An Introduction of BOM Modeling Framework

Qiang He, Ming-xin Zhang, and Jian-xing Gong

***Abstract*—Component based modeling has been a research hotpot in the area of Modeling and Simulation for a long time. In order to increase the reusability and interoperability of models, this paper put forward a component based modeling framework called BOM Modeling Framework (BMF) based on BOM specification. The structure, principles, formalisms and behaviors of BMF were presented in detail first. The description and framework of Atomic BOM model and coupled BOM model in BMF were studied in succession. At last, the developing process of BMF models was introduced. The prototype developed showed that BMF could not only promote the reusability and interoperability of models but also improve the efficiency and easy the process of M&S.**

***Index Terms*--Basic Object Model; BOM Modeling Framework; Atomic BOM Model; Coupled BOM Model**

I. Introduction

To simplify model development and further promote the reusability and interoperability of models, SISO (Simulation Interoperability Standards Organization) put forward the BOM (Base Object Model) specification [1][2] which was proposed as a SISO standard in 2006. The BOM concept is based on the assumption that piece-parts of models, simulations, and federations can be extracted and reused as modeling building-blocks or components. The interplay within a simulation or federation can also be captured and characterized in the form of reusable patterns. These patterns of interplay are sequences of events between simulation elements. The representation of the pattern of interplay is captured in the BOM. BOM has an obvious brand of High Level Architecture (HLA) [3][4]. The origin of BOM is to overcome the shortcoming that HLA federates are highly coupled with the used Federation Object Model (FOM) of federation which makes federates hard to reuse and interoperate directly. If federation's FOM changes, the federates can't be reused anymore and need lots of modification along with FOM. BOM provides an important component modeling mechanism for M&S. And it is considered as a key technology for the next-generation simulation, for that it can enhance standardization, encapsulation, composability, interoperability and reusability etc.

BOM modeling framework (BMF) is a component based modeling framework and it is a little different from others in three aspects: model description, model connection and implementation. The description document is written in XML and conforms to all terms and schemas of BOM specification. The connection between models supports not only directional mode but also undirectional. Directional connection is the same as others that name or ID are using for message passing and matching. Undirectional connection adopts declare management introduced by HLA, which uses publishing and subscribing mechanisms to announce that the ability to generate or receive which kind of message. And the publishing and subscribing can change dynamically. The model employed by BMF is called BOM model, and it uses dynamic load library (DLL) as carrier which can be used without compiling and linking. For great influence of HLA, BOM model only executes on Run-time Infrastructure (RTI) now. But the demand of different engine has been considered, the model can be transplanted to other simulation environments easily.

The remainder of this paper is organized as follows. In section 2 we outline structure, principles, formalisms and behaviors of BMF. In section 3 we study the description and framework of Atomic BOM model and coupled BOM model. In section 4 we present the developing process of BMF models. Section 5 concludes with a discussion and some perspectives.

II. Basics of BOM modeling Framework

A. Structure of BOM modeling and simulation

BOM models bear an analogy with building blocks and they can be executed in a plug and play mode. From the perspective of modeling, BMF can be divided into four layers: atomic model, coupled model, federate and federation. (Federate and federation are terms of HLA. A federate is a relative independent part of simulation system and it can run separately usually. Federation corresponds with the whole system). Fig 1 shows a simulation federation including two federates. And each federate is made up of a coupled model which consists of three atomic models. The example in Fig 1 shows the fact that the same model (atomic model A) can be reused in different federates. Atomic model is the only element needs to be coded by developer, and it is the foundation of the whole system. Several atomic models can compose into a coupled model if necessary. A federate may be composed of at least one atomic model or coupled model.

BMF needs three running establishments to support simulation, including model framework, extensible simulation running framework [5] (called XSRFrame for short) and RTI. The interfaces of atomic model and coupled

Manuscript received September 9, 2011, revised September 13, 2011. This work was supported by the National Natural Science Foundation of China under Grant NO.61074108.

Qiang He is a Ph.D. student in the College of Mechatronical Engineering and Automation at National University of Defense Technology, Changsha, P. R. China (e-mail: heqiang@nudt.edu.cn).

Ming-xin Zhang is Ph.D. student in the Faculty of technology, policy and management at Delft University of Technology, Delft, the Netherlands (e-mail: zh.mingxin@gmail.com).

Jian-xing Gong is a lecturer in the College of Mechatronical Engineering and Automation at National University of Defense Technology, Changsha, China (e-mail: fj_gjx@yahoo.com).

model manager assume the function of model framework. They ensure that models can be scheduled by engine and interact with each other. XSRFrame is the engine that runs BOM models. Actually XSRFrame is a general purpose HLA compatible federate and it can accommodate BOM models. XSRFrame provides a few services (such as FM, OM, DM, TM and DDM) to make BOM models execute like traditional federates. XSRFrame can afford sufficient supports for simulation if executing a federate alone. When more than one federate applied, RTI is necessary. It connects all federates to a federation and makes them can communicate and synchronize.

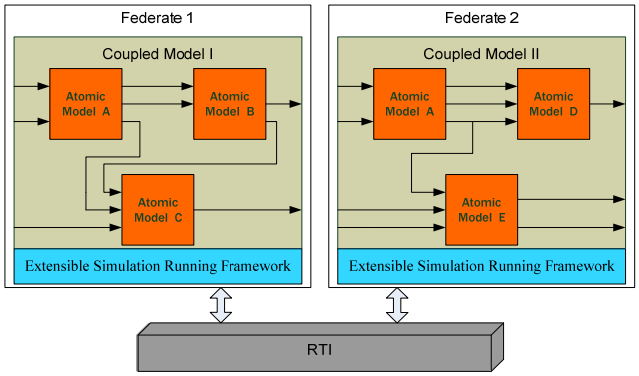

Fig. 1. Structure of BOM modeling and simulation

B. Principles of BOM modeling framework

Four important interoperability principles have emerged from the development of HLA [6]:

1. A standardized software framework with well-defined interfaces is required to interconnect reusable models (e.g., the RTI).

2. The data exchanged within the models must follow an agreed upon standard (e.g., the FOM).

3. Distributed object technology allows models to (1) know about each other's state and (2) invoke actions within other models in a coordinated manner (e.g., TM, OM, DM, DDM, and OWN).

4. The double-abstraction barrier principle allows a model to invoke actions on other models while hiding the details concerning which specific models are participating in the action and which methods those participating models provide to handle the action (e.g., Interactions).

BMF adheres to these rules and expands other principles to simplify and standardize the process of M&S, which are:

1. Separation between models and engine. This builds on the observation that models always change slowly and engine innovates frequently. And then it's necessary to decouple with engine for an outstanding model framework. In BMF a model is divided into two parts: kernel model (KM) and connected model (CM). KM carries out model's actions and it doesn't use any interfaces of the engine directly. CM is a middleware used to connect KM and the engine. This design makes it easier to develop and maintain models, along with a more robust technical framework.

2. Separation between interface and implementation. Any model must provide an abstract interface in which various operations are defined. Concrete one must inherit and implement it. User-defined models never directly invoke methods of other user-defined models. Models must never depend on specific implementations of other models. This formalism also supports dynamically swapping different model representations in and out of the execution environment.

3. Separation between behavior and data. The parameters of models should not be fixed, and they should be configured from external not only at the initial state of simulation but also in running process.

4. Independent standard description documents. There are three kinds of documents used: kernel model document, model mapping document and BOM component document. Each document provides a different aspect of model used by different users and all of them are written by well-defined XML.

5. Identical resource structure. All models must keep agreed formula, for example file naming, directory arraying etc. By this a repository of models can be built easily, and it will provide a rapid way for model finding and using.

Compared to other standards, BMF has two preeminent advantages.

- Increasing the possibility of reuse

BOM model has two forms of reuse. First, the whole model can be used in other systems if the same engine employed. Second, KM also can be reused in different engine. Because it doesn't interact with engine directly, CM can be regenerated according to new platform and this does not require modification of KM.

- Reducing the difficulty of model development while improving the efficiency

Model development follows a very strict programming criterion, and most codes are created by auto-generating tool which can avoid many fallible errors. In addition, BMF makes developers only pay attention to logics of models and needn't care the change of simulation platform.

C. Formalisms of BOM model

1) Atomic BOM model

Atomic BOM model models the behavior of simulation entity, and it can be depicted by parallel DEVS [8]:

$$M = < S, X, Y, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda, ta >$$

where

S is the set of states

X is the set of input events, and $X = \{SubAttr\} \cup \{SubInter\}$. SubAttr denotes subscribe attributes of object classes and SubInter is subscribe interactions.

Y is the set of output events, and $Y = \{PubAttr\} \cup \{PubInter\}$ PubAttr denotes publish attributes of object classes and PubInter is publish interactions.

$\delta_{\text{int}} : S \to S$ is the internal transition function

$\delta_{ext} : Q \times X^b \to S$ is the external transition function, where X^b is a set of bags over elements in X, where $\delta_{ext} = \{\delta_{subattr}\} \cup \{\delta_{subinter}\}$. $\delta_{subattr}$ denotes subscribe object class process function, and $\delta_{subinter}$ is subscribe object class process function.

$\delta_{con} : S \times X^b \to S$ is the confluent transition function,

$\lambda : S \to Y^b$ is the output function, where Y^b is a set of output events combination and $\lambda = \{\lambda_{PubAttr}\} \cup \{\lambda_{PubInter}\}$. $\lambda_{PubAttr}$ functions are called automatically when the states of model changed, but $\lambda_{PubInter}$ function is called by need.

$ta : S \to R_0^+$ is the time advance function.

2) Coupled BOM model

Coupled BOM model is a set of relations among models that can be atomic model or coupled model. The formalism of Coupled BOM model is:

$$N =< X, Y, D, \{M_d\}, \{I_d\}, \{Z_{i,d}\} >$$

where

X is a set of input events

Y is a set of output events

D is a set of component references

For each $d \in D$, M_d is a BOM model

For each $d \in D \cup \{N\}$, I_d is the influencer set of: $I_d \subseteq D \cup \{N\}, d \notin I_d$, and for each $i \in I_d$, $Z_{i,d}$ is a function, the *i*-to-*d* output translation with

$$Z_{i,d} : X \to X_d, \text{ if } i = N;$$

$$Z_{i,d} : Y_i \to Y, \quad \text{if } d = N;$$

$$Z_{i,d} : Y_i \to X_d, \quad \text{if } d \neq N \text{ and } i \neq N;$$

The three functions above are named: EIC, EOC and IC.

BOM model is an extension of DEVS. It establishes input/output events, internal transition function, external transition function and output function detailed. These accessorial definitions can guide modeling better and make models more rigorous.

Component based modeling means that the simulation models reach an agreement on interface mapping, event type, parameter type and time scale according to relevant syntax description, attain the ability to send event and response the request, and correspond with each other finally. The essential of component based modeling is matching and mapping between different model interfaces. In the description of coupled BOM model, the interface set of coupled model (F) is composed of the interface sets of sub-model (C). The relationship between the two interface set is showed as follows.

$$F=C_0 \cup C_1 \cup \cdots \cup C_n,\ n = 0, \cdots, N$$

The description of coupled BOM model contains information describing the ability to interact throughout the coupled model. And it's the common interface for all sub-models to interact with others out of the coupled. Building mapping connection between interfaces of sub-models and coupled model is better than piling up all the sub-model interfaces. The relationship is showed as a formula.

$$MAP: C_i \in C_n \to f_m \in F$$

In this formula, i=0,⋯,n ； n=0,⋯,N ； m=0,⋯,M.

C_i is one element in the interface set of model C_n. MAP means mapping one interface C_i of a model to any interface f_m of coupled model. This mapping mechanism allows multi-to-one connection which means different model interface could map to the same interface.

D. Behaviors of BOM Model

1) Atomic BOM Model

The behavior of atomic BOM Model conforms to algorithm as algorithm 1. The *ta* function generates a time advance request event *TimeAdvEvent* which has a timestamp specified next time, and this function will be called in initialize, δ_{int} and $\delta_{ext.}$ When an advance request approved, the engine will schedule a time advance grant event called *GrantAdvEvent* which has the same timestamp of *TimeAdvEvent* and execute this event. When received *GrantAdvEvent*, the model sets current time first, then executes δ_{int} to update states, and activates λ for output events. When received external events, the model calls δ_{ext} to process these events and schedules a *TimeAdvEvent* for self if necessary. Moreover, the model can use other entry point functions such as changing data connection among other models and creating or deleting date filtering region when running.

$t_c \leftarrow$ *current time*
if receive init – message(*Time t*)
  $t_c = t$
  send TimeAdvEvent
if receive GrantAdvEvent(*Time t*)
  $t_c = t$
  $s \leftarrow \delta_{\text{int}}(s,t)$
  $y \leftarrow \lambda(t)$
  send y – message
  if (*necessary*) *send TimeAdvEvent*
if receive x – message($x \in X$, *Time t*)
  $t_c = t$
  $s \leftarrow \delta_{ext}(s,t,x)$
  if (*necessary*) *send TimeAdvEvent*

Algorithm.1. behavior of atomic BOM model

2) Coupled BOM Model

There are two modes for coupled model execution: as a single object or flatten [9]. In the first approach the coupled joins simulation as one, and the sub-models will not be exposed to engine. The coupled behaviors like atomic and it does not require engine of additional settings, but it needs coupled model assume more functionality. The flatten method removes hierarchy of coupled object, transforms it to a set of atomic models and connections among them. Then the flatten objects will be managed by engine like other atomic ones, and it require engine to do this.

The first is employed in BMF and coupled BOM model works as follows: When a sub-component $d \in D$ generates an output event, it will be passed to all input ports of influencer set I_d by IC and to output port of coupled component by EOC. When the coupled one receives external input, the event will be passed to all input ports of sub-component who requires by EIC.

III. The Design of BOM Modeling Framework

A. The Design of Atomic model

Atomic BOM model contains two parts: description document and model library. Description document is written by XML, and it describes the information of objects,

interactions, interfaces and data types. Model library includes the implementation of model and correlative resources.

1) The description of atomic model

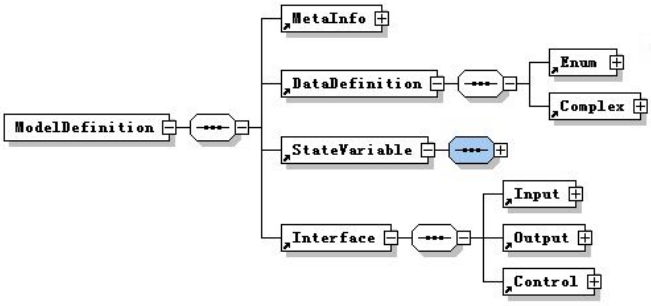

Fig. 2. Schema of kernel model document

The schema of description document is shown in Figure 2. All details of a model are depicted in this document, such as meta-info, data definitions, state variables, and interface. In meta-info section, identifier, model category etc. are specified. There are two types of data definition: enum values and complex structure, and three kinds of functions specified in model interface definition that are classified by input, output and control. Input functions are used to receive external events (subscribe object attributes and interactions); output functions are used to transmit events (publish object attributes and interactions) to other models; and control functions are used to request external services, they can be considered special output functions.

2) The structure of atomic model

As shown in Figure 3, an atomic BOM model is composed of two parts: kernel model (KM) and connected model (CM). KM concentrates on how to construct model's actions with high fidelity and provide the exchange events, but it doesn't care how to transmit them to other models. KM lacks practicable communication mechanism with other models, because its interfaces and data definitions always don't match with others. CM is a delegate which used to provide interoperability for KMs by wrapping them to uniform ones under the constraints of specified engine and application. Then KM can schedule events to other models, receive external events and employ engine's service. But all these do not require themselves to implement, these functions come true via CM. It is a mixture of model proxy and connection middleware which enables KM interoperate with engine. As a result, a separation wall between models and engine has been built. Then KM can get rid of engine's constraints and become completely independent modular.

CM has four primary parts that take effect in the form of class, which are SimObj, SimEntity, SimEntityMgr and EntryPointImplement. SimObj is the top interface of CM, which provides event send and receive functions used to communicate with outside world. SimEntity and SimEntityMgr both are derivate classes of SimObj. SimEntity wraps KM by "has-a" pattern. Every SimEntity object has a reference of KM instance, and this connection can't be changed when created unless simulation terminates. All events scheduled by simulation engine are received by SimEntity first. And then SimEntity extracts useful information from them and calls KM's method to process. So KM needn't understand the event definitions which are interrelated with engine. SimEntityMgr is a management class used to create and delete SimEntity, and it is the only class which can be accessed from outside of CM. When BOM model applied, engine first creates and holds the associated SimEntityMgr object. Then it can create and delete SimEntity instances, and these operations will affect KM finally. EntryPointImplement is a derivate class of KM's entry point interface, in which transmissions of events and requests are carried out. With the help of these setups, engine can manage and schedule models without the knowledge of domain.

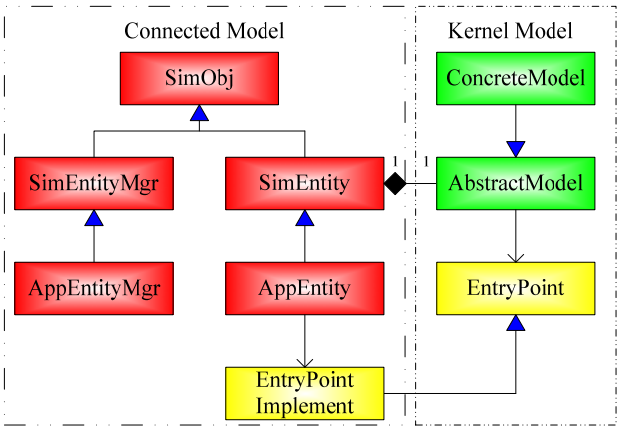

Fig. 3. Structure of CM and relationship between CM and KM

KM is a customized modular by model developer. It is the core of a model, and in which the behaviors and states are maintained. The primary functions of KM's code framework corresponding with the above formalism are shown as Fig 4.

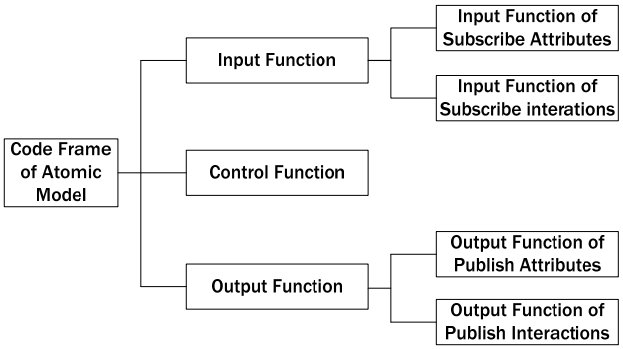

Fig. 4. Structure of KM's Code Framework

B. The Design of Coupled BOM Model

Coupled BOM model is different from atomic in resource organization and execution process. To some extent, coupled model is virtual object for that it looks like a dynamic object created in running process more. Though coupled model has individual resources always, which are assembled by atomic ones ultimately and it needn't code one line. These features make coupled model special from atomic one. Thus coupled model description and coupled model manager comprise a coupled BOM model. Coupled model description depicts its components and their relations. Coupled model manager is the real executor who is in charge of model's whole lifecycle activities such as creation, termination, processing, communication etc.

1) Coupled model description

Besides the basic information, coupled model description mainly contains three sections: (1) Meta information; (2) Compounding Relation; (3) Configuration.

Meta information records the basic information and identification of coupled model, such as type, name, modifying time, ID and the name of coupled model manager. In Compounding Relation section the descriptions of sub-models, their paths and relations are described. Configuration section is composed of the connections of the data flows inside and outside the coupled model, which decides the distribution of messages generating and consuming.

2) Coupled model manager

Coupled model manager includes three parts: (1) Coupled model proxy module; (2) Inner model manage module; (3) Data distribution manage module.

a) Coupled model proxy module

Coupled model proxy module is responsible for corresponding with other modules, and communicating with the other models through engine. This module could access and control the inner models by scheduling event, such as receive interaction, send interaction, update attributes, reflect attributes and advance simulation time etc. Also the proxy is a delegate that all sub-models could be scheduled by engine. It's analogous to a micro simulation engine that runs all objects in coupled model.

b) Inner model manage module

Inner model manage module is in charge of maintaining all object instances of sub-models in their lifecycles, such as creating entities, releasing inactive entities and terminating running ones.

c) Data Distribution manage module

This module manages event distribution among all sub-models. Because BOM model employs publishing and subscribing mechanism to deliver message, and it does not use direct binding that is used more often. Then a message routing table must be implemented, which records connection relationships. With the help of this table a sub-model can schedule events for another model and does not care where it is. If the destination is in the same coupled model, the event will be directly inserted in its input queue. If not, the event will be posted by engine.

IV. The Process of Developing BOM Model

A. Atomic BOM model

The process of developing atomic BOM model is shown in Figure 5. It includes four steps as follows:

a) Editing model description document

First model description document need to edit using the tool naming ModelEditor. With the help of this file, a project containing reference code framework of KM will be generated automatically. In which model interface, state variables, data definitions and an empty implementation of functions are provided. Once the model finalizing the design, these definitions should not be changed.

b) Coding kernel model

When the project of KM created successfully, the developer should complete the functions in corresponding lines. The modelers can modify the framework if necessary, but they must keep the model definition fixed, especially its class structure which is used by CM. After compiled, a DLL is required.

c) Generating connected model

The third step is generating CM and which will create a resource package of whole resources. The tool Connect Model Generator first reads model description document and generates whole sources of CM. In CM sources various formats of input and output data used by KM are transformed to different kinds of events which conform to common definitions, and this ensures KMs can understand each other. If there is no error, the sources will be compiled to another DLL in succession. After these, all resource including description document, DLLs and other correlative files will be packaged in a specified manner. And then an atomic BOM model is completed

d) Testing model

When the model is finished, testing is necessary and a testing tool is provided. This tool can find many errors through inputting some signals and checking their responses. If testing past, the model can be stored in database.

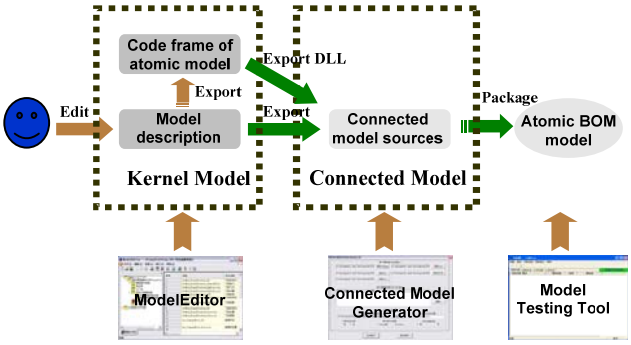

Fig.5. Process of developing atomic BOM Model

B. Coupled BOM model

Composing a coupled BOM model consists of two processes: the assemblage of sub-models' descriptions and resources.

a) Assemblage of descriptions

The description of coupled is configured by modeler on the basis of sub-models manually. As discussed above, the coupled model description comprises of three parts: Meta information, Compounding Relation and Configuration. And the Assemblage is deployed in these three.

First, a new name and ID should be specified for coupled model in Meta information.

Second, all descriptions of sub-models are merged into the coupled one in Compounding Relation. The paths of sub-models and their relationships, like which model is the primary and the whole number of the models should be specified too.

Third, the most important is that the flow and order of messages inside and outside the coupled model is specified in Configuration section. This information aims at message distribution. There are three kinds of configuration. The first is the priority of model processing, it can avoiding simultaneous competition which ensures that the results of different running is the same. The second is the message sending and receiving destination of each model. Two ways are used for this; one is direct connection that the receiving targets are specified by ID, and the other is offered by publishing and subscribing mechanism. The third kind specifies the sending and receiving filters which make message can't be posted or processed until fulfilling some

conditions. Using filter can improve efficiency and decrease network traffic. The filter is registered by interesting expression.

b) Assemblage of resources

Two things happen in this processing.

The first is creating a coupled model manager which manages and maintains both the lifecycles of coupled model and its sub-models. Based on the description and code templates of coupled model manager, BMF can auto generate the sources of coupled model manager. And the sources will be compiled to another DLL. The creation of coupled model manager reveals the idea of black box composing. That is BMF doesn't understand the structure and realization of sub-models and it only needs their interfaces.

The second is packaging the description of coupled model, coupled model manager library and all resources of sub-models into one package. Thus an integrated coupled model is finished and it can be stored in database too.

V. Conclusion

With the development of simulation technology, the requirements for modeling are increasing and traditional way of modeling is out of time. Reusability and interoperability of simulation models are especially important. This paper put forward a component modeling framework called BOM Modeling Framework based on this situation. BMF is built on the basis of BOM specification It supports model editing, code auto generating, testing and component based modeling. From the prototype developed, we found that BMF could promote the reusability and interoperability of models. It decreases difficulty of using and makes M&S models more efficient.

BMF has been used in several applications and we found it could increase the reusability and interoperability of models. Concerning the prospect of component based modeling, this paper is just a beginning and there are a lot of work needs to be done and problems to be solved. Our future work includes:

a) The coupled model composed by BMF is only up to two levels currently and multi-levels composition will be supported later.

b) The state variables and parameters of models are limited of setting at the beginning of execution, but the fact is that models sometimes need to change them in running process. This function will be appended soon.

c) BMF performs poorly when applied distributed mode on RTI, the reason is that RTI is not good at high performance computing and too many conversions are used. In the future a customized parallel simulation engine for BMF models will be implemented.

References

[1] *Base Object Model (BOM) Template Specification*, SISO-STD-003-2006.

[2] *Guide for Base Object Model (BOM) Use and Implementation*, SISO-STD-003.1.

[3] *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) −Frame and Rules*, IEEE Std 1516-2000.

[4] *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) − Federate Interface Specification*, IEEE Std 1516.1-2000.

[5] Gong Jian-xing, "Research on BOM Based Extensible Simulation System Framework", Ph.D dissertation, College of Mechatronical Engineering and Automation, National University of Defense Technology, Changsha, China, 2007

[6] Jeffrey S. Steinman and Douglas R. Hardy, ".Evolution of the Standard Simulation Architeture,"In Proc. The 2004 Command and Control Research Technology Symposium, San Diego, USA, 2004, paper 067

[7] Jeffrey S. Steinman, Craig N. Lammers and Maria E. Valinski, "Composability Requirements for the Open Unified Technical Framework," In Proc. 2009 Fall Simulation Interoperability Workshop, Orlando , USA, 2009, 09F-SIW-022

[8] Bernard P. Zeigler, Herbert Praehofer and Tag Gon Kim, T*heory of Modeling and Simulation-Integrating Discrete Event and Continuous Complex Dynamic Systems*, USA: Academic Press, 2000.

[9] Bin Chen, Hans Vangheluw and Ronghua Zhong, "Integrating Base Object Model Components into DEVS-based Simulation," *Journal of Defense Modeling and Simulation*, vol. 4, pp. 241-256, June 2010.

[10] Yong Peng, "Research on The Test Method of BOM based Simulation Model Comoponent", M.S. thesis , College of Mechatronical Engineering and Automation, National University of Defense Technology, Changsha, China, 2006

[11] Xiaocheng Liu, Qiang He, Ziming Zhong, and Chunguang Peng., "A Rapid Way of Developing BOM-Based-Model," In Proc. 2nd International Conference on Computer modeling and simulation, Sanya, China ,2010, pp.508-512

[12] He Qiang, Peng Yong, Zhang Ming-xin, and Hao Jian-guo, "A Composable Modeling Framework Base-on BOM in War Simulation," In Proc. The 2011 International Conference on Computer Science and Service System, Nanjing, China, 2011, pp.1900-1904

[13] Alexander Verbraeck and Edwin C. Valentin, "Design Guidelines For Simulation Building Blocks," In Proc. The 2008 Winter Simulation Conference, Miami, USA, 2008, pp.923-931.

[14] Perakath Benjamin, Dursun Delen, Richard Mayer and Timothy O.Brien, "A Model-Based Approach for Component Simulation Develepment," In Proc. The 2000 Winter Simulation Conference, Orlando, USA, 2000, pp.1831-1839.

[15] Jeffrey S. Steinman and Jennifer Park, "A Proposed Open System Architecture for Modeling and Simulation," In Proc. The Fall 2007 Simulation Interoperability Workshop, Orlando, USA, 2007, 07F-SIW-044

Qiang He was born in Deyang, Sichuan Province, P.R. China, on February 21, 1982. He is currently a Ph.D. student in the College of Mechatronical Engineering and Automation at National University of Defense Technology (NUDT), Changsha, P.R. China. He received his master degree in Control Science and Engineering from the College of Mechatronical Engineering and Automationin in NUDT, in 2006. His research interests include the technology of parallel discrete event simulation and multi-core computing.

Min-xing Zhang is currently a Ph.D. student in the Faculty of technology, policy and management at Delft University of Technology, Delft, the Netherlands. He received his master degree in Control Science and Engineering from the College of Mechatronical Engineering and Automationin in NUDT, in 2010. His research interests include the technology of discrete event system specification and component based modeling.

Jian-xin Gong is a lecturer in the College of Mechatronical Engineering and Automation at NUDT. He received his Ph.D. degree in Control Science and Engineering from the College of Mechatronical Engineering and Automation in NUDT, in 2007. His research interests include the technology of high level architecture, distributed simulation, simulation architecture and component based modeling.