Advanced Engineering Informatics 24 (2010) 208–218

Contents lists available at ScienceDirect

Advanced Engineering Informatics

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

Towards a collaborative modeling and simulation platform on the Internet

Hongwei Wang [a,*], Aylmer Johnson [a], Heming Zhang [b], Silv Liang [b]

[a] Engineering Design Centre, Cambridge University Engineering Department, Cambridge CB2 1PZ, UK
[b] State CIMS Engineering Research Centre, Tsinghua University, Beijing 100084, PR China

ARTICLE INFO

Article history:
Received 26 April 2008
Received in revised form 7 November 2009
Accepted 12 November 2009
Available online 21 December 2009

ABSTRACT

The design and development of complex engineering systems, e.g. mechatronics, is increasingly characterized as an interdisciplinary process, which entails the co-operative work of distributed teams. However, traditional simulation tools generally emphasize discipline-oriented purposes and centralized running, and cannot adequately support the collaborative simulation in a distributed environment. Although some tools have been developed to enable remote access to simulations, most of them focus on the sharing of information and the provision of simulations via a single and centralized program. The long-term aim of this research is to develop a collaborative modeling and simulation platform, which will provide an integrated environment for multi-disciplinary teams to create, share, and integrate simulation services on the Internet. A step towards such a platform is described which includes the infrastructures for distributed communication, the mechanism for run-time interaction, and the representation of a simulation system at the abstract level. In this paper, we will (1) formulate a collaborative simulation problem; (2) identify the factors influencing the performance of a simulation; (3) propose a method for run-time interaction to achieve optimal simulation accuracy; and (4) illustrate how the components serving different purposes are integrated. A case study is described, based on a prototype implementation of the proposed solution, to explore supporting collaborative simulation in an Internet-distributed environment.

1. Introduction

Numerical techniques and computational simulations are widely applied in the product development domain [1]. Over the past three decades, scientists and engineers have focused on developing simulation tools to effectively solve individual problems, as evidenced in many current commercial software products. However, the very nature of current simulation tools emphasizes a single computer and specific purposes [2]. For simulation analyses of complex systems such as mechatronic applications, a single simulation tool is often not adequate to solve the problem. This problem can be solved in two ways, in general. The first way emphasizes solving simulation problems in a single tool by developing multi-disciplinary model blocks and adding them to the library incrementally. The second way, on the other hand, is centred on the utilization of multiple tools which are integrated at the run-time of a simulation. The pitfalls of these two ways are obvious: (1) the first requires more functionality in a simulation tool and more training for its users; (2) the second requires the efficient and effective integration of multiple tools at run-time, to guarantee the accuracy of simulation.

Currently, the vendors of commercial software products are focusing on the first way by developing more blocks and improving the solvers. However, we believe that the second way is very promising and that developing a software platform to support the integration of multiple models is therefore necessary. Firstly, this allows engineers to use the tools they are used to. Secondly, it will not require the continuous updating of simulation tools. Most importantly, it enables legacy simulations to be re-used, be they developed using commercial software products or in-house codes. Apart from supporting various simulation problems, such a platform should also take into account the shift of paradigm in product development, i.e., to develop distributed and collaborative systems [3]. There are several prototype systems developed to serve such a purpose, as found in literature [2,4–10]. The deficiencies of these systems are (1) they entail lots of development work and (2) they focus mainly on the implementation of distributed communication and the sharing of information, while overlooking the need for efficient integration at run-time. Departing from this point, we propose to develop an integrated environment for multi-disciplinary teams to create, share, and integrate simulation services on the Internet by leveraging the recently-developed information technology.

* Corresponding author. Present address: Engineering Design Centre, Cambridge University Engineering Department, Trumpington Street, Cambridge CB2 1PZ, UK. Tel.: +44 1223 748535; fax: +44 1223 339883.
E-mail address: hw308@cam.ac.uk (H. Wang).

1474-0346/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.aei.2009.11.003

The rest of this paper is organized as follows. In Section 2, we review relevant research work. In Section 3, we describe the formulation of collaborative simulation and indicate the requirements from a technical point of view for a collaborative simulation platform. In Section 4, factors influencing the performance of collaborative simulation are identified and our solution to improve the performance is discussed. In Section 5, we introduce the concepts of our proposed solution and the corresponding approaches to address the key issues of collaborative simulation. In Section 6, the implementation of a prototype system is illustrated, and the development of a ball-beam simulation is studied on the prototype system. Finally the conclusions are given in Section 7.

2. Literature review

A number of issues need to be addressed for future simulation systems supporting engineering design processes, namely the modeling paradigm, the integration with design tools, and the collaborative work supported [1]. Two further issues, identified as requirements for the modeling language and simulation environments for mechatronic products [11], also need to be taken into account: (1) efficient and effective numerical simulation and (2) modular modeling and software re-use. The platform proposed in this research, therefore, needs to deal with the high-level representation of a simulation, effective run-time interaction, the communication between distributed models, and the encapsulation of models. Literature on these topics is reviewed as follows.

2.1. Collaborative and distributed product development

With the development of information technology, both in hardware and software, the way of conducting engineering tasks is changing from centralized development to distributed collaboration. The paradigm of collaborative engineering is hereby proposed by researchers and practitioners to support both the communication among, and the provision of information for engineers. For instance, the SHARE framework [12] and the Madefast program [13] can be viewed as early work to support collaborative product development by making information available and accessible on the Internet. Both of these projects focused on the gathering, organizing and exchanging of information. Meanwhile, virtual reality techniques were also used together with Web technology to make virtual prototypes available on the Web, as well as to improve human–computer interaction [14].

With large amounts of information being recorded in previous designs, design repositories have been proposed to store and retrieve design information based on a representation model created to organize the information. For instance, Szykman and Sriram proposed to implement a Web-enabled NIST design repository [3]; Brown et al. developed a Web-enabled virtual repository for supporting distributed automotive component development [15]. The sharing and integrating of computational resources becomes possible as distributed computing technology goes beyond networked connections. It is, therefore possible to perform distributed calculation for optimizing designs, which boosts the development of Web-based optimization systems [16,17]. In addition, information technology has also been used to support the collaborative work of engineers in a virtual organization [18].

2.2. Collaborative simulation and its modeling paradigm

Collaborative simulation emphasizes simulating an engineering problem with a number of models created using different tools and integrated at run-time. Two issues therefore need to be addressed for collaborative simulation, namely the modeling of an engineering problem, and accurate interaction at run-time. In terms of the modeling paradigm, a modular approach is advocated to achieve a flexible structure and the re-use of components. Breedveld pointed out that common simulation software based on block diagrams or equation inputs do not sufficiently support these features, so used a port-based modeling paradigm instead [19]. Giese et al. proposed a rigorous component concept based on the notion of UML component diagrams to enable modular design and verification of component-based mechatronic systems [20]. Sodja and Zupančič studied and compared two approaches of building a passive-solar-building simulator by using an object-oriented modeling environment and Matlab-Simulink, respectively, finding that the former approach could achieve high-level modularity and easier extension [21].

Besides a modeling paradigm, the efficient exchange of data between different tools is also of significant importance to the accuracy of collaborative simulation [11]. Larsson et al. used the method of characteristics to manage the communication between numerical solvers [22]. Rukgauer and Schiehlen studied how modular components of a simulation could be encapsulated by an input–output representation and proposed a multirate integration to support efficient simulation [23]. In terms of how collaborative simulation can be implemented, there are two main methods, namely: interface-based method and wrapper-based method. The former relies on interfaces between simulation tools, encapsulating the behaviour of communication with other tools as a block of a tool. Larsson et al. implemented the collaborative simulation using Matlab-Simulink [24], MSC.Adams [25], and Hopsan [26] based on this method [22]. However, this method is only suitable for problems where the tools used have mutual interfaces and it is still necessary to collect all the software and simulation models onto a single computer. A more general solution is the wrapper-based method, which wraps models separately and integrates them when necessary. Many simulation systems mentioned above used this method [4–10] and the in-house codes and routines, e.g. Promech [27], can be wrapped and made accessible as services.

2.3. Collaborative simulation in a distributed environment

A number of advantages can be obtained to perform collaborative simulation in a distributed environment. Firstly, distributed services can be created and provided by organizations with different expertise. Secondly, computing loads can be divided and shared within the virtual organization in collaboration. Thirdly, services are loosely coupled whilst being accessible to multiple clients. Although the increasing maturity of current software products provides a partial impetus for collaborative simulation, bottlenecks caused by data processing, distributed collaboration and heterogeneous computing environments still restrict its application [28]. To implement distributed collaborative simulation, not only the issues mentioned above but also the issues concerning distributed communication and interaction need to be addressed.

Shen et al. used TCP/IP connection to send control commands and sensor data between a control model and the virtual prototype in a VR-based environment [29,30]. However, it involves a large amount of coding work for TCP/IP connection once changes are made to models. Standards for distributed computing can be utilized to alleviate this situation as they encapsulate many of the details of implementing network connection. For example, the Common Object Request Broker Architecture (CORBA) was used in the DOME project to integrate distributed resources for engineering design [4–6]. CORBA is available for many programming languages and platforms but still suffers from a noticeable programming overhead [31]. Another standard, the High Level Architecture (HLA), was also used to support collaborative simulation

[32]. HLA is designed with the vision of distributed simulation in mind and provides good support for connecting and coordinating distributed simulators. However, HLA is not a light weight standard, and cannot on its own support collaborative simulation over the Internet [33,34].

The development of Web-related technology paves the way for Web-based modeling and simulation which attracts much attention in engineering research, see for example [2,7,28]. The more recent Web Services technology provides a platform-independent standard for distributed communication, and thereby improves the interoperability of heterogeneous applications. At the same time, it does not impose a large load on client-side programming, which makes it a good candidate for collaborative simulation. It has been used to implement Web-based simulation services, as well as to re-use proprietary computer-aided engineering environments [31,35]. Meanwhile, it is also utilized to implement distributed collaborative simulation [8,10]. The drawback of Web Services in supporting collaborative simulation on the Internet is that it is not a specialized standard for distributed simulation, which degrades its feasibility in complex simulation scheduling, especially for large-scale simulations. Combining HLA and Web Services to supplement each other has also been studied [33,34].

In summary, the paradigm shift towards distributed and collaborative product development systems has resulted in considerable research effort on supporting design and simulation in a distributed environment. However, previous work on collaborative simulation is mainly centred on the sharing of information and the implementation of distributed communication. In our opinion, further work still needs to be done to implement collaborative simulation on the Internet. Firstly, the platform should enable the clear division of tasks to allow engineers to focus on the work on which they have expertise. Secondly, the platform should require as little programming work as possible. Thirdly, an efficient mechanism of interaction needs to be elaborated to guarantee the accuracy of each simulation. This research is motivated by these requirements.

3. Formulation and requirements of collaborative simulation

Collaborative simulation on the Internet implies a scenario of creating, sharing, and integrating simulation services via the Internet, to evaluate design concepts at the system-level. It is worth mentioning that collaborative simulation is not a press-button task, but rather emphasizes the integration of human strengths (decision-making, creativity, analysis) and computer strengths (memory, speed, and connectivity). We prefer to first give a definition to collaborative simulation to define the scope of this research; that is, it is a methodology of creating simulations for engineering design via the division of complex problems and the synergy of sub-system models developed using different simulation tools. The purpose of using collaborative simulation is to evaluate design concepts, negotiate trade-offs, and facilitate decision-making. A typical scenario of collaborative simulation is to develop control algorithms for mechatronic products.

3.1. Formulation of collaborative simulation

As the simulation problems in mechatronic design (e.g. electronics, dynamics, and control) can generally be represented as a set of differential equations or differential algebraic equations (DAEs) [23], we mainly focus on continuous-time simulation in this paper, i.e., simulating the behaviour of a system over a period of time. Therefore, the term 'time' becomes very important for this research. In the following parts of this paper, 'simulation time' refers to a measure used to observe the changing of states of a simulated system whilst 'time step' refers to a measure used to quantify the passing of simulation time, unless otherwise indicated.

Generally, engineering design is a process of generating solutions and fulfilling requirements. The solving of design problems can be viewed as a process of narrowing down a design space by satisfying a set of constraints, until optimal feasible solutions are found. The constraints, which include relation constraints and bound constraints in general [36], can be identified from either the design requirements or the decision of selecting a specific concept. For complex problems, the identification of relation constraints can become very difficult and using specialized simulation tools can make this process easier. Similarly, we can define a simulation using a constraint-based representation, as shown in Eq. (1):

$$\begin{aligned} &\mathbf{I}(\mathbf{t}) = \{I_1(t)\dots I_{n1}(t)\}, \quad O(t) = \{O_1(t)\dots O_{n2}(t)\},\\ &\mathbf{V}(\mathbf{t}) = \{V_1(t)\dots V_{n3}(t)\}, \quad t \in R, \quad n1+n2+n3 = n,\\ &P = \{P_1\dots P_m\}, \quad X(t) = I(t)\cup O(t)\cup V(t),\\ &\text{s.t.}\begin{cases} \mathbf{g}(X,P,t) \leqslant 0 \quad \mathbf{g} = [g_1, g_2, \cdots, g_p]^{\mathrm{T}}\\ \mathbf{h}(X,P,t) = 0 \quad \mathbf{h} = [h_1, h_2, \cdots, h_q]^{\mathrm{T}}\\ X_L \leqslant X(t) \leqslant X_H \end{cases} \end{aligned} \tag{1}$$

In this definition, $\mathbf{X}(\mathbf{t})$ is the n-dimensional design variables vector which consists of the input vector $\mathbf{I}(\mathbf{t})$, the output vector $\mathbf{O}(\mathbf{t})$, and the vector of internal variables $\mathbf{V}(\mathbf{t})$. $\mathbf{P}$ is a group of parameters specified before the simulation, e.g. the dimension of a part or the time span of the simulation. The variable $\mathbf{t}$ represents the simulation time. A group of tightly-coupled mathematical functions is embodied as inequality constraints $\mathbf{g}$ and equality constraints $\mathbf{h}$. The design variables vector $\mathbf{X}(\mathbf{t})$ has lower bound X_{L} and higher bound X_{H}, respectively. In collaborative simulation, such a representation will be divided into two or more sub-systems each of which has a similar structure.

As shown in Fig. 1, a system is divided into three sub-systems which have interactions between each other at the run-time of simulation. Once a set of parameters are specified for each model, it starts the simulation based on the inputs provided, updates its internal states, and generates outputs, at regular time intervals. From the formulation of collaborative simulation, a number of issues can be identified as being critical to a collaborative simulation process: (1) identification of the variables and the parameters involved in the constraint-based equation, for the simulated system; (2) decomposition of the simulated system; (3) specification of the interactions between sub-systems; and (4) efficient data exchange between sub-systems during run-time. All these issues need to be taken into account in the development of the platform.

3.2. A development process of collaborative simulation

A collaborative simulation process generally starts with the simulation objectives and ends with the decisions made for the subsequent design process, involving a number of iterations where several sub-system models are integrated. Based on this definition, a standard development process of collaborative sim-

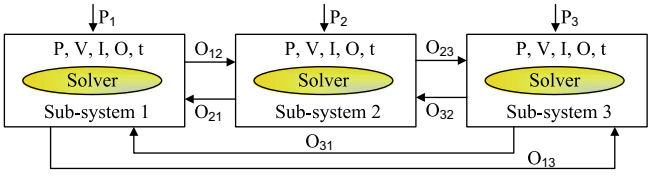


Fig. 1. The coupling between the sub-systems of a simulation system.

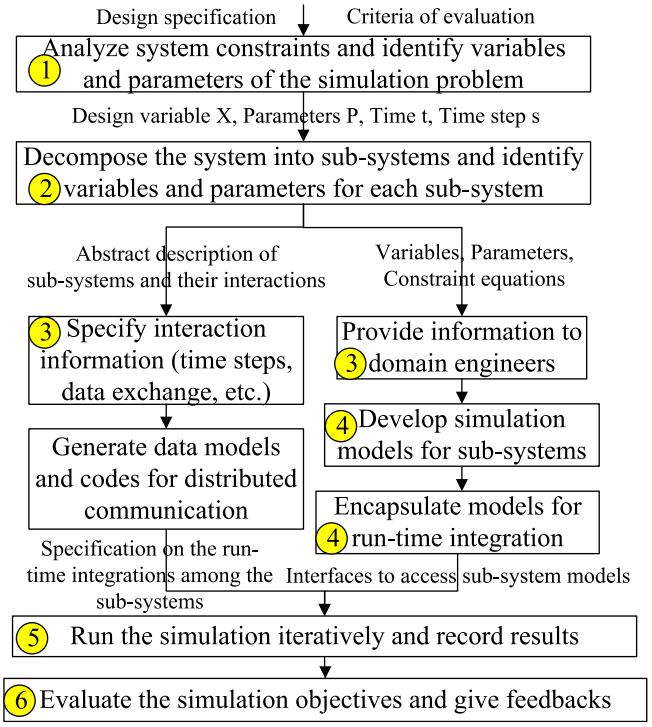


Fig. 2. A development process for collaborative simulation.

ulation can be identified to provide insights for the development of a software platform, as shown in Fig. 2. The process, which emphasizes both the functional decomposition (top–down) and the aggregation of component models (bottom–up), has six main steps each of which is highlighted in Fig. 2 with an indicator:

1. Analyze the system constraints and identify variables and parameters. In this step, the constraints, generated from simulation requirements and relations between components in the simulated system, are identified, together with the design variables, design parameters and time span for the simulation.
2. Decompose the simulated system into sub-systems, each of which will be modeled separately. The decomposition is performed according to the different disciplines that the sub-systems belong to and the division of engineering expertise, as well as the availability of simulation tools.
3. Provide the information generated in Step 2 to the simulation platform and the development tasks of sub-systems, respectively. The simulation platform will use the information to enable run-time interaction while the domain-specific information (variables, parameters, and equations) will be used to guide the development of each sub-system.
4. Develop and encapsulate models for each sub-system. In this step, simulation models for each sub-system are developed and made accessible to an external program, i.e., the simulation platform.
5. Run the collaborative simulation iteratively and record the results. The inputs for this step are the interaction methods between sub-systems, and the methods to accessible simulation services. This step will be repeated by updating the key parameters to obtain the results of simulation under different circumstances.
6. Evaluate the simulation objectives and give feedback to facilitate decision-making in the associated design process.

3.3. Requirements for a software platform

Simply, the core functionality of a collaborative simulation platform is to integrate models of sub-systems, synchronize the advancement of each model, and achieve efficient interactions between the models at run-time. To develop a platform supporting collaborative simulation on the Internet, new requirements need to be identified. For instance, it should provide an interactive environment for distributed development teams and an integrated environment for multi-disciplinary simulation models. A number of requirements for such a platform can thus be identified, as follows:

1. To enable distributed design teams to decompose complex design problems and evaluate simulation results in an integrated environment.
2. To support the high-level modeling of a simulated system and generate the necessary codes for the running of simulations which could be built by re-using legacy simulation models and codes.
3. To provide interfaces for the post-processing of simulation results, and the generation of simulation reports.
4. To effectively synchronize the simulation advancement so as to guarantee accurate results, with different events influencing each other in the correct sequence.
5. To signpost the simulation development by the provision of a process management facility.
6. To enhance the scalability of the platform and address security issues for effective operation.

In our current work, we mainly focus on items one, two, and four, while providing limited support for item three and five. To fulfill these requirements, we identify several solution principles and propose a number of methods to implement a prototype system supporting collaborative simulation on the Internet.

4. On the performance of collaborative simulation

The ultimate goal of collaborative simulation over the Internet is to obtain accurate simulation results useful for a design process, as well as to improve the efficiency of creating simulations. Therefore, the performance of collaborative simulation is crucial to its application in industry. In this section, we will discuss the factors influencing the performance of collaborative simulation, and develop an interaction mechanism to guarantee the accuracy of distributed simulation.

4.1. Factors influencing the performance

Based on our experience of running collaborative simulations, we have identified a number of criteria to evaluate the performance, namely accuracy, speed of running, convergence, and users' interaction with the simulation process. As discussed above, the solving of collaborative simulation essentially involves the numerical integration of a set of differential equations and DAEs using different solvers. Therefore, factors such as the size of time steps and the interaction between solvers can definitely influence the performance. The relevant factors influencing the accuracy, speed of running and convergence are listed in Tables 1–3. Some issues can be resolved using engineers' expertise, e.g. the selection of time steps and tolerances, and the order of starting sub-system models. Some other issues, on the other hand, can be resolved by the platform, e.g. interaction mechanisms, the dynamic adjustment of time steps at run-time, iterative strategies, etc. Apart from these factors, there are also some issues concerning users' interaction with a

Table 1
Factors influencing the accuracy.

Factor	Description
Accuracy of the models	The simulation will have good accuracy if the models of the sub-systems are accurate
Simulation time step	Generally, smaller time steps mean higher accuracy
Algorithms for numerical integration	Generally, using higher order algorithms can obtain higher accuracy
Interpolation	The use of interpolation can improve accuracy
Selection of tolerance	Generally, smaller tolerances result in higher accuracy
The order to start sub-system models	Occasionally, the order in which the models are started can affect the accuracy of the simulation
Interactions between sub-systems models	Using the right interaction mechanism can improve accuracy
Iterative algorithm	In some cases, using an iterative algorithm can improve accuracy, though it may require more computation and a longer running time
Errors in network communication	Communication errors in the network may affect the accuracy and even the stability of the simulation

Table 2
Factors influencing the speed of running.

Factor	Description
Complexity of the models	Generally, more complex models require more running time
Algebraic loop	If algebraic loops exist in models, the speed of running will be significantly affected
Algorithms for numerical integration	Generally, using higher order algorithms results in a longer run-time
Interpolation	The use of interpolation can slow down the speed of running
Selection of tolerance	Generally, smaller tolerances require more iteration, and thereby slow down the speed of running
Control strategy for the time step	Using variable-step simulation can save some run-time, especially for very complex simulations
Encapsulation of models	The ways in which models are encapsulated can significantly affect the speed of running
Delay in network communication	For collaborative simulation over the Internet, the efficiency of network communication is very critical

Table 3
Factors influencing the convergence.

Factor	Description
Continuity of the models	Generally, discontinuities in the models may cause instability and divergence
Simulation time step	Generally, a smaller step will achieve better convergence
Algorithms for numerical integration	Generally, using variable-step integration algorithm will improve convergence
Interpolation	In some cases, very high-order interpolation algorithms may cause instability and divergence
Selection of tolerance	The tolerances specified for the variable-step integration algorithm can affect the convergence
Coupling between sub-systems	Generally, the decomposition of a simulation generates algebraic loops which can affect the convergence, so should be avoided

simulation process, e.g. the users' interventions to start, pause, and stop a simulation.

4.2. A mechanism for efficient interaction

The numerical integration algorithms used by different solvers tend to differ [22], which makes it difficult to choose a suitable time step for all the models. The purpose of an interaction mechanism is to achieve improved accuracy by providing accurate input data for a model at the beginning of a time step. There are two widely used methods for interaction, both of which require the exchange of data at fixed intervals. In the first method (method 1), data exchange takes place at the beginning of a time step, for all the models. In other words, the simulation of a model at time t, depends only on the data generated before t. This method has been used in previous research including our early work [33,34,37]. The second method, on the other hand, allows a model at current simulation time t to obtain data from another model, not only before t, but also after t. This method is used by commercial software tools (e.g. Matlab-Simulink and MSC.Adams) to create mutual interfaces [22].

Generally, the interval of data exchange chosen by the above two methods may not suit the internal time step of every model, so interpolation algorithms are used to find the results at the end of a time step when input data are not given. It is not sensible to employ very high-order algorithms, since the interpolation will be very sensitive to any rounding errors in the sample points. In our case study, a quadratic interpolation method is adequate. Assume that we have three points (t_0, x_0), (t_1, x_1) and (t_2, x_2), then the point at any time t can be calculated in Eq. (2) and the interpolation base functions are calculated in Eq. (3).

$$f(t) = x_0 l_0(t) + x_1 l_1(t) + x_2 l_2(t) \tag{2}$$

$$\begin{cases} l_0(t) = \frac{(t-t_1)(t-t_2)}{(t_0-t_1)(t_0-t_2)} \\ l_1(t) = \frac{(t-t_0)(t-t_2)}{(t_1-t_0)(t_1-t_2)} \\ l_2(t) = \frac{(t-t_0)(t-t_1)}{(t_2-t_0)(t_2-t_1)} \end{cases} \tag{3}$$

Although the use of interpolation algorithms can to some extent improve the accuracy of simulation, the above two methods rely heavily on selecting a suitable interval and do not take into account the time steps used in the numerical integration of each model. We therefore propose an interaction mechanism which while not requiring each model to simulate a fixed step, allows each model to advance the simulation until the current calculation converges, and then constructs outputs for the other models using quadratic interpolation. Such a mechanism can be termed method 3 and an illustration of all the three interaction methods is shown in Fig. 3. To evaluate the performance of these methods, we run a simple simulation in Matlab-Simulink to get standard results and then divide the simulated system into two sub-systems the mathematical models of which are as shown in Eqs. (4) and (5).

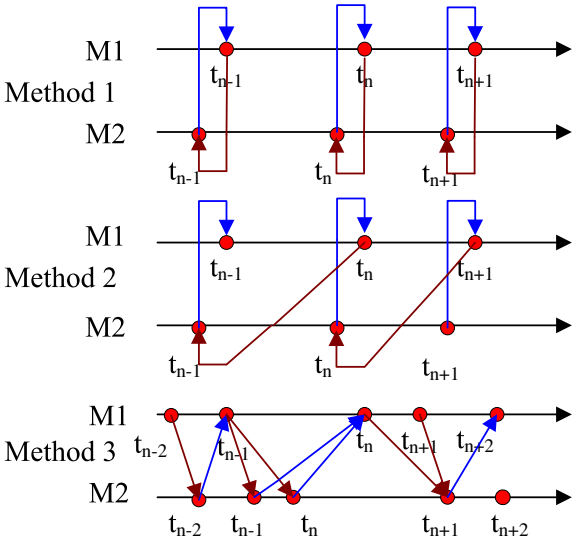


Fig. 3. Three methods for coordinating the interaction between sub-system models.

$$\begin{cases} \dot{x}_1 = -x_1 + y_2 \\ y_1 = x_1 + 2y_2 \end{cases} \quad (4)$$

$$\begin{cases} \dot{x}_2 = -x_2 + y_1 \\ y_2 = x_2 \end{cases} \quad (5)$$

The simulation time span is 1.0 s and the interval for data exchange is 0.01 s. The errors are quantified by calculating the difference between the values obtained from a single, integrated simulation with those obtained from simulations using the three methods, respectively. As shown in Fig. 4, errors in both method 1 and method 2 are very manifest at the beginning of the simulation while the error of method 3 looks small throughout the simulation. Based on this case, we can draw a conclusion that the accuracy of simulation increases from method 1 to method 3. The utilization of interpolation algorithms can improve the accuracy of simulation. We tried method 3 under three circumstances, namely without interpolation, with first-order interpolation, and with second-order interpolation. Errors in these experiments are calculated in the same way as mentioned above. As shown in Fig. 5, the experiment without interpolation is the least accurate while the distinction between the other two is not very obvious.

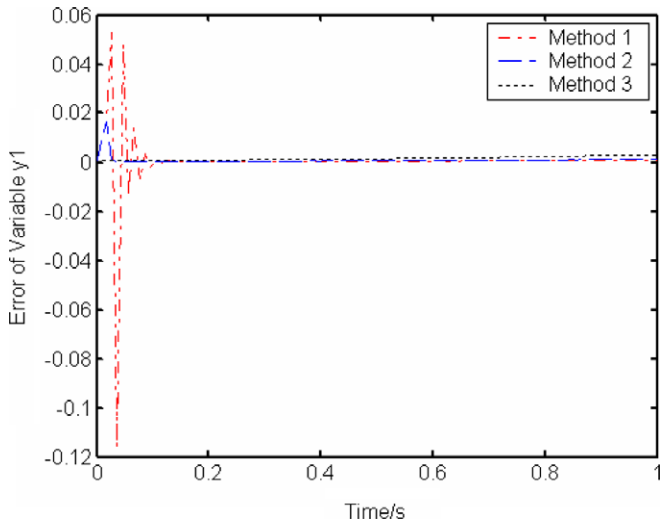


Fig. 4. A comparison of the errors in variable y_1 for the three methods.

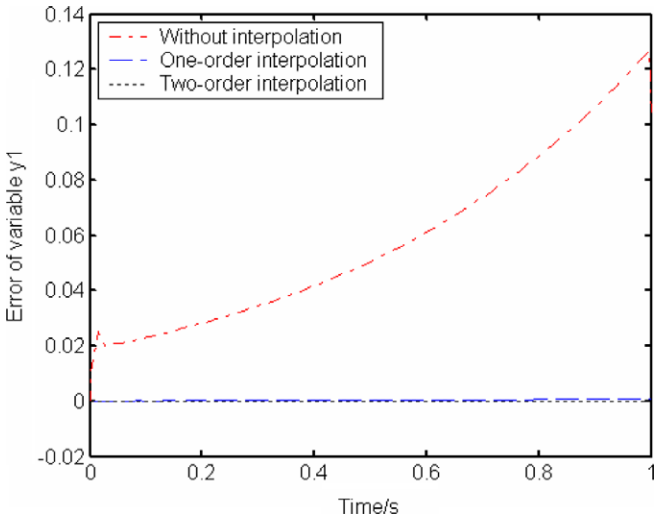


Fig. 5. A comparison of simulation results obtained in different simulation settings.

Therefore, interpolation algorithms are clearly helpful to collaborative simulation, especially when there is a significant mismatch between the step sizes of the different models.

5. Solution principles and approaches

5.1. Architecture and infrastructure

A collaborative platform on the Internet not only assists the cooperative work of various users from different sites, but also supports the run-time integration of several simulation models which may also be highly distributed. The richness of Web technology makes the development of a Web-based interactive environment less complicated and less expensive than it used to be. However, interacting with computer software on the Internet is still restricted by the performance requirements, heterogeneous computing platforms and security issues. Moreover, although a collaborative simulation platform facilitates the sharing of intents and simulation codes, it must be stressed that information and models should still be shared cautiously, from the perspective of security issues and intellectual property [38].

Moreover, although an interaction mechanism has been proposed to guarantee efficient interaction, the issue of managing such interaction in a distributed environment still needs to be addressed. Firstly, the fidelity and causality of simulations need to be fulfilled in terms of distributed interaction. Specifically, fidelity means that a model gets the right input from a right peer at the right time whilst causality means a model can get several inputs from a number of peers in the right order. Secondly, users of the platform can join a simulation process to give input and control at run-time. A framework based on Web Services and HLA is developed in the light of the above issues. The detailed structure of this framework and a model-driven approach to bridge the high-level modeling and infrastructure are beyond the scope of this paper and have been published elsewhere [37].

Web Services can be defined as an open standard which specifies how distributed applications communicate and interoperate on the Web. Web Services is a promising technology to improve the interoperability of heterogeneous computing resources as it is independent of any computing platform and programming language. A number of advantages can be obtained by using Web Services in our solution. Firstly, it can be developed based on the standard Hypertext Transfer Protocol (HTTP), which makes the connection through Web Services friendly to firewalls. Secondly,

it is published by only providing the interfaces while protecting the details about services realization, which allows confidential models to be shared without risk. Last but not least, the enthusiastic Web Services community continues to make regular improvements to Web Services standards and frameworks, which generally reduce the complexity of employing this technology in a simulation platform. HLA, as an IEEE standard for distributed simulation, involves a set of rules, an interface specification, and an object model. HLA can help to synchronize a number of simulation models and enable users' control over a simulation process. The platform architecture and system framework are shown in Fig. 6.

On the left of the figure is the three-layered architecture of a Web-based collaborative simulation platform, in which the 'Web layer' provides an interactive interface for the human computer interaction during the simulation process. For instance, the Web-based graphical interface and the logic of information flow within the platform are included in this layer. The 'Facility layer' can be perceived as a set of facilities, e.g. simulation management and/or high-level model descriptions, designed to meet the functional requirement of the platform. Collaborative simulation stems from a broad spectrum of necessary hardware and software components which are included in the 'Infrastructure' layer.

The underlying infrastructure for distributed computing is of significant importance to the system implementation, and hence a framework based on HLA and Web Services is identified by us as illustrated on the right of Fig. 6. Two functional components are contained in that framework, namely: the Service Encapsulation Component (SEC) and the Federation Execution Component (FEC). SEC serves as the software component which encapsulates simulation software/models as services while FEC is designed to integrate and synchronize the simulation federation. These functional components may reside in different sites on the Internet and communicate with each other to construct a federated simulation execution environment.

5.2. Service Encapsulation Component

Similar to other model wrapping methods, SEC is recognized as the component to control the simulation process of a specific model. A SEC consists of two main parts, namely: Service Communications Shell (SCS) and Embedded Simulation Script (ESS). SCS is the container where ESS resides, and it is implemented based on the specific distributed computing standards and network protocols provided by the system infrastructure. The Web Services community has defined the necessary protocols for Web Services technology and there are plenty of resources which assist in the deployment of Web Services. Consequently, the implementation of SCS will not be very difficult. ESS is the external control script for a specific model, which is implemented based on the application programming interface (API) of each specific simulation tool. The interaction mechanism based on method 3 is implemented in ESS. The data processing algorithm which represents the operation mechanism of ESS is shown in Fig. 7.

5.3. Federation Execution Component

The Federation Execution Component (FEC) is a software component with a structured operation process, which is perfect for semi-automatic code generation. FEC is the most complicated part in our framework since it must conform to the HLA standard as a federate whilst it communicates with SEC as a service client. The operation mechanism of FEC is shown in Fig. 8.

The left part of Fig. 8 shows the operation process of FEC which starts when a particular HLA federate is initialized, and ends when this HLA federate resigns from the simulation federation. A HLA federate is a basic element in a HLA-based simulation federation and contains local time, internal state and published/subscribed data. The steps after initializing a HLA federate, e.g. joining or resigning from the federation, publishing/subscribing data and entering the main simulation loop, are the embodiments of HLA-based simulation management. The main simulation loop is the core of the simulation as depicted on the right of Fig. 8. Any federate of a simulation federation will enter such a loop and complete all tasks (such as advancing the simulation and exchanging data) which are necessary for the HLA-based simulation.

The simulation issues, e.g. fidelity and causality, are addressed in that loop. Now we will explain the key operations, which are numbered 1–5 in Fig. 7. Operation one represents the suspension

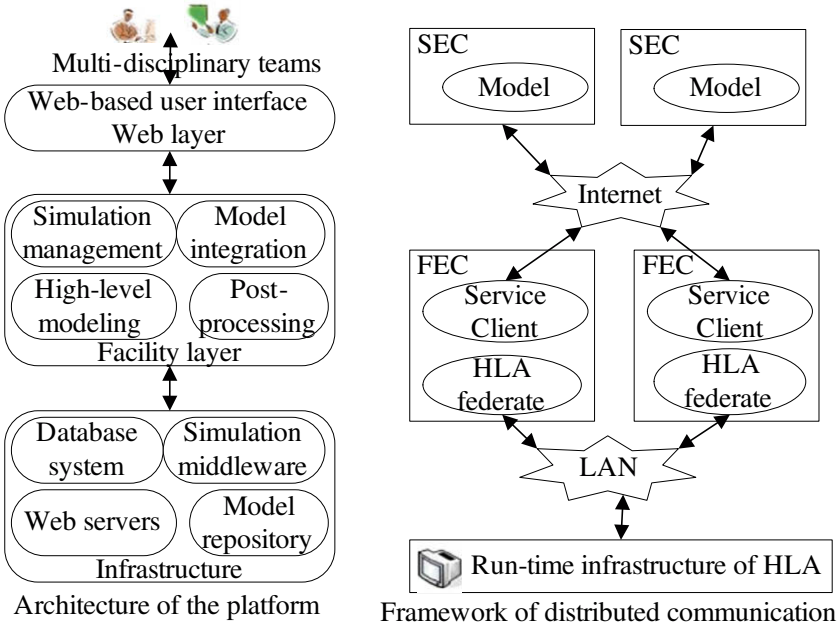


Fig. 6. Architecture and infrastructure of the proposed solution.

Algorithm 1: the input/output processing algorithm of SEC
Input: Start time *Ts*, End time *Te*, Input *Ipt*
Output: Output *Opt*
1. **if** (the vector storing inputs is null)
2. **do** construct an empty vector *IptVct* **end**;
3. **do** assign *Ts* to current simulation time *curTime* **end**;
4. **while** (*curTime*<*Te*)
5. **if** (**sizeof**(*IptVct*)>=2) **do**
6. Get the latest two points *P1*, *P2* from *IptVct*;
7. Using equation (2-3) to get a new point *Np* based on
8. *P1*, *P2*, and *Ipt* **end**;
9. **else if** (**sizeof**(*IptVct*)==1) **do**
10. Get the only point P3 from IptVct;
11. Get a new point *Np* based on *P3* and *Ipt* **end**
12. **else do** Create an new point *Np* and assign *Ipt* to it; **end**
13. **do** Start simulation with *Np* for a convergent time step *ct*;
14. Get simulation result *Sr* from the model;
15. Store *Sr* in output vector *OptVct* and Store *Ipt* in *IptVct;* **end**
16. **if** (*Np*!=*Ipt*) **do** store *Np* in *IptVct* ;**end**
17. **do** *curTime=curTime+ct;* **end**
18. **do** Interpolate the latest three points from *OptVct*; **end**
19. **do** Get the interpolation result *Ir*, return *Opt*=(*Ir*, *Te*); **end**
end Algorithm

Fig. 7. Input/output processing algorithm of SEC.

of the current thread to wait for HLA events (container of exchanged data and messages) triggered in other threads. Operation two shows that the federate is waiting for the time advancing request to be granted. RTI will control the global time of all federates to keep them synchronized and only grant a request when the events prior to this request have all been processed. Operation three marks the action of digesting received data and Operation four is the operation of invoking a remote service to simulate a model for a specified time span. Operation five is the explicit request for exchanging data by updating the published data, which will trigger an event in the following time step. A code template can be derived from this execution process to facilitate the automatic generation of FEC components.

6. Prototype system and case study

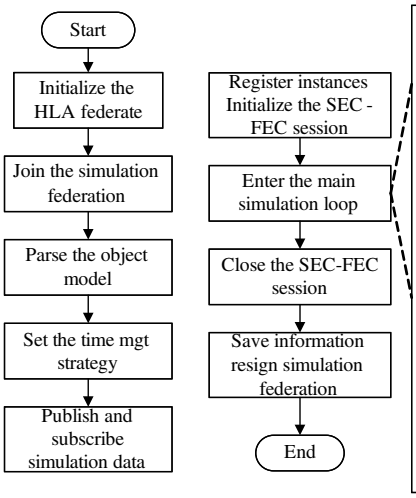
The basic concepts and main solution principles for a collaborative simulation platform on the Internet are discussed above. To explore the process of collaborative simulation with our proposed approaches, we developed a Web-based prototype platform and looked at a case study of a ball-beam controller.

6.1. System implementation

A prototype platform is designed as a Web-based system and implemented in Java, leveraging open-source resources. The framework of this Web system is designed based on the Struts framework of Apache [39] and the Web application is deployed via the Apache Tomcat Web server [40]. The Apache Axis engine [41] is chosen as the Web Services middleware and a limited edition of the Pitch RTI product [42] is applied as the HLA-enabling tool. The server-side data of the Web-based system are stored in the MySQL database management system [43]. The geometric model of the design can be shown in our system provided they are exported as either Virtual Reality Modeling Language (VRML) or extensible 3D (X3D) [44].

Our current work is centred on supporting the collaborative work of engineers to model an engineering system at the abstract level, integrate simulation services, and analyze simulation results obtained from different simulation settings. Some snapshots of five typical Graphical User Interfaces (GUIs) of the prototype are shown in Fig. 9, illustrating how these tasks are supported. Users can work on a task by pressing a link on the left of the Web page, and an appropriate interface will be loaded on the right part of the Web page in response to this. The labels one to five indicate five stages of a project in which the Web-based interfaces are used. Generally, the creation of simulations involves several team members with different roles, namely simulation experts, design engineers, and IT engineers. We will introduce a scenario of working with our prototype, based in Fig. 9.

In stage one, project leader defines the working process of the project which monitors the progress of a project and informs team members of the progress. In stage two, simulation experts and design engineers decompose a simulation system and specify the interactions between the sub-systems. In stage three, IT engineers specify how to access the simulation services. In stage four, simulation experts configure the simulation settings and start the simulation. In stage five, simulation experts and design engineers work together to analyze the simulation results. This process will be repeated each time changes need to be made to the models, until a satisfactory result has been obtained. After the simulation, the data can be saved in a database, to be used in future design work.



Algorithm 2: Main simulation loop of FEC
Input: SEC-FEC session
Output: The simulation status
1. **do**
1. (1) Wait for a while to collect HLA events ;
2. Compute the time for next step *Tc*;
3. Request to advance local time to *Tc*;
4. **Loop**: **begin**
5. (2) **if** (time advancement is granted) **do** Break Loop;
6. **else do** Retrieve events from the event queue ;
7. Process the events;
8. **if** (command to stop simulation is received)
9. **do S**et the flag stop-command-received;
10. Break Loop;
11. (3) **else** Process the values received ;
12. **do** Communicate with SEC; (4)
13. Process the simulation results of SEC ;
14. (5) Update the published data;
15. **until** (flag stop-command-received is set)
end algorithm

Fig. 8. Operation process of FEC.

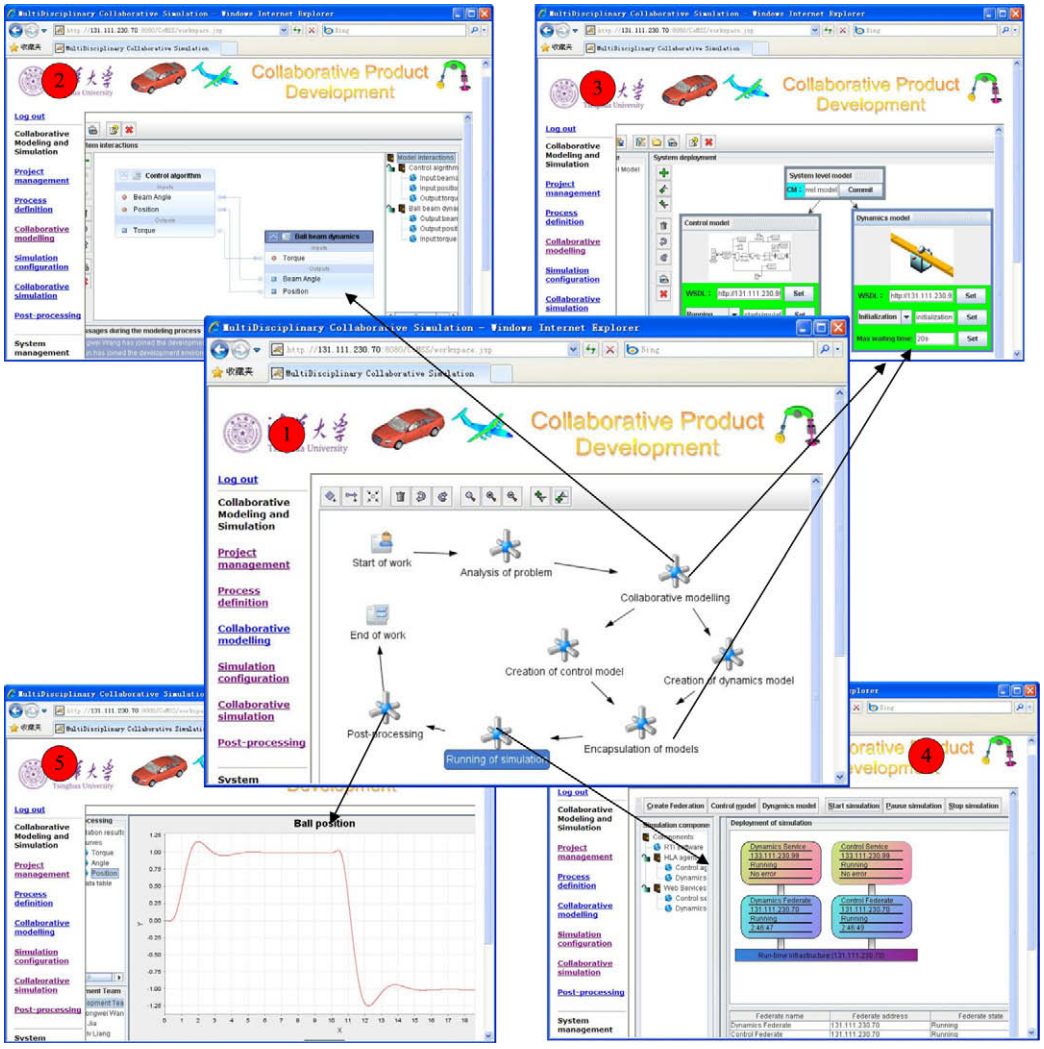


Fig. 9. Performing a collaborative simulation on the prototype platform.

6.2. Case study

To verify the feasibility of the proposed solution and the prototype system, we performed a simple case study based on the platform, to develop a control algorithm for a beam. The purpose of the simulation is simply to see whether the control algorithm can make the ball move by following a specified route. The simulated system can be divided into two sub-systems, namely a control model and a dynamics model. The control model receives the current angle of beam and the current position of the ball from the dynamics model, and gives a torque to drive the motion of the beam in response to these inputs. Fig. 10 shows the interface for simulation configuration, as well as the interactions between the two sub-system models.

The control model is created using Matlab-Simulink and the dynamics model is created using MSC.Adams. The interval of data exchange is set as 0.01 s and the interaction is implemented on the platform using method 3 proposed in Section 4. Both of the models are built to run separately by engineers in different sites, and are then encapsulated as Web Services with the help of IT engineers. When changes need to be made to the models, e.g. adding an input port or updating a parameter, the simulation engineers and design engineers can simply update the model in the platform without concerning themselves about other parts of the system. The total simulation is from 0 to 20 s, which means that the whole simulation process involves 2000 data exchange events. In the network environment of our laboratory, the simulation takes less than three minutes. To evaluate the accuracy of the simulation, we also performed ball-beam simulation based on the interfaces between Matlab-Simulink and MSC.Adams, and compared the results with those obtained based on our prototype. As shown in Figs. 11–13, the prototype platform has a comparable accuracy to the interface-based approach for the ball-beam simulation, with the curves obtained from the prototype looking somewhat smoother in the places where the values change very fast.

6.3. Discussion and comparison

With the prototype, users with different roles, e.g. simulation engineers and design engineers, can work co-operatively to complete complex simulation tasks. Simulation services can be created separately, integrated via the Internet at run-time and re-used in other projects. Meanwhile, good accuracy is achieved by our proposed interaction mechanism. Compared with the interface-based approach, several advantages can be seen in our worked example, even though the simulation takes a little longer. First of all, our approach aims to support collaborative simulation over the Internet, i.e., facilitating the process of collaborative problem-solving and re-using legacy resources in a more general way. Moreover, in terms of efficiency, it takes less time to start a new round of

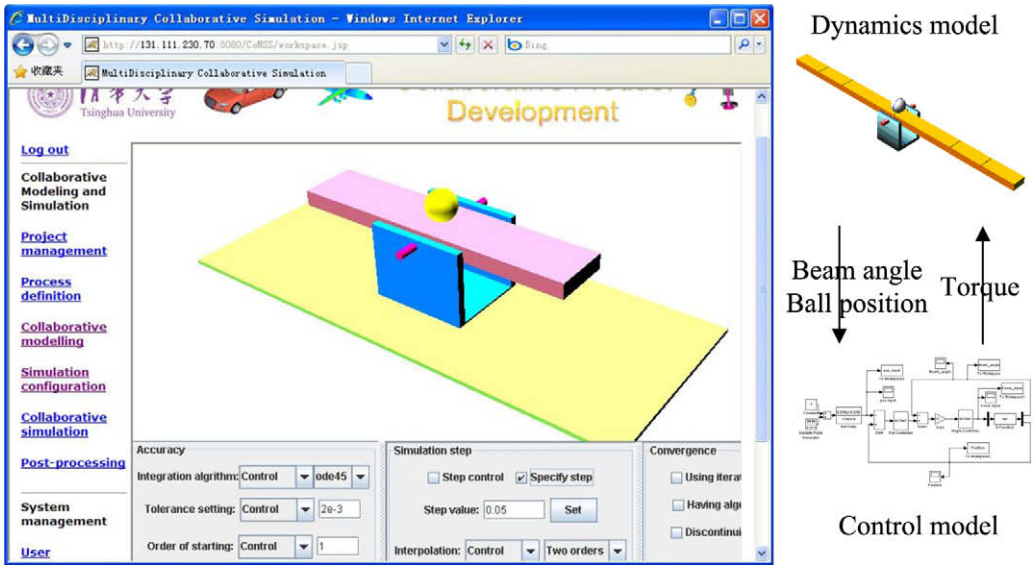


Fig. 10. Graphical interface for the ball-beam case and interactions between sub-systems.

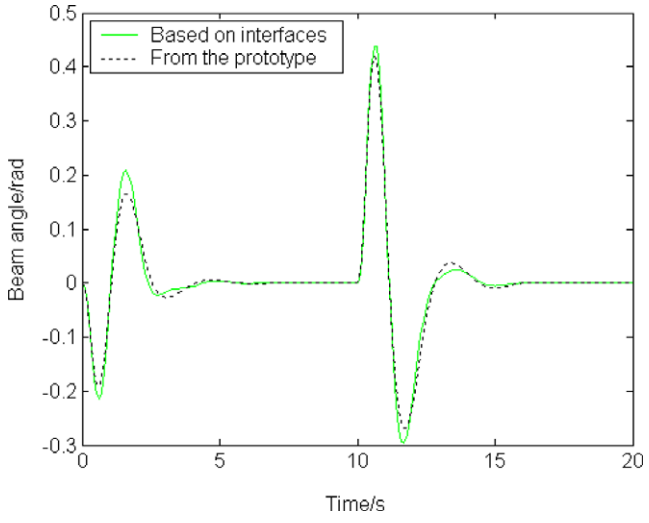


Fig. 11. Comparison of beam angle results obtained from the two simulations.

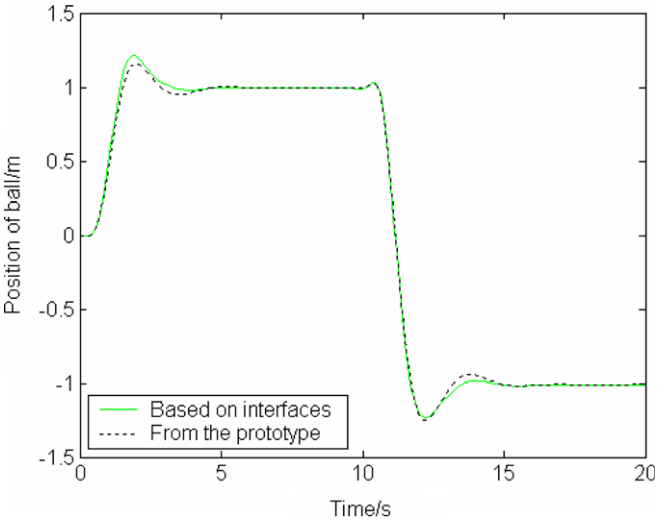


Fig. 13. Comparison of ball position results obtained from the two simulations.

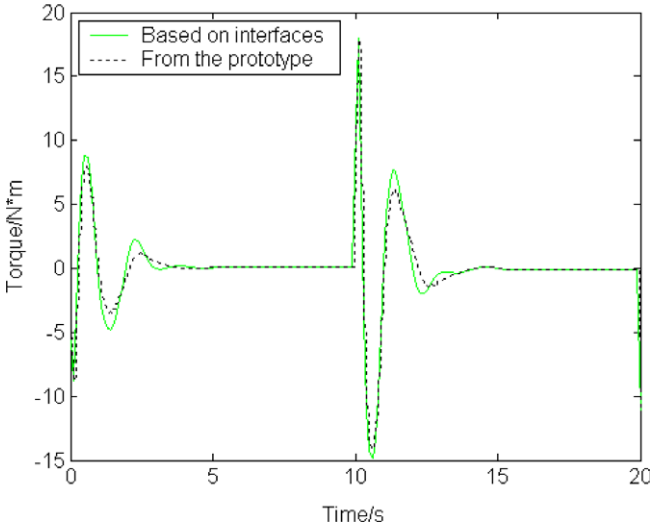


Fig. 12. Comparison of control torque results obtained from the two simulations.

simulation when changes need to be made to the models. This is difficult to achieve using traditional approaches, where the models need to be rebuilt, recollected and re-assembled before new simulation begins, so this advantage becomes more manifest when the design problem requires many iterations.

Compared with other research work on implementing distributed simulation services [4–7,9], our prototype not only focuses more sharply on the sharing of information and the connection of distributed computing resources, but also addresses the issues of model changes and distributed interaction. Although Web Services technology has been used to improve the interoperability between heterogeneous simulation models [4,31,35], we go beyond this purpose and emphasize the dynamic integration and synchronization of simulation services. Compared with the distributed mechanical system simulation platform [8], our prototype aims to support the collaborative work of engineers, and does not require users to specify the low-level interactions between the individual simulations. Although both HLA and Web Services are relatively complex for engineers to use, users only need to deal with the high-level system model; and moreover, the employment

of SEC and FEC reduces development work. Therefore, we believe the proposed solution and methods to be viable, and to represent a further step towards developing products via cross-organization collaboration.

7. Conclusions

In this paper, we present our work on supporting collaborative simulation for complex engineering systems via the Internet. Based on the proposed solution principles and approaches, a prototype platform has been developed, which provides an integrated environment for collaborative problem-solving and distributed simulation. Based on our early work, a solution combining Web Services and HLA is proposed to develop a holistic infrastructure from which a number of advantages can be obtained. Firstly, wrapping models through Web Services enables models to be accessed and re-used on the Internet while the model details are protected by only providing service interfaces. Secondly, managing a simulation federation using HLA can help solve synchronization and coordination problems even for complex simulation scenarios. Thirdly, the Web-based prototype platform supports the collaboration between the team members and enhances the management of projects. In our future work, we will focus on how to control the simulation step and how to deal with problems involving algebraic loops. In addition, we are also working on implementing further functions in our prototype system, to allow the simulation of more complex systems.

References

[1] 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.
[2] A. Skjellum, An open framework for developing distributed computing environments for multidisciplinary computational simulations, Ph.D. Thesis, Mississippi State University, United States of America, 2003.
[3] S. Szykman, R.D. Sriram, Design and implementation of the Web-enabled NIST design repository, ACM Transactions on Internet Technology 6 (1) (2006) 85–116.
[4] N. Borland, D. Wallace, Environmentally conscious product design: a collaborative Internet-based modeling approach, Journal of Industrial Ecology 3 (2–3) (2000) 33–46.
[5] N. Senin, D. Wallace, N. Borland, Distributed object-based modeling in design simulation marketplace, Journal of Mechanical Design 125 (1–2) (2003) 2–13.
[6] S. Abrahamson, D. Wallace, N. Senin, P. Sferro, Integrated design in a service marketplace, Computer-Aided Design 32 (2) (2000) 97–107.
[7] T. Larsson, A. Larsson, L. Karlsson, A modular approach to Web based multibody dynamic simulation, in: Proceedings of the 2001 International CIRP Design Seminar, Design in the New Economy, Stockholm, Sweden, 2001.
[8] J. Wang, Z.-D. Ma, G.M. Hulbert, A distributed mechanical system simulation platform based on a gluing algorithm, Journal of Computing and Information Science in Engineering 5 (1) (2005) 71–76.
[9] C.J.J. Paredis, A. Diaz-Calderon, R. Sinha, P.K. Khosla, Composable models for simulation-based design, Engineering with Computers 17 (2) (2001) 112–128.
[10] B. Johansson, P. Krus, A Web Service approach for model integration in computational design, in: Proceedings of ASME 2003 Computers and Information in Engineering Conference, Chicago, USA, September 2003.
[11] G. Ferretti, G. Magnani, P. Rocco, Virtual prototyping of mechatronic systems, Annual Reviews in Control 28 (2) (2004) 193–206.
[12] G. Toya, M.R. Cutkosky, L.J. Leifer, J.M. Tenenbaum, J. Glicksman, SHARE: a methodology and environment for collaborative product development, Technical Report CDR-TR#19930507, Stanford University, 1993.
[13] M.R. Cutkosky, J.M. Tenenbaum, J. Glicksman, Madefast: collaborative engineering over the Internet, Communication of ACM 39 (9) (1996) 78–87.
[14] T. Tuikka, M. Salmela, WebShaman: shared virtual prototypes for product designers in the World Wide Web, ACM SIGGROUP Bulletin 20 (1) (1999) 51–55.
[15] D. Brown, D. Leal, C. McMahon, R. Crossland, J. Devlukia, A Web-enabled virtual repository for supporting distributed automotive component development, Advanced Engineering Informatics 18 (3) (2004) 173–190.
[16] J.-H. Kim, H.-J. Lee, S.-H. Kim, J.-O. Lee, A problem solving environment portal for multidisciplinary design optimization, Advances in Engineering Software 40 (8) (2009) 623–629.
[17] Y.D. Wang, W. Shen, H. Ghenniwa, WebBlow: a Web/agent-based multidisciplinary design optimization environment, Computers in Industry 52 (1) (2003) 17–28.
[18] L. Schubert, A. Kipp, B. Koller, Supporting collaborative engineering using an intelligent web service middleware, Advanced Engineering Informatics 22 (4) (2008) 431–437.
[19] P. Breedveld, Port-based modeling of mechatronic systems, Mathematics and Computers in Simulation 66 (2–3) (2004) 99–128.
[20] H. Giese, S. Burmester, W. Schäfer, O. Oberschelp, Modular design and verification of component-based mechatronic systems with online-reconfiguration, in: Proceedings of the 12th ACM SIGSOFT 12th International Symposium on Foundations of Software Engineering, Newport Beach, USA, November 2004.
[21] A. Sodja, B. Zupančič, Modelling thermal processes in buildings using an object-oriented approach and Modelica, Simulation Modelling Practice and Theory 17 (6) (2009) 1143–1159.
[22] J. Larsson, P. Krus, J.-O. Palmberg, Modelling, simulation and validation of complex fluid and mechanical systems, in: Proceedings of the Fifth Conference on Fluid Power Transmission and Control, Hangzhou, PR China, April 2001.
[23] A. Rukgauer, W. Schiehlen, Simulation of modular mechatronic systems with application to vehicle dynamics, Acta Mechanica 125 (1–4) (1997) 183–196.
[24] Matlab-Simulink, 2009. Available from: <http://www.mathworks.com/products/simulink/>.
[25] MSC.Adams, 2008. Available from: <http://www.mscsoftware.com/products/adams.cfm>.
[26] Hopsan, 2008. Available from: <http://www.flumes.ikp.liu.se/tools/hopsan/index_en.xf>.
[27] A. Johnson, An open architecture approach to kinematic analysis for computer-aided embodiment design, Computer-Aided Design 30 (3) (1998) 199–204.
[28] J.A. Reed, G.J. Follen, A.A. Afjeh, Improving the aircraft design process using web-based modeling and simulation, ACM Transactions on Modeling and Computer Simulation 10 (1) (2000) 58–83.
[29] 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.
[30] Q. Shen, M. Grafe, To support multidisciplinary communication in VR-based virtual prototyping of mechatronic systems, Advanced Engineering Informatics 21 (2) (2007) 201–209.
[31] S.M. Eissen, B. Stein, Realization of Web-based simulation services, Computers in Industry 57 (3) (2006) 201–209.
[32] 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.
[33] 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.
[34] H. Wang, A. Johnson, H. Zhang, The assembly of computational models for the collaborative development of virtual prototypes, in: Proceedings of the ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, San Diego, USA, August 2009.
[35] E.G. Roselló, M.J. Lado, A.J. Méndez, 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.
[36] Z. Yao, A. Johnson, On estimating the feasible solution space of design, Computer-Aided Design 29 (9) (1997) 649–655.
[37] H. Zhang, H. Wang, D. Chen, G. Zacharewicz, A model-driven approach to multidisciplinary collaborative simulation for virtual product development, Advanced Engineering Informatics, doi: 10.1016/j.aei.2009.07.005.
[38] Z. Zdrahal, P. Mulholland, M. Valasek, A. Bernardi, Worlds, transformations: supporting the sharing and reuse of engineering design knowledge, International Journal of Human–Computer Studies 65 (12) (2007) 959–982.
[39] Struts framework, 2008. Available from: <http://struts.apache.org>.
[40] Apache Tomcat Web server, 2008. Available from: <http://tomcat.apache.org/>.
[41] Axis Web services middleware, 2008. Available from: <http://ws.apache.org/axis/>.
[42] Pitch RTI. Available from: <http://www.pitch.se/>.
[43] MySQL database management system, 2008. Available from: http://www.mysql.com/.
[44] Web 3D consortium, 2009. Available from: <http://www.web3d.org/>.