Formal Description of Simulation Runtime Support Platform Architecture with XYZ/ADL

Sun Li-yang
Nanjing University of Science and Technology
Science and Technology on Information System Engineering Laboratory
CETC 28th, Nanjing, China
Email:li_yang_sun@126.com

Mao Shao-jie
Science and Technology on Information System Engineering Laboratory
CETC 28th, Nanjing, China
Email:mao_shao_jie@126.com

Liu Zhong
Nanjing University of Science and Technology
Nanjing, China
Email:eezliu@mail.njust.edu.cn

***Abstract*— Net-centric simulation runtime support platform (NCS-RSP) provides an environment supporting the construction of community of simulation task (CoST). This paper adopts dual software architecture description framework XYZ/ADL to describe the architecture of NCS-RSP by graphic language and formal language respectively. Then we decompose and refine the core service layer during the construction of CoST. Not only this description method expresses the architecture graphics and behavioral abstraction of NCS-RSP from visual viewpoint, but also validates the correctness and completeness of architecture design from formal view. The research is a new attempt of formal description in military simulation domain and it provides a guideline for the composition and reuse of NCS-RSP service.**

***Index Terms*—net-centric simulation; service composition; runtime support platform architecture; architecture description language**

I INTRODUCTION

Net-Centric simulation (NCS) concept and technology is brought forward to meet the requirement of simulation application in military net-centric warfare. NCS is a novel distributed simulation method, which relies on the information grid infrastructure and service oriented architecture (SOA), defining unified standards for simulation resource description, access and share, and implementing specified simulation application task by dynamically creating and running the community of simulation task (CoST). The target of NCS is to establish a compatible architecture for the simulation and operational systemsby incorporating the simulation and information grid technology. Then, the simulation and operational systems can integrate and interoperate seamlessly while M&S service can be considered as a part of the C^4I system capability package. Runtime Support Platform (RSP) is the core technology of NCS in support of the dynamical CoST construction. CoST is a virtual organization that is motivated by a specified simulation task and is combined by relevant simulation resources and facilities distributed in WAN.

System model validation in software design phase is important to realize its smooth transition from analysis and design to software implementation. We should reasonably and correctly organize system as possible as we can, in order to enhance the reliability and manufacture efficiency of the software. It is an effective method to combine engineering description method (graphics modeling) with formal method (strict semantic and validation) to solve problems during the software architecture design process. Architecture description language (ADL) is the basics of software architecture development, which has well-knit math base and provides a precise definition of coherence, maturity, stipulation, correctness, etc. ADLs are possible method for us to design, develop and validate software in a systemic way.

There is a lot of benefits by using ADLs to design software architecture, as is shown above. This paper firstly introduces variable ADLs and analyzes their characteristics. Then XYZ/ADL language is chosen to describe Net-Centric Simulation Runtime Support Platform (NCS-RSP). Not only does the description by XYZ/ADL provide the intuitionistic graphics description and behavioral abstraction, but also it offers the dynamic semantic analysis and validation of CoST construction. Our research is the development and extendibility of both services composition technology and formal languages in military simulation domain.

doi:10.4304/jsw.7.9.2026-2032

II. Related Work

A. Formal Language

How to describe the architecture is a primary problem to study software architecture and there are variable exhibition forms and methods. For instance, diagram method, module connection language, soft component description and ADL, etc. So far, none of the description of architecture is formal and most of them rely on the experiences and the skills of designers. In traditional software development means, informal diagrams and texts are used to describe software architecture. They are neither able to characterize the ports we expected between components nor describe meaning of composition relation from different systems. In addition, This description method can neither be understood easily by developers nor adapted to formal analysis and simulation. Moreover, the method lacks corresponding support tools which can help accomplish the design work or be used to analyze the consistency or completeness. In contrast, the formal description method is much more precise, integrated and non-different meaning. So it is important to describe architecture in a formal and standard way. But before the formal description, the informal process is inevitable. We abstract some formal tags and symbols from informal courses and then standardize to achieve the formal architecture design and description. ADLs preserve the characteristics of traditional program and define the abstract elements that are suitable for software architecture expression and description for the integrity and abstract of software architecture. So they could describe software architecture precisely and strictly as well as to support the refinement, validation, evolution and analysis of software architecture.

There are variable ADLs which are comportable for certain domains, typically, C2 [3] is a kind of ADL based on component and message which applies to the software architecture of large-scaled frequent interoperation hierarchy GUI. Rapide [4] is a visual ADL which is based on events. It is applicable to the simulation of distributed system architecture. Darwing[5] and Wright[6] respectively adopt π evolution and CSP as their math basic and both languages are usually adopted by the description of distributed and parallel system architecture. There are some other famous ADLs such as Aesop, ArTek, SADL, UniCon, Weaves, etc. Many Chinese scholars propose several novel ADLs, like A-ADL [7], etc. N. Medvidovic and R.N.Taylor [8] introduces an ADL classification framework and analyze advantage and disadvantage of several typical ADLs in detail that indicates the importance of multi-layer, multi-view of architecture description and code creation.

B. XYZ/ADL

XYZ/E [9] is proposed by academician of Chinese academy of sciences TANG ZhiSong. XYZ/E is the first executable temporal logic language in the world which is based on Manna-Pnueli linear temporal logic system and combined with temporal logic operators. XYZ/E is both a logic system and a program design language, and supports real-time, blending, corresponding and visual program design methods. XYZ/E's expression mode is similar to normal advanced language so it is easily accepted by engineers and programmers. The characteristic of XYZ/E is that it can describe different abstract layers from formal specification to executable program within uniform semantic framework. XYZ/E can also express the dynamical semantic of program as well as static semantic of specification. This characteristic is comportable for describing software architecture.

XYZ/ADL [10] is an architecture description language based on XYZ/E. This language can express both the static and dynamical semantic of the software architecture. The characteristic of XYZ/ADL can remedy the shortcomings of other ADLs that cannot describe and analysis two above semantics. Such as the CSP is more suitable for describing dynamical behaviors and the Z is suitable for static properties. They are not executable and usually separate architecture specification description from structure realization. Refinement of specification is hard and is not able to support the whole process of software architecture. XYZ/ADL could not only formally describe software architecture, but also refine different abstract layers of architecture. In addition, we can use XYZ/ADL to validate the consistency of semantic in refinement process. XYZ/ADL is now widely used and many experts bring it to real projects, which push XYZ/ADL's further development. [11-15] list some research achievements and applications of XYZ/ADL.

Zhu Xueyang [16] proposes a dual software architecture description framework. The framework supports the basic concepts of software architecture which is used widely in software engineering. The front end of XYZ/ADL is a collection of graphical languages, including the usual 'box-and-line' diagrams as structure expression, the UML activity diagrams and state charts as behavioral expression. The back end of XYZ/ADL is the linear temporal logic language XYZ/E, which can represent both dynamic and static semantics of systems as its unified formal semantic backbone. The graphical languages at the front end can facilitate the communication among software engineers and their use of this framework. The formal language at the back end is the basis of formal analysis and verification.

XYZ/ADL is adopted as a formal language to research architecture of NCS-RSP. The concept of dual software architecture description framework is used here. Firstly figure the architecture graphics in the front end to describe the NCS-RSP architecture. Secondly we use UML sequence diagram to present the abstract behavior of the architecture. We introduce the reason why we use UML sequence diagram and then explain the abstract behavior of core service layer literally. Finally we make use of XYZ/ADL to describe the interoperation of core service layer formally, refine the services' interoperation and analyze in semantic view. Our work shows that XYZ/ADL could not only avoid the redundancy and imprecision of text, but also describe the architecture precisely, intactly and without misunderstanding. Future more reserach could analyze and validate the architecture

design completeness and correctness from visual viewpoint.

III. NCS-RSP ARCHITECTURE DIAGRAM DESCRIPTION

A. Graphics Description

To meet the requirements of NCS and provide various simulation services, we propose a multi-layer service-oriented NCS-RSP architecture, which will support the dynamical construction of CoST. Fig.1 shows the NCS-RSP architecture and in[1-2] we explain the architecture layer by layer and presents the support relationship between the layers.

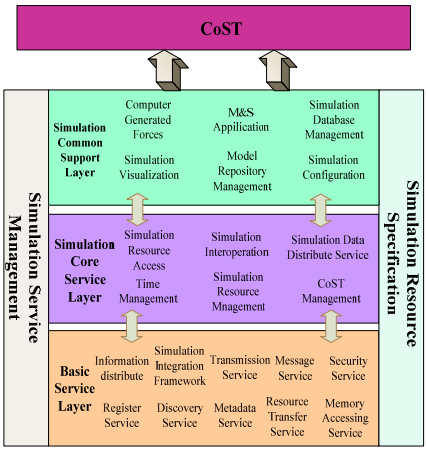

Figure 1. NCS-RSP architecture

B. UML Sequence Diagrams

The figure above displays the architecture from visual method which shows how the NCS-RSP supports CoST's construction and operation. After completing the CoST construction, the CoST runs in a Net-Centric environment. The CoST management service informs the members that the simulation starts by sending them a message. After receiving the message, the CoST members would operate according to the simulation task, while the simulation data distributed service guides the message interaction within the members in the CoST. When the simulation task completed, the CoST management service will inform all the members to end the simulation task and the CoST stops running. During the simulation runtime, CoST is supported by CoST management and simulation resources management service. Other members can apply joining and resigning the CoST at anytime during its runtime. Multi and different CoSTs can be run in NCS environment with various time management strategy, simulation steps, etc. They are supported by time management and simulation interoperability services. Besides the graphics description, we will also use UML sequence diagrams in front end analysis to describe NCS-RSP behavioral expression.

To describe the NCS-RSP which is composed by variable services, we consider the UML state chart diagram or activity diagram to reveal the services' interoperation. State chart diagram is able to show the action of an object in its lifecycle while activity diagram aims at the examples analysis which can help to understand examples' work flow or deal with multithreading applications. But if we want to show the interoperation among variable objects, state chart diagram or activity diagram is not comportable. So in stead we use UML sequence diagrams. In this part, we only present the UML sequence diagrams of core service layer in NCS-RSP to illustrate services interoperation relationship in core service layer, as shown in Fig.2. The other layers in UML charts are similar as the core service layer.

IV. NCS-RSP ARCHITECTURE FORMAL DESCRIPTION

A. Simulation Core Service Layer Formal Description

In the last section we use UML sequence diagrams to describe the behavioral expression of NCS-RSP core service layer; and we literal present the service interoperation. This description method is complex and redundant but is not easily understood or accepted by users. So we adopt XYZ/ADL language to describe the UML sequence diagram. After the formal description, we will refine the interoperation process of the time management and CoST management. By the introduction, Not only could we exactly know the relationship in the core service layer by formal way, but also we validate the correctness and completeness of the software architecture design.

There are variable services in the core service layer and each service exists as service components, which cause the services interoperation among the layer complicated. In NCS-RSP, the core service layer is only a part of it, so this layer also works with simulation basic service layer and common support layer as component. In This section we take core service layer as composite component which is connected according to certain requirement and then we describe its structural models. This is the first step to refine;and reflect the messages transmission relation withinmain components.

Composite component incarnate the configuration and hierarchy relation both in landscape and portrait viewpoint. Composite component ports description is similar to simple ports, but its behavior is denoted by connection of several sub-components. So our description would illustrate which component and connector instances are contained in a composite component and how they get inter-connected with each other. We define these sub-components based on the service contents of core service layer as follows in Fig.3: SimResMan, SimAccess, CoSTMan, SimInter, TimeMan, DataDistri. Fig.3 shows us the description of core service layer.

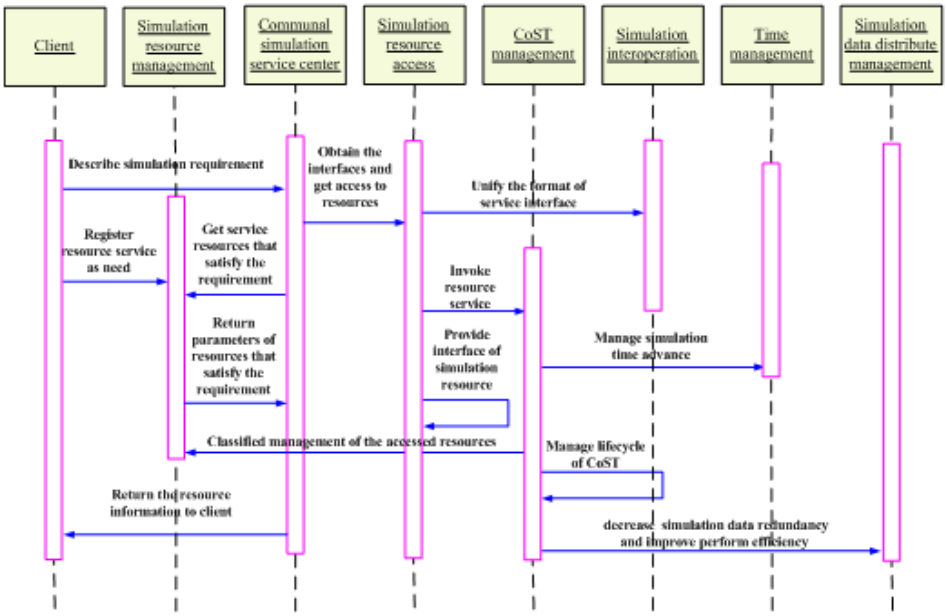

Figure 2. NCS-RSP core service layer UML sequence chart

The simulation core service layer composite component combination announcement indicates which component and connector instances are contained in it and grammar is as follows.

Fig. 3 is the first step refinement process of core service layer composite component and is also the initial formal verification process. We define the composite component input and output port. The first step of the simulation core service layer is given and is also the elementary process of formal validation. Firstly we define the input and output ports of composite component. Secondly we attach and bind the sub-component ports to connector role. The '#' defines attachment operation that component ports in left side are attached to connector roles in right side. Attachment shows the relation of components and interoperation in which the components participate. Some sub-component ports are not attached to any roles; instead they are specified as ports of composite component. We name it binding operation and define as '##'. Right formula is ports of composite component so we omit the component name. Finally, in '%COMPUTATION' part we explain the semantic of composite component, and regard the '%COMPUTATION' as elementary refinement.

B. Refinement

RONG M[17] introduced three methods of architecture refinement: basing on behavior substitute, style and component. The method based on component can be evolved into two processes: refinement from coarse grained to fine grained, and refinement from informal to formal. This method separates the architectural level coarse grained components into components in terms of specifications. Components also can determine their sub styles according to requirement and form fine grained components step by step.

In the light of refinement above, we are aware of the interoperation relationship within services in core service layer. However it is only the basic description of the abstract software architecture. In this section we will adopt refinement method based on component to construct model of core service layer architecture. This method will be more exact and complicated in realization details than previous refinement. Refine to this degree, we could not only achieve the executable program level, but also validate the correctness and completeness of architecture design roundly. We take time management and CoST management as examples to illustrate the refinement process.

a) Service component description

For instance, we design the time management component as Time_Management in Fig.4 which owns a port TimeServiceport. Similarly we can also define CoST management component CoST_Management and its port CoSTServiceport, etc.

Other services could also be defined by the component definition method but they are not listed here.

b) Service connector description

This part we shows the relation between CoST Management and Time Management. They are connected through the connector defined by XYZ/ADL. Fig.5 illustrates the definition of time management connector Time_ManagementConn.

c) Refinement

It is shown above that text description can not depict problems clearly. But graphics description is more intuitional and understandable than text. I.e. Integral

Description of Core Service Layer

```
%COMPONENT CoreService = = [
  %PORT   Recive1 = = MESSAGE;
          □ [LB = START_receive=>Receive1 ? ComService ∧ $O LB = e1]
          //receive messages from common support layer
  %PORT   Recive2 = = MESSAGE;
          □ [LB = START_receive=>Receive2 ? BasicService ∧ $O LB = e1]
          // receive messages from basic service layer
  %PORT   Send1 = = MESSAGE;
          □ [LB = e0=> Send1! state1 ∧ $O LB = e1]
          // send messages to common support layer
  %PORT   Send2 = = MESSAGE;
          □ [LB = e0=> Send2! State2 ∧ $O LB = e12]
          //end messages to basic service layer
%COMPOSITION == [SimResMan; SimAccess; CoSTMan; SimInter; TimeMan; DataDistri;
                 c1:Conne1; c2:Conne2; c3:Conne3; c4:Conne4; c5:Conne5]
%ATTACHMENT = = [ CoSTMan.Send1 # c1.Source; SimResMan.Receive1 # c1.Sink;
SimAccess.Send1 # c2.Source; SimInter.Receive1 # c2.Sink;
SimAccess.Send2 # c3.Source; CoSTMan.Receive1 # c3.Sink;
                 CoSTMan.Send2 # c4.Source; TimeMan.Receive1 # c4.Sink;
                 CoSTMan.Send3 # c5.Source; DataDistri.Receive1 # c5.Sink;
                 CoSTMan.Send1##Send1; SimResMan.Send2##Send2;
                 CoSTMan.Receive1##Receive1; SimResMan.Receive2##Receive2 ]
%COMPUTATION = = [
      LB = START_receive1=> $O LB = e1;
LB = e1=> $O Receive1 = CoSTMan.Receive1 ∧$O Receive2 = SimResMan.Receive2 ∧$O Send1 =
                 CoSTMan.Send1∧$O Send2 = SimResMan.Send2 ∧$OLB = e2;
      LB = e2=> || [SimResMan.COMPUTATION; c1.GLUE; SimAccess.COMPUTATION; c2.GLUE;
            CoSTMan.COMPUTATION; c3.GLUE; SimInter.COMPUTATION; c4.GLUE;
            TimeMan.COMPUTATION; c5.GLUE; DataDistri.COMPUTATION ]]
```

Figure 3. NCS-RSP core service layer description

Description of Time Management Component

```
%COMPONENT Time_Management = = [
%PORT TimeServiceport = timeserviceprovide (% Recive1 int.identifyID;
                                             % Recive2 DWORD timestep;
                                             % Send1 BOOL block;
                                             % Send2 BOOL AdvState)
        //define messages in TimeServiceport port
        //define advance node ID、time step、deadlock identifier、time advance state
        //similarly define messages timeservicereceive
];
        //define operations in TimeServiceport port
[timedatareceive [Required; Type: Input; Message: timeservicereceive];
timedataprovide [Provided; Type: Output; Message: timeserviceprovide]];
        //define behavior in TimeServiceport port
[//provide messages to CoST management and receive call back information]
%SERVICEBEHAVIOR = = [//component behavior is the same as port behavior]
%PROPERTY = = Resource Specification
```

Figure 4. Service Component description

NCS-RSP core service layer architecture in Fig.1 and Fig.2 is showed legibly and pellucidly. But graphic description only can show service contents. It is hard to analyze the connection relation among services or validate the correctness of design. So we use formal language to describe the architecture and also we refine the architecture from high abstract level to low concrete level. This description and refinement forms an architectural layer that every refinement step is corresponded to a refinement mode,,which guarantees the correctness of refinement and also validates the completeness of design. The formal refinement of time management service is shown in Fig.6

Description of Time Management Connector
%CONNECTOR Time_ManagementConn = = [
//define CoSTManagementRole messages: costmanagementresponse, costmanagementrequest;
TimeServiceRole messages: timeserviceprovide, timeservicereceive
//define operations of CoSTManagementRole: Serviceinvoke, ReturnServiceinvokeMessage;
Operations of TimeServiceRole: timedatareceive, timedataprovide
%ROLE CoSTManagementRole = = [costmanagementresponse; costmanagementrequest];
[Provided_Operation; Serviceinvoke;
Required_Operation; ReturnServiceinvokeMessage];
[//receive service calls information and return results]
//define behavior of connector role
//similarly define role CoSTManagementRole
%GLUE = = [//describe interoperation of two roles: provide time management information by timedataprovide and execute time management service]]

Figure 5. Time Management Connector Description

Description of Time Management Component
%COMPONENT Time_Management = = [
%PORT TimeServiceport = timeserviceprovide (% Recive1 int.identifyID;
% Recive2 DWORD timestep;
% Send1 BOOL block;
% Send2 BOOL AdvState)
//define messages in TimeServiceport port
//define advance node ID、time step、deadlock identifier、time advance state
//similarly define messages timeservicereceive
];
//define operations in TimeServiceport port
[timedatareceive [Required; Type: Input; Message: timeservicereceive];
timedataprovide [Provided; Type: Output; Message: timeserviceprovide]];
//define behavior in TimeServiceport port
[//provide messages to CoST management and receive call back information]
%SERVICEBEHAVIOR = = [//component behavior is the same as port behavior]
%PROPERTY = = Resource Specification
//service components should meet the specification requirement

Figure.6 Time Management Service Refinement

V. DISCUSSION

This paper introduces the NCS-RSP architecture, and adopts a XYZ/ADL based architecture description framework. Firstly we introduce architecture graphics and UML sequence diagram while they describe the architecture in visualization method. Then we refine the services in architecture from formal semantic point of view. The research above provides a formal theoretical guidance of services reuse and composite in NCS-RSP. It is an innovative attempt of service composite formal description in military simulation domain.

An ADL should not only has strict formal semantic, but also consider engineering usage in practice. Consequently ADLs can really perform a function in software development process and denote different abstract layers conveniently. We use XYZ/ADL language to refine the CoST system component and realize smooth link from visual graphics to formal method. The interoperation of connectors' roles, components' ports and ports binding becomes an ordinal combined model of the system. The formal description of NCS-RSP supports

services composition semantic analysis and is an elementary attempt in military simulation field.

UML and its extended mechanism are widely used in practice development as software architecture description tool. We combine the software architecture with UML and adopt UML sequence diagram as architecture behavioral description.

In the formal description of NCS-RSP architecture, we only introduce several services but the whole platform architecture description is similar. Time management is one of the key technologies of NCS-RSP and our refinement only presents a primary time management algorithm. The detailed algorithm will be given in future research

REFERENCES

[1] Mao S J, Li Y P, Lin J N, Deng K B, Sun L Y. Research on Concepts and Technique of Network-Centric Simulation [J].Journal of System Simulation, Vol.22, No.7, Jul, 2010.

[2] SUN L Y, LIU Z, MAO S J, DENG K B. Research on the Runtime Support Platform for the Net-Centric Simulation [J]. ICACTE, 2010.

[3] R.N.Taylor, N, Medvidovic, et al. A component-and-message-based architecture style for GUI software [J]. IEEE Trans. Software Engineering, 1996, 22(6):390-406

[4] D.C.Luckham.et a1. The Rapide language. Stanford University, http://pavg.stanford.edu/ rapide/language.html, l997-07-29

[5] J. Magee. J.Kramer. Dynamic structure in software architectures. In: Proc. the 4th ACM SIGSOFT Symposium on Foundations of Software Eng. New York: ACM Press. 1996. 3-l4

[6] R. Allen, D.Garlan. A formal basis for architectural connection. ACM Trans. Software Eng, and Methodology, 1997, 6(3):213-249

[7] MA J T, FU S Y, LIU J R. A-Adl: A Multi-agent System Architecture Description Language. Journal of Software, 2000, 11(10): 1382-1389

[8] N. Medvidovic, R.N.Taylor. A classification and comparison framework for software architecture description languages. IEEE Trans. Software Engineering, 2000, 26(1): 70-93

[9] TANG Z S *et al*. Temporal Logic Design and Software Engineering. Beijing, Science Press,2002

[10] ZHU X Y. Research on Software Architecture Formal Description [D].Chinese Academy of Science, 2004.

[11] CHEN L L, ZHANG G Q. A Matching Method from XYZ/ADL to UML [J]. Journal of Suzhou University (Nature Science), Vol.22,No.1 Jan, 2006

[12] LIU Y L. XYZ/ADL-Based Electives Management System Architecture Description [J]. Journal of Yunnan University, Vol.26

[13] RAO Y, LI Z C. XYZ/ADL-Based Web Services Architecture Description [J].System Engineering Theory & Practice, 2006

[14] YANG J Z, RONG M, ZHANG G Q. Aspect-oriented Software Architecture Description Language AO-ADL [J]. Computer Engineering, Vol.34, No.10, 5, 2008.

[15] RONG M, ZHANG G Q, Software Architecture Description Approach Integrating Formal Methods and Visual Methods [J]. Computer Science, Vol. 32, No.4, 2005.

[16] ZHU X Y. The Dual Software Architecture Description Framework XYZ/ADL [J]. Journal of Computer Research and Development, 2007, 44(9): 1485-1494.

[17] RONG M, ZHANG G Q. Research on Software Architecture Refinement Methods [J]. Computer Science, 2003, 30(4):108-110.

Sun Li-yang was born in Nanjing, China, 1985. He received his B.S. degree in Jiangnan University in 2008. Now he is studying his postgraduate and doctoral programs in Nanjing University of Science and Technology, Jiangsu Province, China. His research interest is network information system modeling and simulation,

Dr. Sun is membership of Chinese System Simulation Association and Japanese institute of Electronics, Information and Communication Engineers.

Mao Shao-jie was born in Jiangyin, China, 1963. He received his B.S. degree in Nanjing University, China. Now he is Research Fellow senior engineer in CETC28th. His research interests include modeling and simulation of military command and control system evaluation.

He is deputy Director of C4ISR National Defence Science & Technology Key Lab, Since 2000, he has chaired more than 10 research projects under the National Basic Research Program of China (973 Program) and the twelfth five-year plan, etc. He was awarded one 2nd class National Science and Technology Advance Award. Up to now, he has published more than 30 papers and published10 monographs.

Prof. Mao is Executive director of Chinese System Simulation Association member of Chinese Modeling and Simulation Standardization Committee.

Liu Zhong was born in Anhui, China, 1963. In 1988 he received PhD degree in the University of Electronic Science and Technology, China. In 1991 to 1993, he was funded by Japanese Ministry of Education Foundation to do postdoctoral research at the University of Kyoto, Japan. 1997-1998 he was Visiting Research Scholar to Hong Kong Chinese University. Up to now, he is professor of the Department of Electronic Engineering and the chief of human source department in Nanjing University of Science and Technology. He is mainly engaged in non-linear chaos dynamics; signal processing; radar and communications.

Prof.Liu has undertaken a national outstanding Young Teacher Foundation,Fok Ying Tung education Fund, the former Department of Electrical and Ordnance Industry Corporation and other research funds research. Prof.Liu has published over 80 articles in international academic journals and conferences. Prof.Liu was awarded two ministerial-level National Science and Technology Advance Award and was honored as Fok Ying Tung young Teacher Award (research) and Jiangsu excellent young teachers and so on.