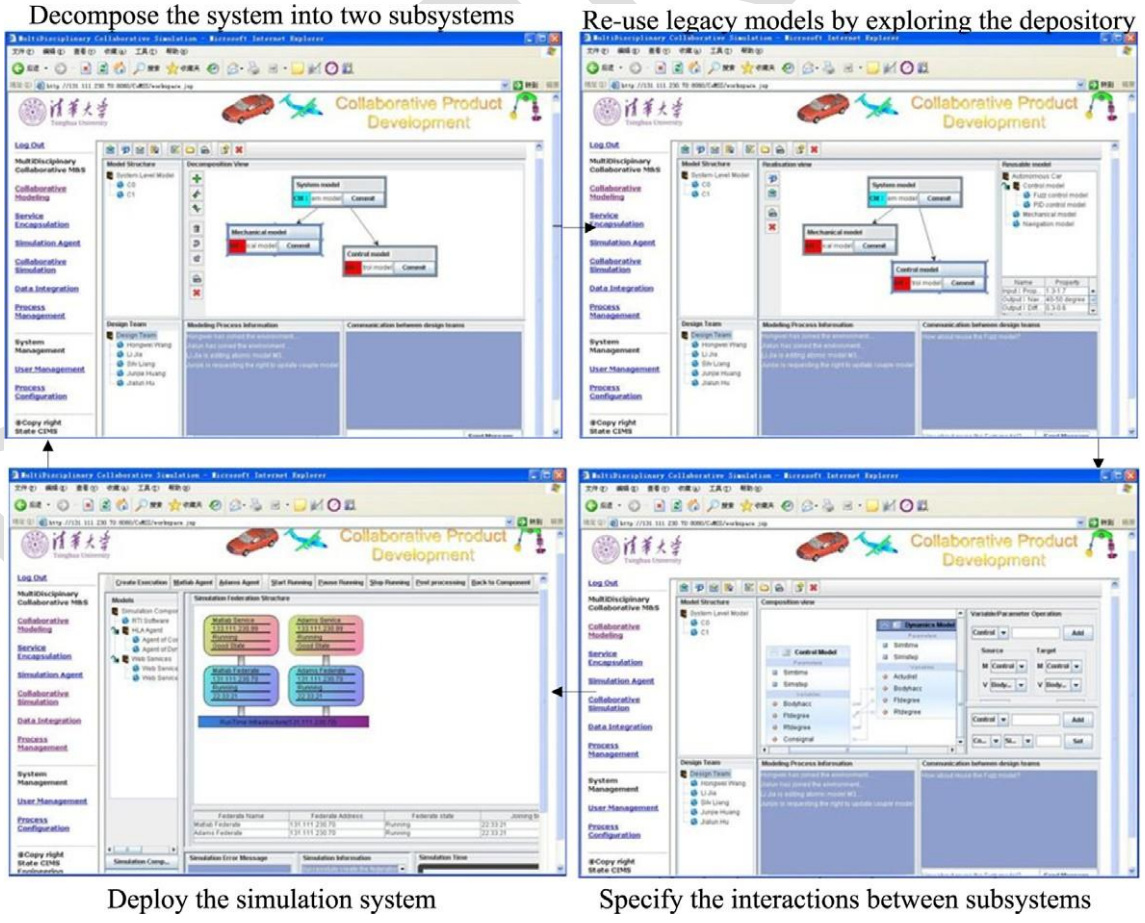

Fig. 9. Collaborative modeling based on the prototype.

codes generated by the tool. Finally, since the two models are not in the repository, the software engineers should work together with the designers to encapsulate the developed simulation models as Web Services.

In Stage 3, designers and simulation engineers execute the simulation and analyze the results obtained from the simulation. A multidisciplinary simulation can be started from the GUI when the necessary information for infrastructure is acquired in Stage 2. Simulation engineers can monitor a simulation run by obtaining run-time messages from the tool. During each design iteration, the parameters of the simulation can be modified, allowing a wide range of design concepts to be evaluated. The results obtained during each simulation can be shown. For instance, the run-time data of the azimuth position and the control torque in the antenna simulation are shown in Fig. 11. Developers are able to query the value of a specific parameter at any time in the simulation process.

6.3. Running the antenna simulation

As mentioned above, the accuracy of simulations is also an important criterion for evaluating the performance of a multidisciplinary simulation system. We believe that results obtained using the interface-based approach to be the most accurate because the interfaces have been validated by the vendors before being released. Therefore, the antenna simulation is run in our prototype tool and compared with the results obtained using the interface-based approach to evaluate the accuracy of our approach as well as demonstrating its functionality. As shown in Table 2, we set the simulation steps of the two models as 0.0125 s and 0.0294 s respectively. This is a deliberate setting because we want to test the simulation in an extreme case when the two steps are completely not in match. A quadratic interpolation algorithm is used to find the results at the end of federation simulation step.

The simulation is developed using the prototype, and the proposed process is followed by the designers. This example is quite simple and the simulation can be constructed very quickly. Once the high-level modeling and model encapsulation are completed, simulation engineers can start and monitor a simulation process. They can start, pause and stop a simulation at any time. As evidenced in our experiment, the simulation can be performed as if all the models are running in a single computer, although they are actually distributed on the network.

Simulation engineers can easily find problems with the design since they can check the data of any parameter at any time of a simulation. They can modify parameter values until the best trade-off is applied. In our experiment, we did not perform this because the models of the antenna simulation have been verified and our objective is to test the accuracy of our approach. In total there are three design variables in the antenna simulation, namely control torque, rotor velocity, and azimuth position. The simulation results for all of them are shown in Figs. 12–14. The curves in dashed lines are obtained from our prototype tool while those in solid lines are obtained from simulations based on the interfaces between MSC.ADAMS and Matlab. As shown in the figures, curves obtained in both cases follow a similar trajectory, although small differences appear at some points in the simulation process. In both cases, the azimuth position of the antenna converges to three degrees. Therefore the model-driven approach has a comparable simulation accuracy to the interface-based approach for the antenna simulation.

6.4. Discussion

As evidenced in the prototype implementation and the case study, the proposed integrated framework and the model-driven approach can be used to perform multidisciplinary simulations in a distributed environment. The advantages of the approach have

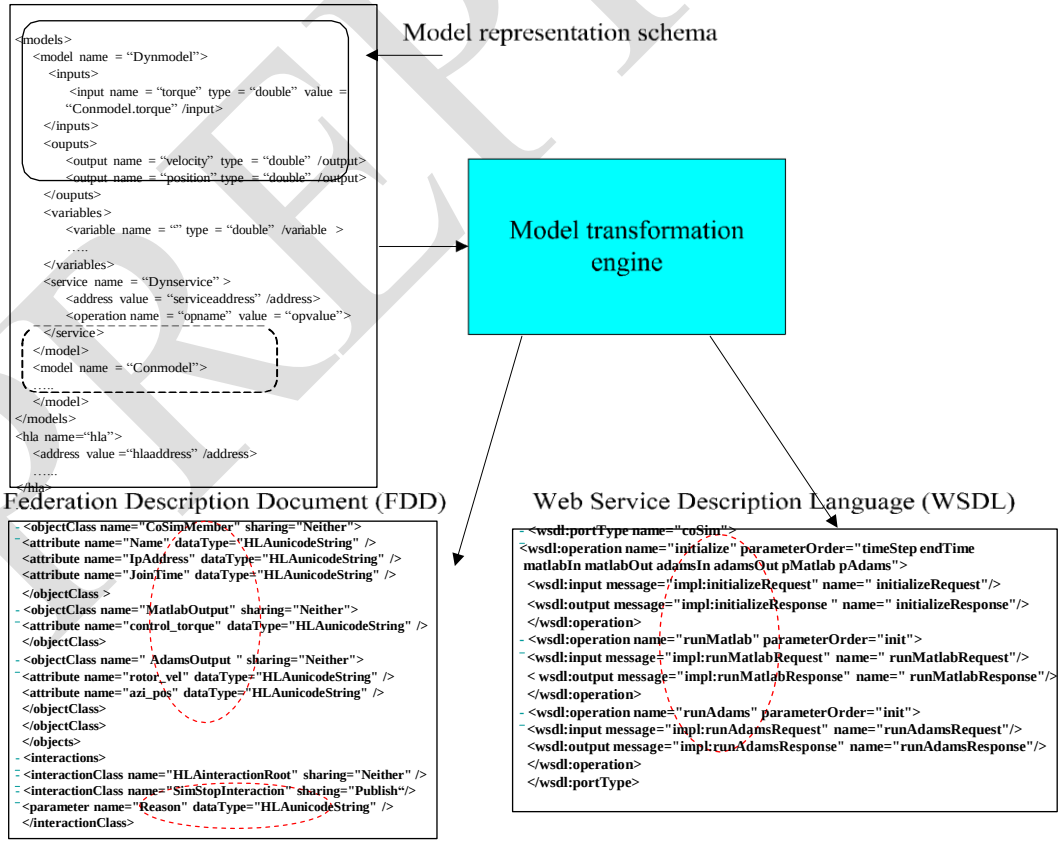

Fig. 10. Object models of Web Services and HLA generated by the prototype system.

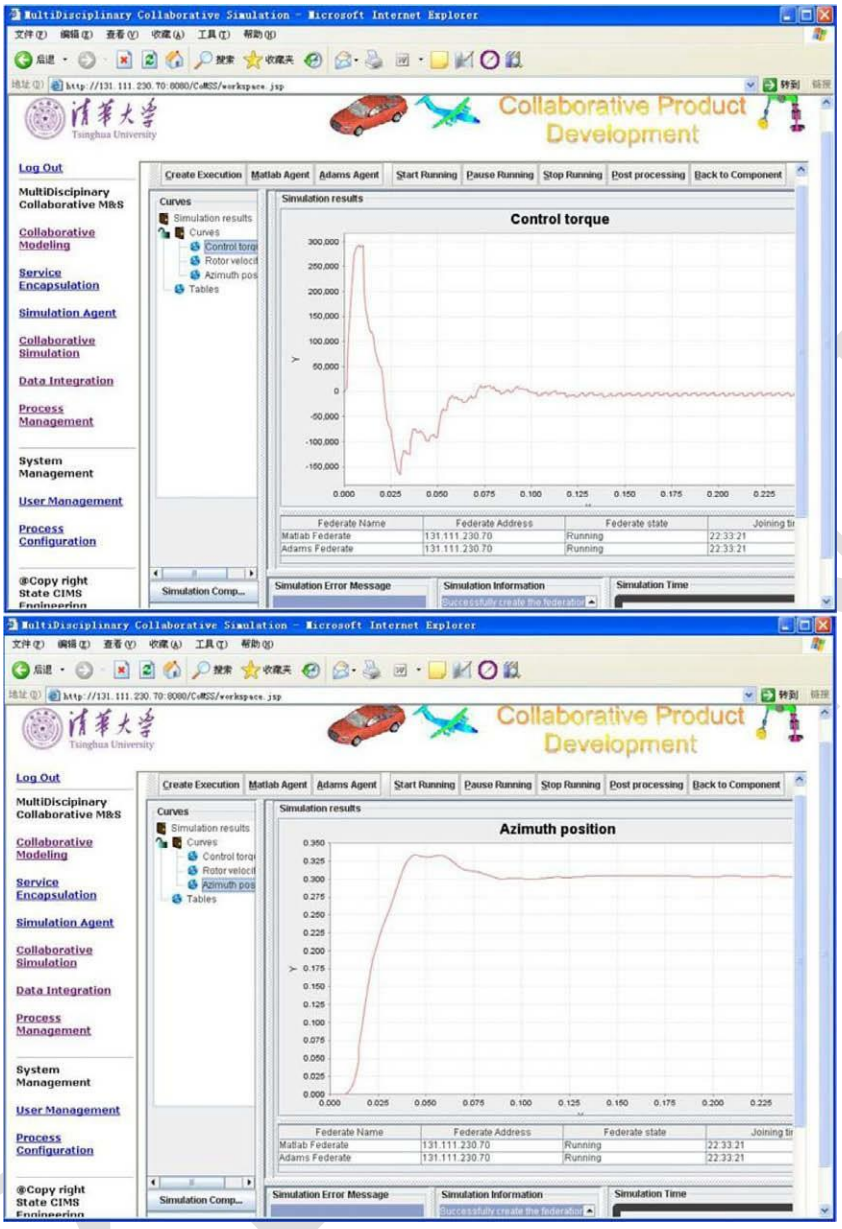

Fig. 11. Simulation results shown in the GUI of the prototype.

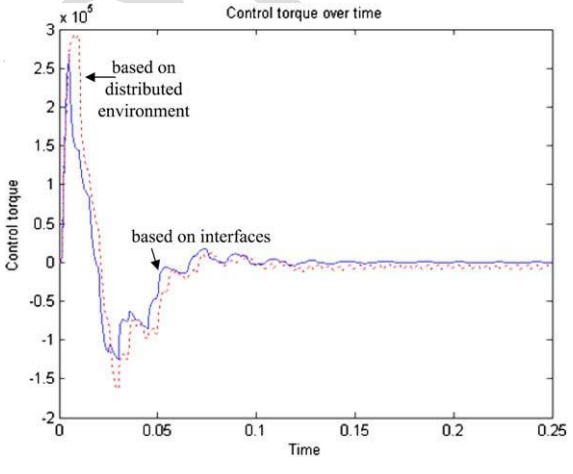

Fig. 12. Control torque over the simulation time.

been discussed in the comparison. In our opinion, there is scope for further discussion of the usability of the approach. First, a model-centred view of a simulation application can improve the management and re-use of simulation models, and is useful to the system implementation. Second, encapsulating models as Web Services enable them to be kept confidential yet integrated during runtime. It is particularly useful when models cannot be shared be-

Table 2
Parameters of the distributed simulation.

Parameter name	Parameter value
Total simulation steps	50
Simulation time	0s, 0.25s
Step size of the federation	0.05s
Step size of dynamics model	0.0125s
Step size of control model	0.0294s
Algorithm for interpolation and extrapolation	Quadratic interpolation algorithm

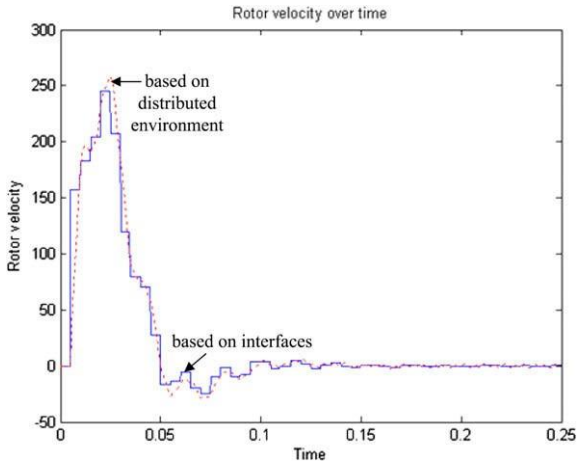

Fig. 13. Rotor velocity over the simulation time.

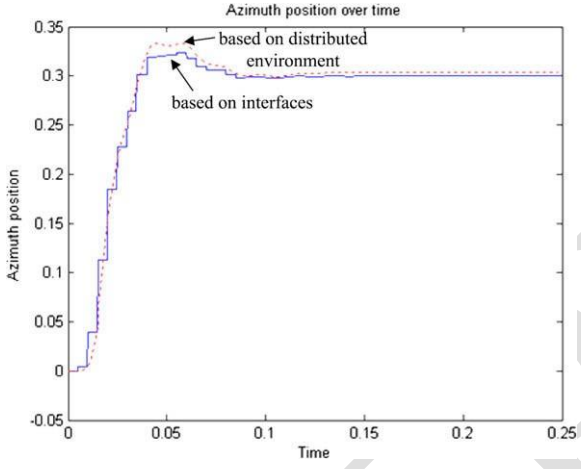

Fig. 14. Azimuth position over the simulation time.

tween participants involved in a collaborative development project. Third, the network load for performing simulations is not significant, since only simulation results are transferred over the Internet while simulation management (within a HLA federation) can be performed in LAN or on a server. However, the system is not suitable for some applications. For instance, simulations which have strict requirements on the run-time, e.g. real-time simulations, cannot be performed in our prototype tool. In addition, problems whose sub-problems are highly coupled during numerical calculation may not be solvable in our prototype as our approach is based on multiple solves, and may not be accurate enough for that situation.

7. Conclusion

Collaborative product development is being widely studied in both academia and industry. However, most research focuses on collaborative work for creating designs while little attention has been given to collaborative work for evaluating designs. Our research is motivated by this situation, aiming to develop an approach to support simulations performed by multidisciplinary teams in a distributed environment.

Specifically, we propose an integrated infrastructure based on HLA and Web Services to connect distributed simulations. However, it is difficult for designers to acquire the knowledge to use these technologies although applying them for distributed simulation has been justified in previous research. To resolve this problem, a model-driven approach is developed to enable designers to focus on the high-level structure of a design. The high-level model is independent of any technology, and is therefore stable enough to be stored and re-used even if the infrastructure is changed. Our approach is evaluated by comparing it with existing approaches and running an engineering simulation. As evidenced in our experiment, the proposed framework and approach are viable for developing multidisciplinary simulations. In addition, this approach has certain advantages compared with existing ones. Such a solution could be adapted to any other engineering system which aims to manage and integrate distributed computing resources.

In our future work, we will explore the factors influencing the accuracy of a multidisciplinary simulation, and study how to improve the convergence of simulation problems. In addition, we will develop more functionality for the prototype software tool, to allow more complex engineering simulations to be performed.

Acknowledgements

This paper is supported by the National Natural Science Foundation of China (Grant No.60674079) and the Key Laboratory of Beijing Simulation Centre (Grant No. B0420060524). The authors are grateful to the anonymous reviewers for their valuable suggestions and to David Wyatt for proofreading this paper.

References

[1] A. Diaz-Calderon, A composable simulation environment to support the design of mechatronic systems, Ph.D. thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 2000.
[2] M.S. Shephard, M.W. Beall, R.M. O'Bara, B.E. Webster, Toward simulation-based design, Finite Elements in Analysis and Design 40 (12) (2004) 1575–1598.
[3] R. Sinha, V. Liang, C.J.J. Paredis, P.K. Khosla, Modeling and simulation methods for design of engineering systems, Journal of Computing and Information Science in Engineering 1 (1) (2001) 84–91.
[4] G. Ferretti, G. Magnani, P. Rocco, Virtual prototyping of mechatronic systems, Annual Reviews in Control 28 (2) (2004) 193–206.
[5] L.U. Gokdere, K. Benlyazid, R.A. Dougal, E. Santi, C.W. Brice, A virtual prototype for a hybrid electric vehicle, Mechatronics 12 (4) (2002) 575–593.
[6] J.V. Amerongen, P. Breedveld, Modelling of physical systems for the design and control of mechatronic systems, Annual Reviews in Control 27 (1) (2003) 87–117.
[7] A. Rukgauer, W. Schiehlen, Simulation of modular mechatronic systems with application to vehicle dynamics, Acta Mechanica 125 (1–4) (1997) 183–196.
[8] W. Marquis-Favre, E. Bideaux, S. Scavarda, A planar mechanical library in the AMESim simulation software. Part II:Library composition and illustrative example, Simulation Modelling Practice and Theory 14 (2) (2006) 95–111.
[9] T. Makkonen, K. Nevala, R. Heikkilä, A 3D model based control of an excavator, Automation in Construction 15 (5) (2006) 571–577.
[10] I. Nakhimovski, Contributions to the modeling and simulation of mechanical systems with detailed contact analyses, Ph.D. thesis, Department of Computer and Information Science, Linköpings University, 2006.
[11] M.M. Da Silva, O. Bruls, B. Paijmans, et al., Concurrent simulation of mechatronic systems with variable mechanical configuration, in: Proceedings of the ISMA 2006, 2006, pp.69–79.
[12] N. Bakis, G. Aouad, M. Kagioglou, Towards distributed product data sharing environments-progress so far and future challenges, Automation in Construction 16 (5) (2007) 586–595.
[13] W.D. Li, A web-based service for distributed process planning optimization, Computers in Industry 56 (3) (2005) 272–288.
[14] S. Szykman, R.D. Sariram, Design and implementation of the Web-enabled NIST design repository, ACM Transactions on Internet Technology 6 (1) (2006) 85–116.
[15] Q. Shen, J. Gausemeier, J. Bauch, R. Radkowski, A cooperative virtual prototyping system for mechatronic solution elements based assembly, Advanced Engineering Informatics 19 (2) (2005) 169–177.
[16] Q. Shen, M. Grafe, To support multidisciplinary communication in VR-based virtual prototyping of mechatronic systems, Advanced Engineering Informatics 21 (2) (2007) 201–209.
[17] F. Pahng, N. Senin, D. Wallace, Distribution modeling and evaluation of product design problems, Computer-Aided Design 30 (6) (1998) 411–423.
[18] Y.D. Wang, W. Shen, H. Ghenniwa, WebBlow:a web/agent-based multidisciplinary design optimization environment, Computer in Industry 52 (1) (2003) 17–28.

[19] J. Jian, H. Zhang, B. Guo, et al., HLA-based collaborative simulation platform for complex product design, in: Proceedings of the 8th CSCWD International Conference, IEEE Computer Society, Xiamen, China, 2004, pp. 462–466.
[20] IEEE Std1516-2000. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Framework and Rules, IEEE, 2000.
[21] IEEE Std1516.1-2000. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Federate Interface Specification, IEEE, 2000.
[22] H. Wang, H. Zhang, An integrated and collaborative approach for complex product development in distributed heterogeneous environment, International Journal of Production Research 46 (9) (2008) 2345–2361.
[23] E.G. Roselló, M.J. Lado, J.G. Dacosta, M.P. Cota, A component framework for reusing a proprietary computer-aided engineering environment, Advances in Engineering Software 38 (4) (2007) 256–266.
[24] B. Johansson, Model management for computational system design, Ph.D. thesis, Department of Mechanical Engineering, Linköpings University, 2003.
[25] L. Schubert, A. Kipp, B. Koller, Supporting collaborative engineering using an intelligent web service middleware, Advanced Engineering Informatics 22 (4) (2008) 431–437.
[26] B. Dong, G. Qi, X. Gu, X. Wei, Web service-oriented manufacturing resource applications for networked product development, Advanced Engineering Informatics 22 (3) (2008) 282–295.
[27] N. Bakis, G. Aouad, M. Kagioglou,Towards distributed product data sharing environments-Progress so far and future challenges, Automation in Construction 16 (5) (2007) 586–595.
[28] Matlab, the Mathworks. Available from: <http://www.mathworks.com/>.
[29] MSC.ADAMS, the MSC software. Available from: <http://www.mscsoftware.com/>.
[30] Model driven architecture, the Object Management Group. Available from: <http://www.omg.org/mda/>.
[31] R. Jardim-Goncalves, A. Grilo, A. Steiger-Garcao, Challenging the interoperability between computers in industry with MDA and SOA, Computers in Industry 57 (8–9) (2006) 679–689.
[32] G. Zacharewicz, D. Chen, B. Vallespir, HLA supported, federation oriented enterprise interoperability, in: Proceedings of the MOSIM 2008, 2008, Paris, France.
[33] G. Zacharewicz, N. Giambiasi, C. Frydman, Improving the lookahead computation in G-DEVS/HLA environment, in: Proceedings of the DS-RT 2005, 2005, pp. 273–282.