On HLA-based Collaborative Simulation Techniques

Miaomiao Fan, Yushun Fan
National CIMS engineering research center
Tsinghua University
Beijing, P.R. China
fanmm06@mails.tsinghua.edu.cn, fanyus@tsinghua.edu.cn

Abstract—The design and development of a complex system as well as the simulation of its dynamic behavior are usually involved in multidisciplinary field knowledge. HLA-based collaborative simulation is an effective way to integrate different disciplinary models together in distributed environments. In this paper, the HLA adaptor is adopted to realize model transformation so that models created through commercial software are compatible with HLA. Moreover, in order to solve the difficulty of wrapping ABAQUS software, an ABAQUS engine encapsulation technique is proposed. It is helpful for developing HLA adaptor corresponding to ABAQUS software, and can facilitate the integration of ABAQUS models with HLA-based collaborative simulation platform. Next, a time-advance algorithm based on time-stepped is described, which is the main time-advance way in complex system simulation. Finally, a prototype system is developed to demonstrate the high fidelity of collaborative simulation and the effectiveness of our simulation platform.

Keywords—HLA; multidisciplinary collaborative simulation; engine encapsulation; ABAQUS

I. INTRODUCTION

The forging manipulator system is one of the key equipments in the manufacturing industry, which is characterized by complex nonlinearity, strong coupling, multiple variables and multiple degrees of freedom. The design and development of such a complex system as well as the simulation of its forging and manipulating process is multidisciplinary in nature. It involves many different fields, including mechanics, kinetics, kinematics, hydromechanics, plastic deformation theory of metals and so on. To analyze such a complex system accurately, not only the problems in each individual discipline, but also the interactions among different disciplines should be simulated in high fidelity. As a single domain simulation cannot analyze the multidisciplinary behavior of a complex system comprehensively and accurately, it is indispensable to adopt multidisciplinary collaborative simulation, which is able to integrate models in different disciplines and simulation tools together, and facilitate better communication and cooperation among different development teams as well.

The High Level Architecture (HLA) was initiated by the US Defense Modeling and Simulation Office (DMSO) in 1996. It defines the rules and interface specification to support reusability and interoperability among simulation object models or federates, and reusability and interoperability are the two most important features of HLA [1].

HLA consists of 3 parts: HLA rules, interface specification [2] and object model template (OMT). HLA attempts to specify a standard interface between simulations and separates these interfaces from the implementation of any specific simulation. HLA represents the software system architecture and RTI (Run Time Infrastructure) is its implementation. Within HLA, federations are defined as a group of federates forming a community. The formation of federations is based on a number of simulation components called federates that exchange information in the form of objects or interactions. The possible messages among federates of a federation are defined by the Federation Object Model (FOM). The capabilities of a federate to interact with others are defined by the Simulation Object Model (SOM) [3].

In this paper, a HLA-based multidisciplinary collaborative simulation technique is studied. More specially, a method of simulation engine encapsulation is proposed to facilitate the wrapping of commercial simulation tools and corresponding models. This paper is organized as the follows: The next section introduces the framework of model transformation; In Section 3, the scheme of simulation engine encapsulation is presented; Section 4 propose a method of wrapping ABAQUS simulation tool; a time advancement algorithm based-on time stepped is described in Section 5; In Section 6, a prototype system for forging and manipulating process is implemented to verify the collaborative simulation technology adopted in this paper; A conclusion is given in Section 7.

II. MODEL TRANSFORMATION

In nowadays, during the simulation fields, most of the analysis models are created using off-the-shelf commercial software, such as Matlab for control system, ADAMS for kinetics, ANSYS and ABAQUS for finite element analysis (FEA). There may exist interfaces between these software, but all of them do not support HLA standard and specification. To make them compatible with HLA, models have to be transformed to some degree. In order to inherit and reuse the single domain model information and the modelling knowledge of experts as well as to improve the flexibility and expandability of simulation platform, we adopt the simulation tools and corresponding collaborative simulation adaptor, which is also called HLA adaptor to transform original model, making them meet HLA standard. Under this situation, the whole transformed model can be used as a federate to directly integrated into HLA-based collaborative simulation platforms.
A. Framework of HLA adaptor

The structure of HLA adaptor is shown in Fig. 1. HLA adaptor mainly contains [4]:

- Models transformation. It concerns on model operation, data interface mapping and reconstruction of soft bus data to improve simulation precision.
- Simulation time advance. It copes with the advance of simulation time (commercial software, federate) and the coordination caused by the step mismatch between commercial software and federates through interpolation of commercial software output.

![Figure 1. Structure of HLA adaptor](image)

Real-time message interaction is realized through calling RTI service and RTI callback. This mechanism is implemented by RTI through two ambassador paradigms: RTIAmbassador and FederateAmbassador.

B. Model Transformation

The model transformation facility consists of three parts: model operation, data interface map, interaction maintenance.

Model operation is a low level operation for disciplinary model which concerns on specific software utilized to construct the model. Model operation facility wraps the behaviours of commercial software (start, pause, simulate to a specific time, stop, etc).

Data interface mapping is proposed to realize the high level operation of disciplinary models. Here, high level means caring nothing about the details of a specific model and the corresponding commercial software. The map is between the variables in disciplinary model and the SOM data of the federate model.

Interaction maintenance is a mechanism for processing interactions properly. Since the RTI callback can happen at any time, the interaction maintenance facility deals with the callback in a multi-threaded way and stores the interactions as events in a queue. When it is driven by simulation step or event, the adaptor dequeues the interactions and handles them for message dissemination or simulation time advancement.

III. SIMULATION ENGINE ENCAPSULATION

A. Scheme of Simulation Engine Encapsulation

Model operation is of vital importance in developing HLA adaptor corresponding to simulation tools. The realization of model operations is through simulation engine encapsulation technique, which, according to developing simulation engine agent, fulfils the following functions: the operation on model input and output port, the modification of model parameter and the control of simulation process.

In order to identify the function and boundary of encapsulation, we define simulation engine agent SimEngineAgent=<Model, Input, Output, Parameter, Operation, Configuration >, where:

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

B. Requirement for Encapsulation

We can wrap simulation engine according to above definition under the condition that simulation software and its disciplinary model must meet certain requirements.

Simulation software must provide secondary development interfaces, which enable operations on disciplinary model and control of simulation process.

For the disciplinary model:

- Provides explicit input and output, which are named according to some rules, and to elaborate where the variable comes, goes, and the description of variable function.
- For internal variable and state variable, simulation software based on the disciplinary model must be able to provide the current simulation result automatically, which is to be used as the initial state of the next simulation period.

IV. ENGINE ENCAPSULATION OF ABAQUS

The forging and manipulating process of Forging manipulator system is involved in different domain models, including control, hydraulics, mechanics, deformation of workpiece and so on in which the deformation subsystem is established by commercial FEA software. However, the research on HLA adaptor is mainly focused on the redevelopment of the domain software Matlab/Simulink, which has already achieved much progress. By contrast, there is little work on the equally important FEA software.
ABAQUS is one of the most advanced general nonlinear FEA software. It is competent for the static and dynamic analysis of complex structure, tackling huge problem as well as emulating the highly nonlinear influence between structure and material. As a result, it is preferred to work on the techniques for wrapping ABAQUS simulation engine.

In this section, a technique of wrapping ABAQUS engine is proposed to meet the requirement of distributed simulation. The main idea is to develop a simulation agent of disciplinary model under distributed environment, and utilize the secondary development interfaces provided by ABAQUS to fulfill the encapsulation.

A. Introduction of Secondary Development

The approaches to redeveloping ABAQUS are as follows:

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

The script interface of ABAQUS is a programming library based on object, embedded with object-oriented programming language python, and provides a set of APIs, which are used to control ABAQUS/CAE and realize modelling and post process. Abaqus extends Python with approximately 500 additional objects, and there are many relationships among these objects. In general terms the Abaqus object model is divided into the Session, the Mdb, and the Odb objects, as shown in Fig. 2 [5].

![Figure 2. The Abaqus object model](image)

Every object has its corresponding data member and method. After the object is created, one can use provided method to process the data member.

Here, the third way is adopted, which is to control ABAQUS kernel to achieve automatically analysis and post process through python script.

B. Operation Fulfillment

- SetInput(parameter){ Modify InputFile }, where the modification of .inp file should conform to the format and specification of .inp file, and the time parameter also can be altered by such an operation.
- GetOutput(parameter){ Exec python script }, that is to run python script file wrote by user self to access the data in result database .odb file.

Data computed by ABAQUS is stored in the result database in object-oriented form. To access the result data, the path where the data is stored must be found in the first place, and then the data can be accessed by referring to the corresponding variable or through calling the methods of the data object. Sometimes it is necessary to read data from .dat and .fil files, and the text processing function provided by python will be used.

The realization of these two operations above demands in-depth communication between the developer and the modeller. The developer must know sufficiently well about the .inp file or the original .cae model and be able to identify which hierarchy the variable belongs to (refer to Fig. 2).

Task submit, suspend, resume, terminate etc, all of which can be realized by executing ABAQUS internal command. You can execute abaqus command through this statement{ system(cmd); } in VC++.

- Task submit: call abaqus job= job-name
- Task suspend: abaqus suspend job=job-name
- Task resume: abaqus resume job=job-name
- Task terminate: abaqus terminate job=job-name

C. ABAQUS Engine Encapsulation Paradigm

Adopting the secondary development technique aforementioned can satisfy the requirement of model agent developing for distributed simulations, and the paradigm is shown as Fig. 3.

![Figure 3. ABAQUS Engine encapsulation paradigm](image)

In Fig.3, the client application program is used as the schedule engine to call the ABAQUS engine agent to complete the standard API operation. ABAQUS engine agent is to call the background ABAQUS engine through the execution of ABAQUS internal command. ABAQUS engine is to complete the operation on data in workspace through calling the user subroutine.
not receive any timestamp-order (TSO) messages with a timestamp less than its current logical time. Consequently federates can only request RTI for the advancement of their logical times. This time advancement is not allowed until RTI explicitly grants it. As a result, the mechanism for advancing the logical time is the core of the HLA Time Management services.

There are three types of time advancement mechanism supported in HLA: event-driven, time-stepped and the optimistic time advancement. In the collaborative simulation of complex systems, time-stepped time advance is the dominant mechanism. First, the interaction of data is of major concern in such simulations while internal events rarely happen. Secondly, the internal simulation time-step is usually very small, and hence it is possible to get the same amount of results in each step of federate time which will improve the precision of simulation.

The time advancement algorithm based on time-stepped is shown as Fig. 4.

- **Step 1**: initialize federate, set time management strategy, set current simulation logic time \( T_c = 0 \), final simulation logic time \( T_f \), federate simulation time step \( T_s \), federate lookhead \( T_{ld} \);
- **Step 2**: if current simulation logic time equals \( T_f \), then turn to Step 7, else get required data from the simulation bus;
- **Step 3**: wait a moment for the HLA events;
- **Step 4**: request time advance to RTI for a time step \( T_s \);
- **Step 5**: process all the HLA events until receive the callback function timeAdvanceGrant;
- **Step 6**: simulation runs to \( (T_c + T_s) \) and set current local federate logic time \( T_c = T_c + T_s \);
- **Step 7**: update local federate information to RTI, save required data to data file, return to Step 2 and begin the next loop;
- **Step 8**: resign federation, release related resource, and then terminate simulation.

Figure 4. time advancement algorithm based on time-stepped

VI. PROTOTYPE IMPLEMENTION

A. Experiment Scenario

The horizontal passive submissive action of forging manipulator is of great importance for relieving reaction force caused by workpiece deforming. Otherwise, it will result in the key components of manipulator being overloaded or even worse accidents such as the tipping over of the equipment.

When the reaction force reaches the given valve value of the cushion dashpot, it unlocks, and clamps moves backward. In the meantime, manipulator performs horizontal passive cushioning movement. The movement of clamps is dependent on reaction force, and simultaneously the reaction force is influenced by the movement of clamps. In order to consider the coupling between reaction force and clamps movement, it is necessary to implement the collaboration simulation.

The federation model consists of several domains principally, including the mechanical multi-body dynamics model, the control model and the hydraulic servo model of the manipulator, the deformation model of workpiece. Here, the deformation model is created using ABAQUS while the rest of domain models are created using the same software Matlab/Simulink, and they are finally connected to create a complete model. As a result, the interaction is mainly between Matlab/Simulink model and the ABAQUS model. The interaction relation model is shown in Fig. 5.

![Interaction relation model](image)

Figure 5. Interaction relation model

With the aforementioned HLA adaptor and its responding model transformation method, we wrap the Matlab model and the Abaqus model, making them conform to HLA standard and specification. More specifically, the wrapping of ABAQUS model adopts the proposed simulation engine encapsulation technique in this paper. After the transformation, each disciplinary model could join federation directly on our HLA/RTI platform as a federate.

B. Experimental Results

Fig. 6 shows the simulation result of the force load that the forging workpiece acted on manipulator in this process (the effect of forging process is assumed to be 1 second).

![Horizontal load acting on manipulator](image)

Figure 6. The horizontal load acted on manipulator between the single discipline and the multidiscipline simulation

Compared to the peak value of horizontal load in the single discipline experiment, the value on this point of multidiscipline experiment decreases significantly, and it results from the passive submissive action of manipulator. Such an action could reduce the load largely acted on the manipulator, and therefore guarantee the protection of the manipulator. Collaborative simulation experiment reflects on the spot’s real working condition: cushion dashpot valves are forced to unlock twice so as to reduce the horizontal load on manipulator.
It can be seen that this experiment verifies the high fidelity of the multidisciplinary collaborative simulation and the validity of the proposed simulation engine encapsulation technique in this paper.

VII. CONCLUSIONS

The collaborative simulation techniques were studied in this paper with focus on complex systems such as forging and manipulator systems. The HLA adaptor was employed to transform the models created through commercial software to HLA federate following the specification of HLA in order to improve the reusability of models and the expandability of our platform. In order to solve the difficulty of wrapping ABAQUS software, an ABAQUS engine encapsulation technique was proposed, which facilitated the integration of ABAQUS software with HLA-based collaborative simulation platform. Subsequently, a time-stepped time-advance way is introduced. Finally, a prototype system was presented. By comparing the simulation results between single discipline and multi-discipline platforms, the high fidelity of the collaborative simulation and the validity of our simulation platform were empirically verified.

One of the major directions for future work is to improve the performance of the multi-disciplinary collaborative simulation platform. In the meantime, the HLA-based platform supporting WAN (Wide Area Network) simulation can be established in combination with SOA (Service-Oriented Architecture) techniques.

ACKNOWLEDGMENT

This work was granted by National Basic Research Development Program (973 Program) of China (No. 2006CB705407) and National R&D High-Tech Plan (863 Program) of China (No. 2007AA04Z150).

REFERENCES