Design and Implementation of Centralized and Distributed Self-Adaption of Simulation System Framework

Jianxing GONG, Jiangtao KONG, Jianguo HAO, Jian HUANG

College of Mechatronics Engineering and Automation, National University of Defense Technology, Changsha 410073, China

Abstract — Firstly, the integration techniques of simulation systems, such as component-based, middleware-based and bridge-design mode and so on, have been introduced in this paper. Then, based on these, a common simulation system framework has been designed on the view of software architecture, which enables the simulation system based on the simulation system framework to run on the distribution mode and the centralized mode. Moreover, based on the two running mode, the simulation system could use the same modeling specification, the unified protocol of data interoperation and integrative simulation running infrastructure. Lastly, according to the above design idea, this thesis has put forward the implementation process of key software modules, application modes and the work flow, which simulation model components and simulation running infrastructure based on the simulation system framework would meet the requirement of self-adaption of distribution mode and the centralized mode so that it has improved the reusing and interoperability of resources of simulation system.

Keywords - design mode based on bridge; simulation system framework; centralized and distributed self-adaption; simulation model component; simulation running infrastructure

I. INTRODUCTION

Reusability and interoperability have always been one of the basic principles of designs of simulation system. How to max the reusability of simulation system, improve the developing efficiency of the system and expand the field of application are one of needs of simulation system’s reconstruction at present and in the future. Simulation system framework is defined as a kind of simulation system or semi-reusable design which represented by a set of reusable simulation models and correlations of those models or simulation system frame that can be instantiated by system developers. Therefore, simulation system framework is a partial implemented simulation software system structure and a technology that can reuse simulation system and implementation.

Nowadays, HLA[1] (High Level Architecture) has been widely used in simulation system design and development in distributed simulation domain and has been the standard of distributed simulation development. The framework, rules, OMT (Object Model Template) and federate interface provided by HLA ensure the reusability and interoperability of distributed simulation system as far as possible. Thus, the former developing incompatible HLA simulation system (such as, JSIMS, JMAS, OneSAF) generally use centralized running mode and provide the modules to connect with HLA which can be implemented in high-speed using the integration and use the interoperation with distributed simulation system based on HLA and non-HLA system. Because this centralized simulation systems only draw support from HLA, not their own characteristics and can’t compatible with HLA completely, they can hardly get best reusability and interoperability in distributed running mode, for example, they have own standard of modeling, protocol of data interoperability and running platform.

Design and implementation of a universal simulation system framework is proposed in this paper which has great compatibility in distributed and centralized simulation running mode and use consistent standard of modeling, unified protocol of data interoperability and integrative running platform. This simulation system framework allows users to select different running modes based on their needs, adapt the centralized or distributed mode automatically and implement the high reusability and interoperability of simulation system resources.

II. RELATED TECHNOLOGY OF SIMULATION SYSTEM INTEGRATION

Component technology [2][3]has been one of the popular and mature technology in most simulation system designs which integrates simulation systems in high speed using the combination and reusability of component modules. Middleware technology[4][5]detaches the universal and dedicated modules efficiently, reduce the coupling degree between universal modules in low level and application modules in high level and improve the reusability and adaptability of simulation system. Bridge-Design method enables detach the abstract interface and implementation efficiently, the basic structure of which is free form changes of implementation because of changes of application requirements and improve the adaptability and robustness of simulation system. In this paper, these three methods are the basic technology of simulation system framework and integrate simulation systems based on above framework fast and conveniently.
A. Integrated technology base on component

Component technology has been widely used in many industries and also in simulation field. Because of different limitations of application requirements and field characteristics, component technology doesn't have a united specification and standard. Thus, BOM[6] (Base Object Models) is used in this paper which is also an object model [7]and a model description standard for composition models based on the object models of HLA. With the purpose of providing a kind of mechanism promoting interoperability, reusability and composability, BOM specification raised by SISO (Simulation Interoperability Standards Organization) helps to implement the composability based on components in the field of modeling and simulation. Regarding the interoperability, reusability and composability as its core, BOM has the idea of DIY (Do it by yourself) through combining reusable simulation resources into customized simulation system according to requirements. With the standard specification describing component-oriented models, BOM provides an open architecture for extendable, composable and interoperable simulation systems.

Figure 1. BOM Architecture

As illustrated in Fig. 1, BOM architecture [8] is composed of three levels. The top level is the view of federates and also a view of simulation system design, that is federates structuring federation in the distributed simulation field based on HLA. This view shows that simulation system is composed of one or several simulation models described by reusable BCI (BOM Component Implementation). In the concept interface level, IF-BOM (Interface BOM) describes the simulation model component's ability of data exchange and interrelationship with external though one or several related object class or interactive class.

Design of simulation system as shown in Fig. 2, firstly, simulation system assembles corresponding ECAP BOM (Encapsulation BOM) according practical needs. Secondly, according to different mapping rules based on the idea of MDA, PIM [9] (Platform Independent Model) ECAP BOM models are transformed into PSM (Platform Specific Model) BCI models. At last, the interface assembly and combination of functions are implemented. This design can integrate simulation systems rapidly and improve the self-adaptation ability of the system.

Figure 2. BOM-Based Integration of Simulation System

B. Intergradation Technology Based on Middleware

As shown in Fig. 3, infrastructure of simulation system mainly includes two parts: simulation system middleware and services (usually provided by simulation runtime platform, such as RTI [10]) . The middleware is a narrow-sense software middleware, acting as a bridge between application system and simulation system services. This helps to isolate the realization of runtime services and the detailed application system, which enables the simulation application software developed based on this structure running in a distributed and centralized runtime services. Thus the infrastructure realizes the cross-platform transplantation of simulation model components and simulation system under the circumstances of little changes or even no compilation.

Figure 3. Structure of Simulation System Base on Middleware

C. Intergradation Technology based on Bridge-Design Mode

The simulation service in infrastructure of simulation includes simulation service abstract interface and implementation two parts. Simulation service abstract interface is provided for users and supplies necessary data for simulation service, according to which, the call of simulation service can be completed. Connected with the simulation service by a simulation adaptor, the same simulation service
abstract interface can have multiple simulation service realizations.

Figure 4. Implementation of Simulation Services Base on Bridge-Design Mode

As shown in Fig. 4, in the implementation structure, the left is abstract interface of simulation service. The right is implementation of simulation service and the adapter is in the middle. While the abstract structure remains the same, the bridge-design mode adapts simulation services in different simulation platform wonderfully.

III. DESIGN OF SIMULATION SYSTEM FRAMEWORK

Generally, simulation system framework is a universal module in simulation system. As a specialized module, the simulation model component designs a favorable soft framework for supplying itself an efficient data interoperation mechanism.

A. Design of software framework

In this section, the data layer and logical layer of simulation system soft framework are discussed.

1) Structure of Data Level

As illustrated by Fig. 5, the simulation system framework is very data centric. And flowing of data drives the running of the whole framework. Thus, according the different needs of data, the structure of simulation system framework is divided into four main layers: data providing layer, data manager layer, data processing layer and data communication layer.

Each layer can only calls the services provided by contiguous layers. The top layer is simulation model components related to applications which use and provide data. In order to guarantee the independence of components, they cannot communicate directly and the distribution, filtration and cache of data are completed by data manager layer. After data processing layer, the components are transferred into prescriptive simulation events. Then, simulation events will call corresponding simulation services. At last, the data communication layer will implement the data interoperation of internal components and other components of simulation system in distributed running mode which makes the interoperation between components more efficient.

2) Logical structure

As shown in Fig. 6, a simulation system is abstracted and divided into changeable and unchangeable parts. On the application layer of the simulation system which is user-oriented, the changeable part is mainly composed of different model components with different functions and other application components. The paper defines the unchangeable part as a simulation system framework, controlling the data filtration, cache and distribution of the simulation system. Besides, it also dispatches simulation model components based on the BCI, drives the harmonious advancement of model components in the simulation system and invokes simulation services module provided by simulation system framework through the simulation adapter.

The simulation system framework will not couple with any specific simulation model components and is also independent of detail simulation application. It is customized according to user’s requires. This framework is not a specific simulation application system, but a general framework for constructing different kinds of simulation system through combination of model components.

B. Data Interoperability of Simulation Model Components

The Fig. 7 is interoperability implementation approach based on the simulation model component of BOM, the I/O data channel of simulation system framework is the inner data channel of simulation system (in distributed system, it means federate) and the data interactive way among simulation model component. The framework does not use the broadcast way of distributing data produced by model components, while the data filtering net regarding IF BOM
as its components used in data regional filtering mechanism. The simulation model components are mainly composed of two parts: BOM implementation and BOM interface, which depicts the interaction of model components and the ability of the interaction among simulation model components and between simulation model components and the outer environment respectively. Through the BOM information provided by each simulation model component, the simulation scheduler can get all the publishing and subscribing information of model components in the whole framework and filter the data within federates effectively, reduce redundant data transmission and improve the transmission ability of I/O data line in framework.

![Figure 7. Interoperation among Simulation Model Components](image)

C. Simulation Scheduling Algorithm

The scheduling algorithm of the simulation system framework is actually based on the method of BL [11] protocol which first defines the unique lookahead[12] for every simulation model components to ensure the visibility of all those TSO events in the future. The safe execution time window, i.e. LBTSmin, can be computed by lookahead. The simulation model components inside the time window are permitted to advance based on the priority to determine the calling order. The specific scheduling process is shown in Alg.1.

![Alg.1 Simulation Scheduling](image)

```plaintext
Known: time lookahead of each simulation model component (LPi.lookahead) and LPi.requested_Time, IB (system framework of inner cache area), IE(inner event), EE (external event).
1. LBTSmin=\min\{LPi, LPi.lookahead+LPi.requested_Time\} IB (system framework of inner cache area), IE(inner event).
2. L←\{LP | \forall LPi, LPi.requested_Time ≤ LBTSmin\}
3. SortByPriority(L)
4. while(L≠∅) {
5. LPi.lookahead←head(L)
6. ELRO←ELRO ∪ CheckOutSimEventOFRO(LPi)
7. ELTSO←\{e | e∈ELTSO, e.timestamp≤LBTSmin\} ∪ CheckOutSimEventOFTSO(LPi)
8. LPi.local_Time←LPi.requested_Time
9. LPi.state←update
10.}
11. while(ELRO≠∅) {
12. ELRO←ELRO-{e}
13. {IE,EE}←execute(e)
14. IB←IB ∪ IE
15. Send(EE)
16.}
17. SortByTimestamp(ELTSO)
18. while(ELTSO≠∅) {
19. e←head(ELTSO)
20. {IE,EE}←execute(e)
21. IB←IB ∪ IE
22. Send(EE)
23.}
```

IV. THE REALIZATION OF SIMULATION SYSTEM FRAMEWORK

The above discussion is described mainly from the angle of architecture design of simulation system framework. This section will described the core module of simulation system framework, the application model of framework and the processing of key module from the implementation perspective.

A. key modules of realization

As shown in Fig. 8, the simulation system framework mainly includes: simulation execution control module, component management module, data distribution module, simulation service module and network service module.

1) Simulation Execution Control Module

This module is used to provide the human-computer interaction interface, control the running state of simulation such as starting, pausing, continuing and ending and so on, and analyze the configuration information of the system, controlling commands of simulation. Moreover, through simulation service module, it establishes data interaction table among components, as well as the table between system and external federate in the distributed running mode. Under the centralized running mode, the simulation execution control module loads the data distribution module and the component management module, inputs the script data for data distribution module and the component initial information of system configuration for component management module respectively.

2) Simulation Model Component Management Module

The simulation model component management module includes the component combination management unit, the component scheduling management unit and the component event management unit. The component combination management unit obtains the system configuration information parsed by the execution control module, checks the completeness of component resources of simulation models, and loads components composed to build the simulation system. The component scheduling management unit stores the component instances created by the combination management unit through the scheduling
manager and manages the scheduling states of components during the simulation running. The component event management unit manages the simulation events generated in the process of scheduling management unit and calls the simulation service module to realize data interaction among model components.

3) Data Distribution Module
This module is used to parse the input simulation script data file under the centralized running mode, obtain the information of simulation entities and their corresponding components. In the process of simulation running according to the time of its advancement, this module outputs temporal system configuration files, sends corresponding script data and calls the service module to distribute script data to model components by simulation events.

4) Simulation Service Module
This module is used to provide model components with simulation service set, realize data publishing and subscribing among components. Moreover it establishes data interaction table among components as well as table between system and external federates in the distributed running mode, maintains the registration and remove of model entities within system and coordinates time advancement of the runtime simulation system.

5) Network Service Module
The network service module is used for the transformation among the RTI serves and the framework simulation services under the distributed running mode. It can separate model components and RTI, and ensures the data interaction between system’s runtime infrastructure and external federates.

B. Application mode
By the simulation system framework, it can make simulation system auto-adapt the operating requirement of the two modes which is distributed application mode and centralized application mode without modification even compiled code.

1) Distributed Application Mode
As shown in Fig. 9, the runtime infrastructure of simulation system reads the specified configuration files of simulation system through human-machine interaction unit provided by simulation execution module in the distributed running mode. And those files are parsed by the configuration analysis unit. The result of the parse information is used as the input of the component management module. According to the information, component management module loads simulation model components (SCMs) in specified locations to construct of simulation system. Through invoking the RTI services, successfully established system would be added to the Because the runtime infrastructure and specified version of RTI are separated. HLA distributed simulation system as a federate in compliance with HLA specification and realizes interoperability with other federates.

2) Centralized Application Mode
As shown in Fig. 10, under the centralized running mode, the runtime infrastructure of simulation system reads a simulation script data file through human-machine interaction unit provided by simulation execution control module. Firstly, data files are converted into system configuration files by data parse unit of simulation execution control module. Then the script data is embedded into the system configuration file as an independent information node. Just as the distributed running mode, component management module loads components required by the construction of simulation system according to the system configuration information. While different
from distributed running mode, the infrastructure would not need the network service module anymore but as an independent-running simulation system.

C. Processing of Modules

The processing of core modules of the simulation framework is as follow.

1) Processing of Simulation Model Component Management Module

As shown in Fig. 11(a), SMC management module constitutes the configuration information from simulation module, auto loading and combining SMCs to construct the simulation system. Through the BCI provided by SMC to build data interaction among SMCs, and then waiting for the simulation command from simulation execution control module. SMC management module maintains SMC schedule list, according to the schedule list coordinates scheduling sequence of simulation model component.

As shown in Fig. 12(a), simulation service module constitutes configuration information based simulation system and build interactive map data between the inner SMCs and the outer simulation system, it will monitor the simulation events when simulation operates. The simulation service module parses simulation events, obtain the event type and invoking the corresponding simulation service once monitoring. If the running mode is the distributed application mode, it delivers the simulation event to the simulation service module, otherwise, waiting for new simulation event.

4) Processing of Network Service Module

5 CONCLUSIONS

The distributed and centralized-integrative Runtime Infrastructure of Simulation System proposed in this paper could load component-based models according to specific requirements of simulation application system. Through composing model components, it not only improves the developing efficiency of simulation system, but also enhances the reusability of models, thus realizing flexibility of structure and customization of functions for simulation system. Besides, the separation between models and implementation details of specific runtime infrastructure of simulation system helps models to be applied in large-scale distributed simulation system as well as small-scale centralized system, which enhance the adaptability of models.
By using the component technology and data publishing and subscribing mechanism, the coupling degree between models has been reduced, which improves the parallelism of simulation models. Achieving parallelism of model calculation, simulation scheduling [13], time management in parallel and distributed real-time simulations [14] and simulation cloning [15] is one of the research hotspots in simulation application area, which is also the further work of this paper.

ACKNOWLEDGMENT

This work was supported by the National Nature Science Foundation Grant, funded by the Government of the People’s Republic of China (Project No.61074108). The authors would like to thank an anonymous referee for their constructive comments.

REFERENCES