# AUTOMATIC MODEL CONSTRUCTION AND EXTENSION FOR DISTRIBUTED SIMULATION

**HU YU**

*(Bachelor of Engineering, Tsinghua University, China)*

**A THESIS SUBMITTED**

**FOR THE DEGREE OF MASTER OF SCIENCE**

**DEPARTMENT OF COMPUTER SCIENCE**

**NATIONAL UNIVERSITY OF SINGAPORE**

**2003**

# Acknowledgement

I would like to thank my supervisor, Dr Gary Tan Soon Huat, for his direct guidance and help throughout my study and research period. He organized many seminars and guest talks, from which I learned a lot of background knowledge in the Distributed Simulation and the HLA OMT. Also he gave me advice and encouragement during my work and study. Without his generous support and suggestions, my thesis work cannot be finished this smoothly.

I should also thank Professor Rassul Ayani in Royal Institute of Technology (KTH), Sweden and Mr. Farshad Moradi in Swedish Defense Research Agency (FOI). They invited me to join this research project and supported my research trip to Sweden. There I integrated the collaboration work with colleagues in this project and exchanged our ideas. I really appreciate their good suggestions to my study and their efforts to help me solve the problems I met.

At the same time, I am grateful to all the staff members of the Modeling and Simulation Group for their help in my research area. I also acknowledge my lab fellows, Yu Jun, Zhao Na, Ng Wee Ngee and Hu Yanjun for their help both in my research and in my life, and for the pleasant and friendly environment of the computer system lab.

Finally, a debt of thanks is owed to my family and best friends, who provide me with endless help and support in my life.

# Table of Contents

# List of Figures

# List of Tables

# Summary

The High Level Architecture (HLA) is widely used in Modeling & Simulation as a common structure which aims to facilitate the reusability of simulation models and the interoperability among the models. In order to support these goals, the HLA Federation Development and Execution Process (FEDEP) [1] was defined to provide a high-level framework to build a set of multiple interacting simulation models, the HLA federation. It is a generalized process for building HLA federations from scratch. However, simulation model development, implementation, testing, and execution are time consuming and expensive processes. So, is there a way to reuse the existing simulation models and develop new models more efficiently?

In the year 2001, a project “A net based modeling and simulation platform (NetMas)” [2] was initiated aiming at developing a platform to utilize the simulation models/codes and computing resources more efficiently. One of the main parts of the NetMas platform is the Model Construction (Design) Environment (MCE), in which a model builder may use the existing sub-models to compose new models. The models used here are Federation Object Model (FOM) and Simulation Object Models (SOM) based on the HLA OMT [3]. The key aspect of the MCE is to ensure that the models are compatible and interoperable, which is named the “Matching Algorithm” here. Also the comparison process has to be done automatically.

This thesis describes the design and development of the current MCE, in which the composing FOM/SOMs are checked by the Matching Algorithm and the new FOM

is developed. In addition, it examines whether the HLA OMT is sufficient for ensuring the compatibility and interoperability among federates in the federation. After performing two case studies and theoretical research, it is found that the HLA OMT may bring some substantive interoperability problems and non-exchange data problems. An Extensible Element scheme is introduced in this thesis later to improve the semantics of the federate/federations. The scheme is helpful to strengthen the identity and accuracy of the object models.

This work attempts to investigate the possibility to build HLA federations automatically. Meanwhile, it argues for an alternative method for federation development. It can be useful to further facilitate the reusability and interoperability of the HLA federate/federations.

# Chapter 1 Introduction

## 1.1. Overview

Simulation is useful in such situations where it is difficult, expensive or impossible to do experiments with the real system for some special reasons, or, in some cases, the experiment has to be done for several times but the resources are unable to be recovered after each time. Simulation enables people to represent some behavior of the real systems, to simulate it and to get the result before they actually build the system. By getting the results in advance, simulation can avoid wasting time and money on those low cost-efficiency projects/plans.

### 1.1.1. Computer Simulation

Computer simulation is the process of designing and executing a model on a digital computer [4]. It is one of the most important applications in the computer science area. With the emergence of low-cost and high-power computers, computer simulation is making great strides in recent years.

There are several areas that computer simulation applications can make contributions to. With the strong computational power of the current computer systems, computer simulation can help the wide-spread use of computational intelligence. Also, computer simulation facilitates the development of multimedia user interface and virtual reality, which further promotes the popularity of computer-based interactive games. As Object-Oriented Analysis and Design is getting more mature, computer

simulation can play a more and more important role as a useful computer application area.

### 1.1.2. Parallel and Distributed Simulation

Sometimes the simulation would be involved with heavy computation or would vary in problem size. Thus, increasing the numbers of processors and memories to parallelize the computation would help in this situation. There are two types of concurrent computation, one is parallel simulation, and the other is distributed simulation.

Parallel simulation is often used to improve the performance of the system. Usually parallel simulation uses multiple CPUs and data stores. By decomposing the simulation into different parts and increasing the computation power, parallel simulation can reduce the execution time and increase the problem size.

Distributed simulation is used on both Local Area Network (LAN) and Wide Area Network (WAN). The purpose of it is to strengthen the collaboration among computer systems in different locations. Distributed simulation can potentially increase the scalability of the system. Also, it would increase the fault tolerance of the simulation system [5]. With the existence of reliable and broadband data communication technologies, distributed simulation has been applied in the areas below:

- ***Military Applications,*** including War Gaming Simulations, Training Environments and Test & Evaluation;

- ***Education and Training;***
- ***Entertainment and Gaming;***
- ***Telecommunication Networks;***
- ***Transportation.***

### 1.1.3. High Level Architecture (HLA)

One of the first concerns of distributed simulation is how to shape and organize the simulation models in a unified format. The High Level Architecture (HLA) has been developed under the leadership of the Defense Modeling and Simulation Office (DMSO) to provide a common architecture for distributed modeling and simulation (M&S). The HLA was widely used in M&S for it facilitated the reusability of the simulations and the interoperability among them, and it was approved as an open standard through the Institute of Electrical and Electronic Engineers (IEEE) in September 2000.

To support the general goals of the HLA, the HLA Object Model Template (OMT) [3] was introduced to prescribe the format and syntax for recording the information in HLA object models. It not only provides a template for documenting the HLA-relevant information as individual models (federates), but also facilitates understanding and comparisons of different simulation models in a unified simulation environment (federation).

There are many ways to construct an HLA federation. Among them, the HLA

Federation Development and Execution Process (FEDEP) [1] provides a high-level framework and a common sense system engineering methodology for HLA federation. It divides the federation construction into six basic steps in sequence, from federation objective definition, model development to federation execution and results collection. It can be used to meet all kinds of individual application requirements. Thus, it is deemed as a generalized process for building HLA federations from scratch.

However, simulation model development, implementation, testing, and execution are time consuming and expensive processes. With the wide spread use of the Internet, more and more resources can be shared online, which facilitates the possibility to reuse these resources. So, is there a way to reuse the existing simulation models and develop new models more efficiently? The question will be explored in this thesis.

## 1.2. Related Work

Many researchers have tried to establish the modeling and simulation environment for construction and maintenance of simulation models. The HOMME (Heterogeneous Object-oriented Multipurpose Modeling Environment) [6] was developed to construct heterogeneous models with different techniques, such as the differential algebraic equations (DAEs) and neural networks. The user can use an editor in the HOMME system to generate intermediate representation of meta-classes or meta-objects. Besides, the user can store the models in an object-oriented database. However, a match for the modeling process still needs to

be developed in the HOMME.

The NanoComp Project [7] was started in 1998 and aimed at investigating the feasibility of future electronics based on quantum devices. A web-based collaborative environment supporting reuse and interoperability of M&S components was developed in the project [8]. To select from the heterogeneous models and tools, the environment relies on the Dublin Core Metadata Initiative (DCMI) [9], which is an open forum engaged in the development of interoperable online metadata standards that support a broad range of purposes and business models. The environment supports on-line research and development by sharing and reuse of various models in different formats. But the compatibility among the different formats of the models is doubted.

A novel design environment for developing multi-agent systems (MASs) for applications in mobile robotics was introduced in [10]. This automated design environment used generic algorithm to select the best candidate designs, then created and managed the new models. The environment was based on the HLA.

## 1.3. Background

In year 2001, a project “A net based modeling and simulation platform (NetMas)” [2] was initiated aiming at developing a net based platform to utilize the simulation models/codes and computing resources more efficiently. It is a cooperative project between NUS and two major Swedish sponsors, the Royal Institute of Technology (KTH) and the Swedish Defense Research Agency (FOI). The main purpose of this

project is to investigate how the advances in networking technologies, such as the Internet, can contribute to improve interoperability, portability and reusability of simulation models and codes.

One of the first concerns is collaboration. The Internet enables people from different locations to work together. By sharing all the outcomes as a big virtual library to the authorized members of a group, each member gains a lot more than what he/she can generate. Thus, those who can only build small specialized sub-models can also contribute to the society in building large stand-alone monolithic models. As reward, they gain access to the entire range of models in the library. Figure 1-1 shows an illustration of the collaboration among different organizations, e.g. NUS, FOI and KTH.

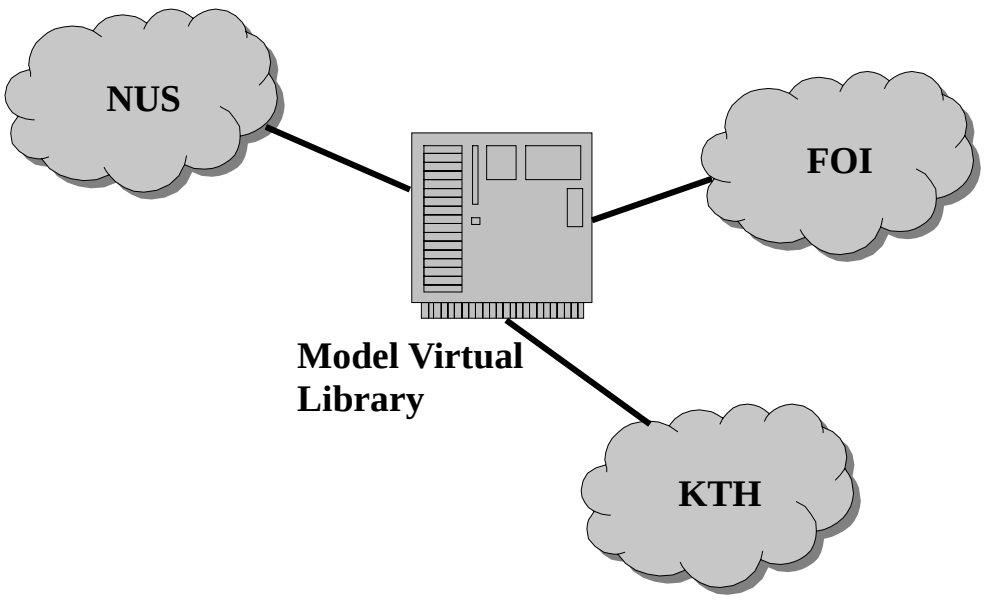


**Figure 1-1 The Collaboration Among Different Organizations**

Also, a modeling environment should provide the development tools and storage and

management of the models. The model development tools (textual or graphical) should be available for the user to build the models. After creating the models, the user should be able to store it in a permanent structure that could be understood by other users.

## 1.4. Objective

This thesis work focuses on one of the main parts of the NetMas platform, the Model Construction Environment (MCE). In the MCE, a model builder may use the existing sub-models to compose new models. The key aspect of the MCE is to ensure that the models are compatible, interoperable and can communicate with each other. This is achieved through an algorithm, called the “Matching Algorithm”, which automatically checks compatibility among the sub-models.

All these require the models to be built in a well structured template or format. The HLA OMT is chosen as our model construction standard for its great support for model reusability and interoperability. The sub-models that are used in this platform are Simulation Object Models (SOM), the description of a federate according to the OMT. Besides, the corresponding Federation Object Model (FOM) is created under this scheme and saved for future reuse. The FOM/SOM will be discussed in detail in Section 2.1.3.

Another interesting question is whether the HLA OMT can ensure the compatibility among the sub-models or not. Although HLA OMT has defined the syntax and structure of the HLA federate/federation, it does not guarantee the fidelity levels

(how well the simulation makes its representation) among the simulation models are the same. Thus, we propose an Extensible Element with priority level scheme to help in this study.

## 1.5. Contribution

This work attempts for an alternative method to automatically build HLA FOMs with existing HLA SOMs, which further facilitates the reusability of HLA federates and the interoperability among them. It is a first attempt to investigate the possibility to build HLA federation automatically. To check the compatibility among the simulation models, a Matching Algorithm is set up and tested with real examples.

Furthermore, this work examines whether the HLA OMT is sufficient for ensuring the compatibility, interoperability and communicability among federates in the federation. Some practical suggestions are made towards the current HLA OMT specification, which show a promising and potentially effective way to describe and shape federates through the HLA.

This work can contribute to more cost-efficient methodologies for development and execution of simulation models and codes. Meanwhile, it brings forward some helpful suggestions and early stage implementations to the augmentation of the current HLA OMT standard.

## 1.6. Organization of the thesis

The rest of this thesis is organized as follow:

Chapter 2 describes the background of the distributed simulation field, including the HLA OMT, the FEDEP model and the interoperability and fidelity challenges of the model construction.

Chapter 3 introduces the MCE environment, which is made up of a client interface and a file server, and how they work together. The infrastructure part introduces the XML based DIF as the format for models as well as a DIF-XML converter. The framework part introduces the client interface and the file server and how they communicate.

Chapter 4 first explains the Matching Algorithm used in the MCE, which is made up of four consecutive processes: the DataType Check, the Routing Space Check, the Object Match and the Interaction Match. The Matching Algorithm is then enhanced with some concepts based on the FOM Agility. Finally, this chapter explains how to build a FOM using the data which are parsed in the Matching Algorithm.

Chapter 5 investigates on two case studies in building FOMs with the MCE. The feasibility of the MCE is discussed and some deficiencies of the scheme are given as well as the reasons behind them. The limitations of the current HLA OMT are also discussed in this chapter.

Chapter 6 introduces the traditional schemes that are used to strengthen the semantics of the simulation model. Besides, an embedded Extensible Element scheme for the DIF is introduced in this work and is suggested as an augmentation of the current HLA OMT.

Chapter 7 concludes the thesis and suggests some aspects for future work.

# Chapter 2 Basic Concepts in Distributed Simulation & HLA

This chapter provides the reader with some necessary theoretical background in the distributed simulation field. The HLA OMT is a standard format and syntax for recording the information in HLA object models. The FEDEP model is a generalized process for building HLA federations. And we still need to think about the fidelity and interoperability challenges of the models to ensure that what we have built represents exactly what we want. These are described in the following three subsections respectively.

## 2.1. HLA OMT

### 2.1.1. HLA Overview

Different simulations are developed by different organizations. This raises the problem of interoperability among the different simulations when people want to utilize others' simulation. At the same time, for the reason of the high cost in constructing a new model every time, there exists the big desire to reuse the existing simulations. Setting up flexible and abstract standards for simulation is under demand.

The High Level Architecture (HLA) provides a common architecture for modeling and simulation and is widely used across large amount of simulation application areas [11]. The HLA was developed under the Department of Defense (DoD)

Modeling and Simulation Master Plan [12] to facilitate interoperability among simulations and promote reuse of simulations and their components.

The HLA is made up of three main components:

- ***HLA Rules,*** which describes the general principles defining the HLA and delineates the set of rules that apply to HLA federations and federates;
- ***HLA Interface Specification,*** which provides a specification for the DoD HLA functional interfaces between federates and the runtime infrastructure (RTI);
- ***HLA Object Model Template (OMT),*** which prescribes the format and syntax for recording the information in HLA object models.

In the HLA, the simulation unit could be one or several federates or federations. A federation is defined as a set of simulations that are used to form a larger model or simulation. A federate is a member of a federation. Federate and federation are the basic entities of the HLA infrastructure.

The HLA is widely adopted in M&S for the following reasons: Firstly, it enables the simulation to be shaped as interacting components/models, which could be easily implemented by different parties. Secondly, it provides the simulation planning method, which is essential in the simulation preparation period. Thirdly, it is a good way to model the simulation. Lastly, it provides the users with means for model validation and verification.

### 2.1.2. Runtime Infrastructure (RTI)

A simulation is not yet accomplished until it is implemented with certain software. The Runtime Infrastructure (RTI) is defined to work with the HLA as a collection of software that provides common services required by multiple simulation systems. It is also an architectural foundation encouraging portability and interoperability. The RTI consists of six service groups:

- ***Federation Management:*** Create and delete federation executions join and resign federation executions control checkpoint, pause, resume, restart
- ***Declaration Management:*** Establish intent to publish and subscribe to object attributes and interactions
- ***Object Management:*** Create and delete object instances; Control attribute and interaction publication; Create and delete object reflections
- ***Ownership Management:*** Transfer ownership of object attributes
- ***Time Management:*** Coordinate the advancement of logical time and its relationship to real time
- ***Data Distribution Management:*** Supports efficient routing of data

Figure 2-1 illustrates how a simulation is built with HLA and how HLA and RTI play in the framework.

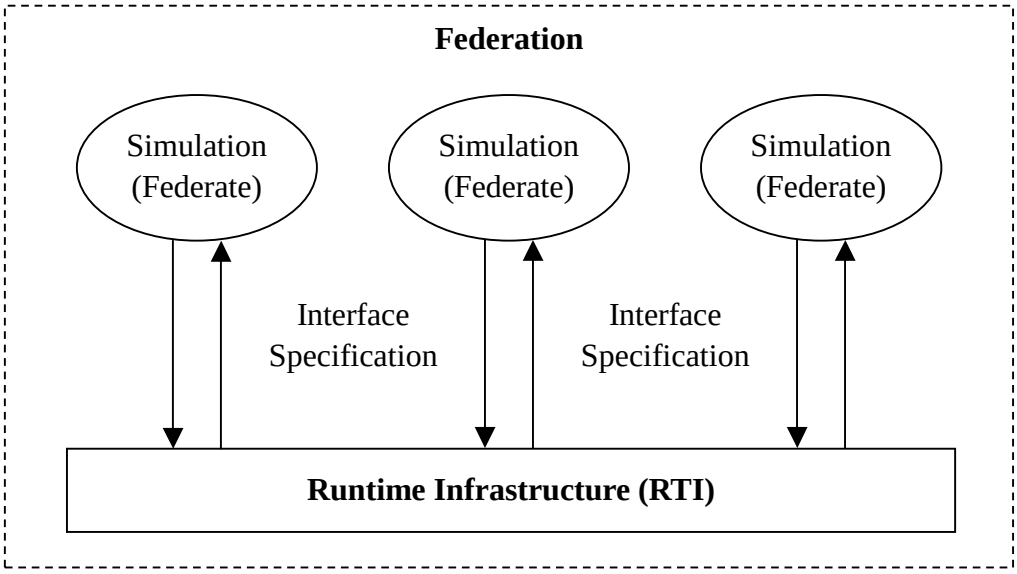


**Figure 2-1 An HLA Federation**

### 2.1.3. HLA OMT

The HLA OMT is a standardized structural framework for specifying relevant information of the simulation models. It is an essential component of the HLA as it provides a common mechanism for specifying the data exchange and federate coordination, and a foundation that design and application tool sets for HLA object model construction can base on.

In the OMT, a Simulation Object Model (SOM) is used to describe an individual federation member (federate), while a Federation Object Model (FOM) is used to describe a named set of multiple interacting federates (federation). In either case, the primary objective of the OMT is to facilitate interoperability among simulations and reusability of simulations or simulation components.

Functionally, the FOM should ensure the critical requirement of a federation, that is,

all federation members should achieve a common understanding among all the participating federates. At the same time, it provides a specification for data interchange among federates in a common and standardized format. Differing from the FOM, the SOM expresses the suitability of simulation systems for best meeting the overall objectives in a federation.

The OMT consists of the following components in the form of tables:

#### 2.1.3.1. Object Model Identification Table

The Object Model Identification Table is used to document important identifying information with the HLA object model. These information, such as the point-of-contact (POC), is necessary for other users who wish to reuse the model and want to know the details about how a federate/federation was constructed. Table 2-1 shows an example of the Object Model Identification Table:

| **Object Model Identification Table** | |
|---|---|
| **Category** | **Information** |
| Name | Mission Planning Simulator |
| Version | 00.6.1 |
| Date | 01/24/2000 |
| Purpose | Provide space-time location and orientation of simulated test objects and instruments to data collectors. |
| Application Domain | Test and Evaluation |
| Sponsor | Army Test and Evaluation Command |
| POC | Mr. Jeffrey Thomas |
| POC Organization | White Sands Missile Range |
| POC Telephone | (505) 678-4597 |
| POC Email | thomasj@wsmr.army.mil |

**Table 2-1 An Example of the Object Model Identification Table**

### 2.1.3.2. Object Class Structure Table

In the OMT, an HLA object class is a collection of objects with certain characteristics or attributes in common. It is essential for specifying characteristics (attributes) of simulation objects. Also, it provides the means for federation participants to subscribe to information about all individual instances of HLA objects with common attributes. An HLA class structure is defined in terms of hierarchical relationships among classes of objects.

The Object Class Structure Table is used to record the namespace of all simulation/federation object classes and to describe their class-subclass relationship.

### 2.1.3.3. Interaction Class Structure Table

An interaction is defined in the OMT as an explicit action taken by a simulated entity (or aggregation of entities) in a federate that may have some effect or impact on another federate. An interaction structure is composed of relations of generalization (or specialization) between different types of interactions.

The Interaction Class Structure Table is used to record the namespace of all simulation/federation interaction classes and to describe their class-subclass relationship.

### 2.1.3.4. Attribute Table

Attributes of HLA object classes are specified to support subscription to their values

by other interested members of a federation. During the federation execution, knowledge of object attributes is commonly required for effective communication between federates.

The Attribute Table is used to specify features of object attributes in a simulation/federation.

#### 2.1.3.5. Parameter Table

Interaction parameters are used to associate relevant and useful information with classes of interactions. These parameters are used to support calculation of new attribute values for objects affected by the interaction.

The Parameter Table is used to specify features of interaction parameters in a simulation/federation. For every interaction class identified in the interaction class structure table, the full set of parameters associated with that interaction shall be described in the parameter table.

#### 2.1.3.6. Routing Space Table

Routing spaces are the most fundamental Data Distribution Management (DDM) concept which is defined in the HLA RTI Programmer's Guide [13]. A routing space is a multidimensional coordinate system through which federates either express an interest in receiving data or declare their intention to send data.

The Routing Space Table is used to specify routing spaces for object attributes and interactions in a federation.

#### 2.1.3.7. FOM/SOM Lexicon

The FOM/SOM Lexicon provides a means for federations to document the definitions of all terms (such as the objects, the interactions) utilized during the construction of simulation models. It helps to achieve a common understanding of the semantics of the model. Federation/federate developers are provided maximum flexibility in this lexicon. With the lexicon, constructing libraries of reusable data views and making the libraries available for general use is possible in future application.

The FOM/SOM Lexicon Table is used to define all of the terms used in the tables.

### 2.1.4. Data Interchange Format (DIF)

The OMT Data Interchange Format (DIF) is a standard file exchange format used to store and transfer FOMs and SOMs between FOM/SOM builders. It is also specified in the OMT [3]. The DIF is formally defined in terms of extended Backus Naur Form (BNF), which is normally used to describe inductive specifications. As defined in [14], BNF has three main parts:

- *Terminals*: require no further definition
- *Non-terminals*: are defined in terms of other non-terminals and terminals
- *Productions*: state how the non-terminal is constructed for each of them

The DIF is built on a common meta-model that represents the information needed to represent and manage object models. It documents the major components in OMT,

for instance, the object model identification, the objects, the interactions, the attributes and parameters. For example, Figure 2-2 illustrates the HLA OMT DIF BNF definition for HLA Object:

```
<Class> ::= “(Class (ID ” <<CLS_ID>> “)”
   “(Name ” <<CLS_Name>> [<NoteRef>] “)”
      [“(MOMClass ” <<CLS_IsMOMClass>> “)”]
      “(PSCapabilities ” <<CLS_PSCapabilities>> “)”
      [“(Description ” <<CLS_Description>> “)”]
      {<ClassComponent>}* “)”;
<<CLS_ID>> ::= <Identifier>;
<<CLS_PSCapabilities>> ::= “P” | “S” | “PS” | “N”;
<<CLS_Name>> ::= “"” <TextString> “"”;
<<CLS_IsMOMClass>> ::= “True” | “False”;
<<CLS_Description>> ::= “"” <TextString> “"”;
<ClassComponent> ::= <Attribute> | <SuperClass>;
```

**Figure 2-2 An Example of HLA OMT DIF BNF Definition**

As one can see from Figure 2-2, an HLA object is defined with class name, class ID, PSCapabilities, Name, Description, class component, etc. Each component of the object class can be further defined using the similar BNF format as shown in this example.

The DIF is structured as a stream of object model meta-data, and it is always represented in a simple ASCII file. Furthermore, the DIF content consistency defines a set of rules to meet the requirements of the OMT. Thus, it could be regarded as a representation of the FOM/SOM.

## 2.2. Federation Development and Execution Process (FEDEP)

### 2.2.1. FEDEP Introduction

With the fast development of HLA, many software engineering methods and procedures are used in the M&S industry. Most of them focus on the function development, concept evaluation and testing. However, the advantage of HLA that data can be exchanged dynamically at run-time is always ignored. As more and more simulations are migrating to HLA, there should be a framework to guide people when building HLA federations.

The Federation Development and Execution Process (FEDEP) model is a generalized process for building HLA federations [1]. It was designed to provide a high-level framework for HLA federation development and execution, rather than replace other engineering processes. It should also be aware that the FEDEP is not designed as a universal and common procedure for federation development. The reason is that different applications have different realities, for instance, the size, complexity and documentation requirements.

Since the needs and requirements of the simulation applications vary a lot, the FEDEP is designed as a starting framework for identifying and addressing the general issues for distributed simulation. It is also flexible in the process so that HLA applications could be composed to achieve the objectives of particular needs. To sum up, this framework is a foundation for all federation development, while it can be

tailored to design applications for specific purpose at the same time.

### 2.2.2. Six-Step Process

Although the FEDEP can be different among different federation builders, these processes can be summarized into several basic steps in the theoretical level. The FEDEP defines six steps for these processes, though the actual implementation may not be restricted to them [1]. Table 2-2 summarizes the six steps in sub-tasks. These steps will also be discussed below.

| Index | Steps | Sub-Tasks |
|---|---|---|
| Step 1 | Define Federation Objective | • Identify needs<br>• Develop objectives |
| Step 2 | Develop Federation Conceptual Model | • Develop Scenario<br>• Perform conceptual analysis<br>• Develop federation requirements |
| Step 3 | Design Federation | • Select federates<br>• Allocate functionality<br>• Prepare Plan |
| Step 4 | Develop Federation | • Develop FOM<br>• Establish federation agreements<br>• Implement federate modifications |
| Step 5 | Integrate and Test Federation | • Plan execution<br>• Integrate federation<br>• Test federation |
| Step 6 | Execute Federation and Prepare Results | • Execute federation<br>• Process output<br>• Prepare results |

**Table 2-2 The FEDEP Six Steps**

- ***Step 1 Define Federation Objective:*** As a beginning, the federation user and federation development team should identify the basic needs and requirements of the federation. They should also give a detailed objective statement in this step.

- ***Step 2 Develop Federation Conceptual Model:*** A function specification is developed in this step, followed by a representation of the real problem space. The fidelity requirements of the federation are then identified.
- ***Step 3 Design Federation:*** The participant federates are selected from all resources and their functionality and responsibilities are determined. A coordinated plan is prepared and documented.
- ***Step 4 Develop Federation:*** The Federation Object Model (FOM) is developed in this step, together with some agreements on the details of the software, database and algorithm to be used. Modifications that are necessary are implemented as well.
- ***Step 5 Integrate and Test Federation:*** In this step the plans for the whole execution and testing process and the measurement for evaluation are prepared. All participant federates including their software and hardware are integrated and installed. The interoperability among the participants is then tested.
- ***Step 6 Execute Federation and Prepare Results:*** The federation is executed and the simulation data are collected. Outputs are generated and analyzed if necessary. The objectives of the federation are checked whether they have been met or not and if the answer is yes, all the federation products are saved and kept for future reuse.

### 2.2.3. Implementation

This six-step process can be implemented in many different ways depending on the nature of the application [1]. This is because a lot of issues and requirements of an HLA federation could vary significantly. For example, the time and effort requirement, the degree of formality, personnel requirements, etc. can vary greatly from application to application, so that the procedures in developing the models can be quite different.

The degree of reuse of existing federation products may affect the implementation part, too. In some cases, federations may be developed largely from a scratch. Thus, the development process is rather new and time consuming. In other cases, users would like to follow long-standing requirements and care about the extensibility for each new product. In these situations, reusing is more often adopted and therefore both cost and development time is saved.

The six-step process provides a top-level view of the FEDEP, a comprehensive, generalized framework for HLA federation construction. However, users must be aware that during implementation, this process model will normally need to be adjusted and modified as appropriate to address the unique requirements and constraints of their particular application area.

## 2.3. Fidelity & Interoperability Challenges

A computer simulation is designed to represent some behavior of some things in the real world. However, no one computer simulation can simulate the objects in the real

world 100% accurately. The simulation fidelity is defined as “the accuracy of the representation when compared to the real world” [15]. It reflects how well the simulation responses and results correspond to what the simulation represents. The model designers should consider the fidelity before they actually build a model.

Another problem will occur if a model designer wants to reuse some models developed by others: the model designer cannot ensure the models from elsewhere can fit into his. This is defined as substantive interoperability, “the capability of federates, when connected, to provide adequate, accurate and consistent simulated representations that adhere to the principles of ‘fair fight’ and address the mission objectives” [16].

In order to produce meaningful simulation results, the entities represented across the federation must work together in a manner consistent with the needs of the federation application. A model designer should be aware of the following situations: Representational Anomalies, Functional Dependencies and Manifold Representations [17] so that the execution will adequately accomplish the mission.

### 2.3.1. Representational Anomalies

Representational anomalies are those states and events that would not occur in the *stimuland* (the real system being simulated by a simulation) under identical conditions. Whenever an anomaly occurs, it indicates that a simulation has omitted or incorrectly represented some aspects of object coupling that exist in the physical world.

There are mainly four kinds of representational anomalies which were defined in [17] and listed below:

- ***State Error Anomalies***: state error anomalies occur when there exists a difference between the state that a simulated object assumes and the state that object's *referent* (a codified body of knowledge about the thing being simulated) assumes under identical conditions and that difference is beyond levels tolerable by the application.
- ***Event Ordering Anomalies***: Event ordering anomalies occur when a simulated object produces the same events that the simuland would under identical conditions but in a different order.
- ***Event Phase Anomalies***: Event phase anomalies occur when a simulated object produces the same events in the same order that the simuland would under identical conditions but with a timing or phase error.
- ***Registration Anomalies***: Object state registration anomalies occur when the simulated states of two coupled objects differ from what the states of their coupled simulands would under the same conditions.

### 2.3.2. Functional Dependencies

Functional dependencies occur when the computation of one or more object states in one simulation depend upon the result produced by another simulation. Figure 2-3 illustrates an example of functional dependencies: the result of Simulation A is a dependent variable of the result of Simulation B, which means that the latter can

only be calculated after the former is ready.

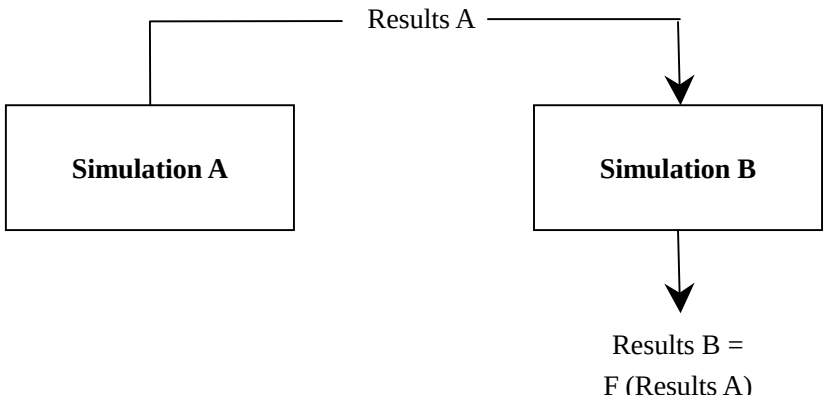


**Figure 2-3 Functional Dependencies between Two Simulations**

Function dependencies would bring some problems if there are some differences in the dependent variable between the two simulations. For instance, the result might be wrong if the measure of unit is different in the two simulations, or, it may cause exceptions if the result of Simulation A is out of range in the function that uses the result in Simulation B.

### 2.3.3. Manifold Representations

Manifold representations occur when two or more interacting simulations represent the same state or behavior of the same object. They are often used to reduce communication between the simulations, for example, the dead reckoning algorithm.

In some simulations, different objects may wish to know the positions of other objects. To reduce the communication cost among the objects, the dead reckoning algorithm is used [18]. In the algorithm, each object publishes its starting position and velocity vector only, and all the other objects will calculate the current position

themselves. Thus, there is no need to exchange the large amount of information in every time step.

# Chapter 3 Model Construction Environment

The Net-based Modeling and Simulation platform (NetMas) [2] project was initiated in 2001, and aimed at building simulation models more efficiently. A major new idea about the model building process is to reuse the existing simulation models through the network. Based on this concern, one of the main parts of the Netmas is to set up an environment for the users to select some existing model/sub-models from the resource library to build a new model.

The Model Construction Environment (MCE) is designed to meet this goal. It shows all the shared candidate simulation models which are documented in XML format. All the selected models should be carefully examined through a compatibility check before a new model is created. The compatibility checker is defined as the Matching Algorithm in this work and will be discussed in detail in chapter 4. The MCE also provides the users with a means to save the newly created model for future reuse.

This chapter introduces the MCE in design space approach and explains how it works. The DIF-XML converter which is used to generate the models in XML format is also introduced in this chapter.

## 3.1. Infrastructure

As mentioned in chapter 1, the HLA OMT is selected as our standard to describe the simulation models. Thus, the “simulation model” appearing in this thesis refers to

the HLA FOM/SOM.

The HLA OMT data interchange format (DIF) is a standard file exchange format used to store and transfer HLA FOMs and SOMs between FOM/SOM builders. The simulation models used in this work are all in the DIF format.

The Extensible Markup Language (XML) [19] is the universal format for structured documents and data on the Web. XML is a meta-language, a language for describing custom languages or formats. From the day of being invented, XML has been widely used to describe object models. It is worth investigating of the possibility to structure OMT DIF based on XML.

### 3.1.1. XML Based DIF

Many researchers have studied the feasibility of using XML to define OMT DIF [20, 21]. Basically, XML has several advantages as presented below:

Firstly, XML is supported by many Commercial Off-the-shelf (COTS) software applications and libraries. It is a noticeable fact that during simulation model development and analysis, various different HLA tools might use the data that is exchanged among simulations. As XML is proven to be an effective tool for data interoperability, more and more COTS software have been built by simulation industry to support the data interchange among simulations [20]. Therefore, HLA developers can focus on developing HLA FOM/SOMs and do not have to worry about creating interoperable tools.

Secondly, XML provides users with means to validate the format of the model. XML

documents are described in Document Type Definition (DTD) or schemas. XML tools can automatically validate the format of a compliant XML data file with the DTD/schemas. With the validation scheme, the syntax of the FOM/SOMs is ensured [21]. For example, the validation scheme will check whether required elements are present and associated information with that element is provided.

Thirdly, XML has been recommended to IEEE as the standard for HLA DIF descriptions. XML DIFs are being developed for many HLA-related data application, e.g. the Unit Order of Battle (UOB) DIF [22]. Thus, the FOM/SOMs in this work are stored in XML format based on DIF. Part of the DTD to describe the HLA object definition for the XML based DIF is listed in Figure 3-1.

```
<!ELEMENT Class (ID, Name, MOMClass?, PSCapability, Description?,
   SuperClass?, Attribute*)>
   <!ELEMENT ID             NMTOKEN>
   <!ELEMENT Name           NMTOKEN>
   <!ELEMENT MOMClass       (TRUE | FALSE)>
   <!ELEMENT PSCapability   (P | S | PS | N)>
   <!ELEMENT Description    CDATA>
   <!ELEMENT SuperClass     NMTOKEN>
   <!ELEMENT Attribute (Name, DataType, Cadinality?, Units?,
      Resolution?, Accuracy?, AccuracyCondition?, UpdateType?,
      UpdateCondition?, TransferAccept?, UpdateReflect?,
      Description?, RoutingSpace?)>
      <!ELEMENT Name              NMTOKEN>
      <!ELEMENT DataType          NMTOKEN>
      <!ELEMENT Cadinality        CDATA>
      <!ELEMENT Units             CDATA>
```

```
<!ELEMENT Resolution         CDATA>
<!ELEMENT Accuracy           CDATA>
<!ELEMENT AccuracyCondition  CDATA>
<!ELEMENT UpdateType (Static | Periodic | Conditional)>
<!ELEMENT UpdateCondition    CDATA>
<!ELEMENT TransferAccept     (T | A | TA | N)>
<!ELEMENT UpdateReflect      (U | R | UR)>
<!ELEMENT Description        CDATA>
<!ELEMENT RoutingSpace       NMTOKEN>
```

**Figure 3-1The HLA Object Class DTD Definition**

### 3.1.2. DIF-XML Converter

The original DIF files are usually stored in ASCII text files. Thus, these files need to be converted to XML format before we can use them in this work. To solve the problem, a DIF-XML converter is built and tested.

The DIF and XML format are similar in that both of them can be deemed as tree structure. For the nature of the DIF files, the non-terminals are translated into XML elements and the terminals are translated to XML texts. A stack is used to record the XML element string so that it can be reached once the element finishes.

However, the XML format is different from the DIF that only one top level element is allowed in an XML document while the DIF is not restricted to that. One solution is that for each DIF file, a top level element is added when it is converted to XML file. The tag of the top level element comes from the file name (the file suffix is excluded) from the original file.

The conversion procedure is summarized in Figure 3-2:

```
void convert ()
   read(file, buffer, filelength)  //read the original file
   while !(end of file)
      switch (*buffer)
         case ‘(‘ : //the beginning of a non-terminal
            string = the string that follows the ‘(‘
            write_string(“<”+string+”>”)
            //generate an XML element
            stack.push(string) //put the tag in stack
         case ‘)’ : //the end of a non-terminal
            write_string(“</”+stack.top()+”>”)
            stack.pop()  //pop out the tag
         case ‘”’ : //the text string defined in the DIF
            find out the next ‘”’
            write_string(the contents between the quotes)
         default : //the contents of the rest DIF
            write_char(buf)
      buffer++
End procedure
```

**Figure 3-2 DIF-XML Conversion Procedure**

The DIF-XML converter is built with C++ under Windows platform. It is an executable program, as illustrated in Figure 3-3. To use this converter, a user should first press the “Open” button and select a source DIF file. The file is then opened and the contents of the file will be displayed as well. By clicking the “Save As” button, the DIF file is converted to XML format and the user can choose a directory path to save the converted file. Figure 3-4 illustrates the windows for saving the file.

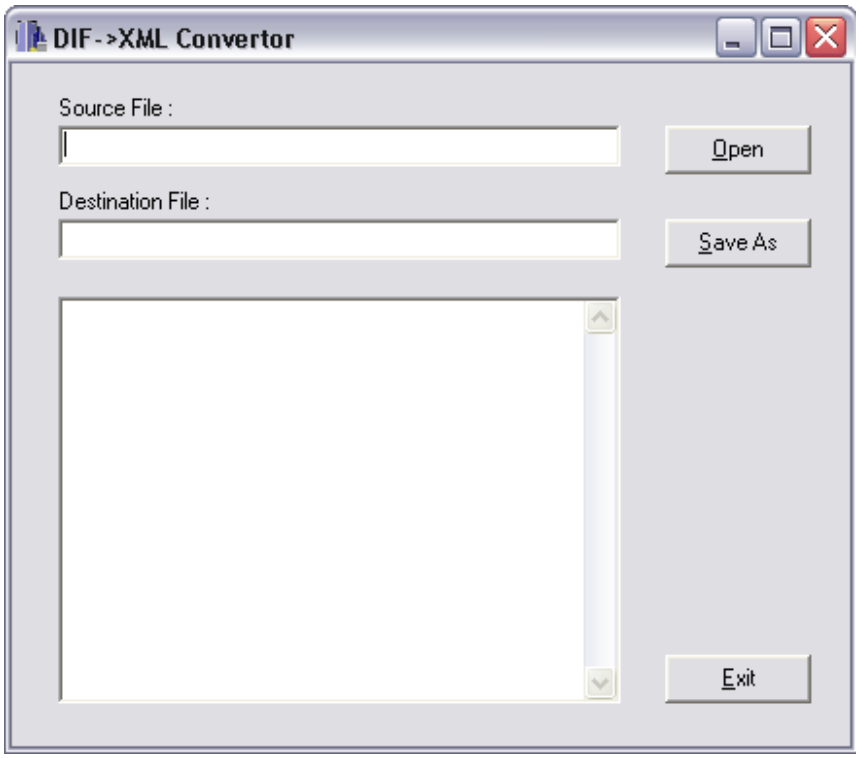


**Figure 3-3 DIF -XML Converter - Initialized**

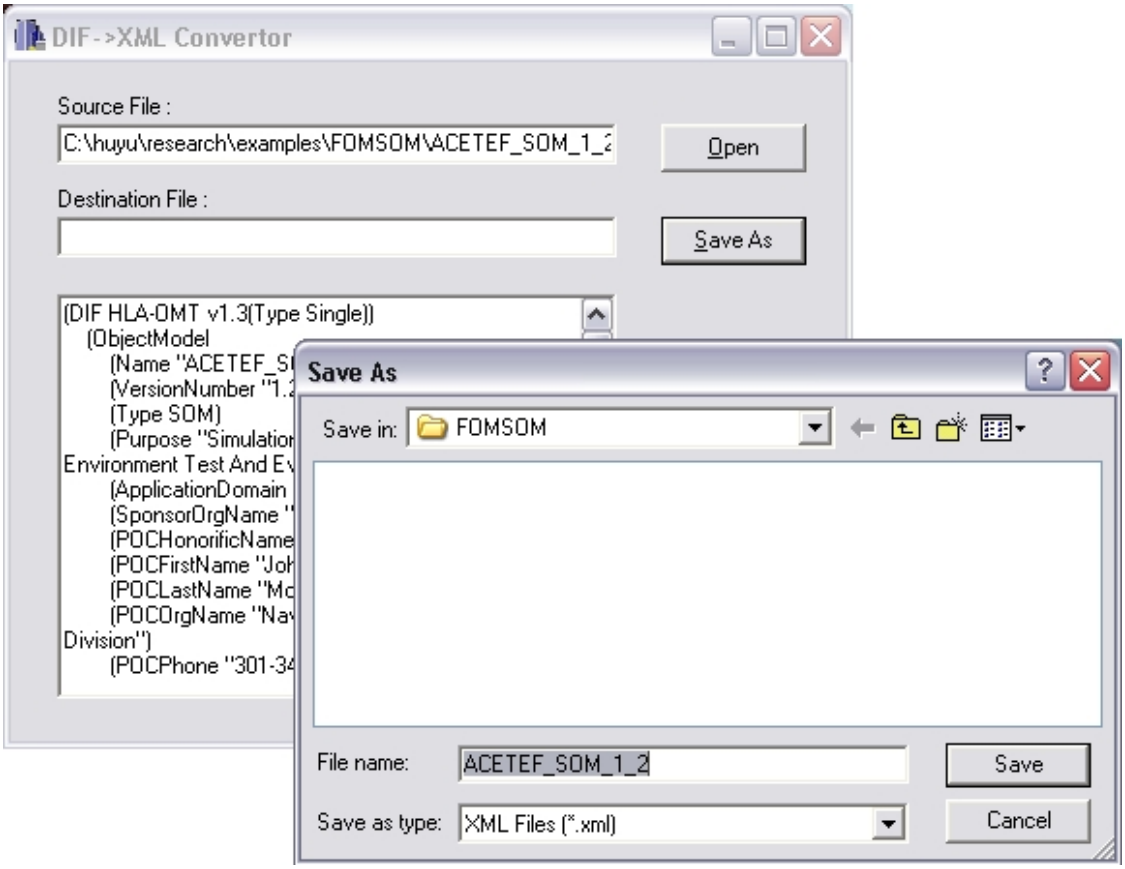


**Figure 3-4 DIF-XML Converter – Save As**

Figure 3-5 illustrates part of a SOM file which is generated by the DIF-XML

converter.

```
<?xml version="1.0" ?>
<ITBSOM__1_0V2>
<DIF>HLA-OMT v1.3<Type>Single</Type></DIF>
   <ObjectModel>
      <Name>ITBSOM</Name>
      <VersionNumber>1.0V2</VersionNumber>
      <Type>SOM</Type>
      <Purpose>To implement subset of the RPRFOM as defined in
the GRIM </Purpose>
      <ApplicationDomain>  Integrated  Test  Bed  Simulations
</ApplicationDomain>
      <SponsorOrgName>Simulation  Interoperability  Standards
Organization </SponsorOrgName>
      <POCHonorificName>The</POCHonorificName>
      <POCFirstName>Ken</POCFirstName>
      <POCLastName>Stauffer</POCLastName>
      <POCOrgName>AFRL/IFSC</POCOrgName>
      <POCPhone>937-255-4827 X3341</POCPhone>
      <POCEmail>kenneth.stauffer@wpafb.af.mil</POCEmail>
      <ModificationDate>10/28/1999</ModificationDate>
      <EnumeratedDataType>
         <Name>HLAboolean</Name>
         <Enumeration>
            <Enumerator>HLAtrue</Enumerator>
            <Representation>2</Representation>
         </Enumeration>
         <Enumeration>
            <Enumerator>HLAfalse</Enumerator>
```

```
            <Representation>1</Representation>
         </Enumeration>
      </EnumeratedDataType>
      <Class>
         <ID>4</ID>
         <Name>HLAobjectRoot</Name>
         <MOMClass>true</MOMClass>
         <PSCapabilities>PS</PSCapabilities>
         <Description>N/A</Description>
         <Attribute>
            <Name>HLAprivilegeToDeleteObject</Name>
            <DataType>any</DataType>
            <Cardinality>1</Cardinality>
            <UpdateType>Conditional</UpdateType>
            <UpdateCondition>N/A</UpdateCondition>
            <TransferAccept>N</TransferAccept>
            <UpdateReflect>UR</UpdateReflect>
            <Description>N/A</Description>
         </Attribute>
      </Class>
...
```

**Figure 3-5 Part of a SOM Example**

As one can see from Figure 3-5, every component of the original DIF file is transferred into XML format here. In this example, the SOM covers the contents in the Object Model Identification Model, an Enumerated Data Type definition and an object class with one attribute. This is just an example of part of the SOMs that are used in the MCE. The actual SOMs are bigger and more complicated than what is shown here.

## 3.2. MCE Framework

The main purpose of the NetMas project is to share the simulation models with people in different locations. In order to promote sharing through the network, the MCE should have the world wide accessibility. An easy solution is to build the MCE as a Java applet. However, saving the newly created model needs the file accessibility to the file system which is forbidden in Java applet for network security reasons. Thus, the MCE is designed as a client-server framework: a user interface built in Java applet on the client side and a Java application on the server side.

Figure 3-6 illustrates the relationship and the data flow chart between the two parts.

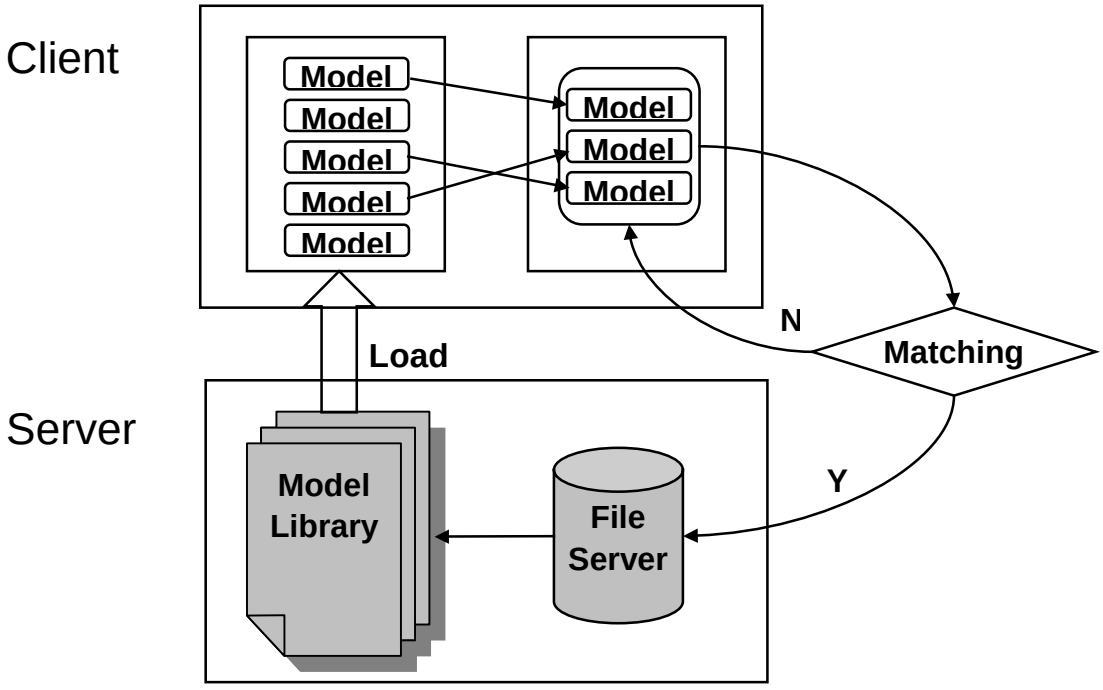


**Figure 3-6 The MCE framework**

As one can see from Figure 3-6, the client interface first loads all the models from the model library and lists them before the user. Then the user decides which models he needs and puts them together in the model pool. Later on, the models selected

will go through a Matching Algorithm to check for compatibility. If the matching result is false, the user should be informed of the details where the models are not compatible, and some of the models will be either modified or replaced to meet the user's desire. But if the matching result is successful, a new model is created and if the user wants to, the new model will be saved on the server side for future reuse.

### 3.2.1. User Interface

The user interface is the platform for the users to construct FOMs with existing SOM/FOMs. Figure 3-7 shows the User Interface when initialized:

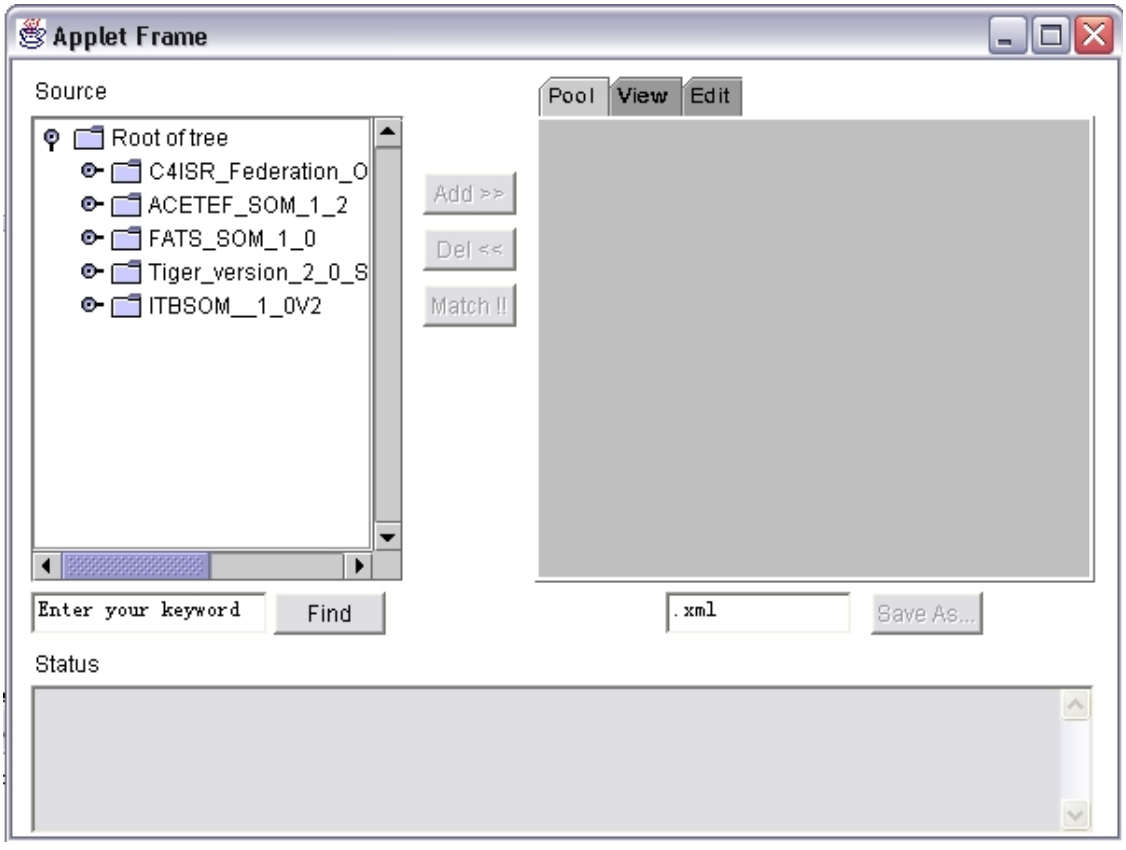


**Figure 3-7 The User Interface – Initialized**

As one can see from Figure 3-7, the user interface is primarily made up of four parts. The Source panel on the upper left side is a place to display all the models in the

model library. The tabbed panel on the upper right side is a container for the user selected models. The three function buttons between the two panels are used to manage the components of the new FOM. The Status panel on the bottom is used to record the result and/or error status during the matching process.

#### 3.2.1.1. Source Panel

The Source panel shows all the existing models in the library. It is designed in the Tree View mode. Every model (in XML file) is converted to a tree node of the root of the tree. Every child node of the model comes from the elements of the XML file. The nodes can either be expanded for viewing the details, or be collapsed as single nodes. The Source panel with the models is illustrated in Figure 3-8:

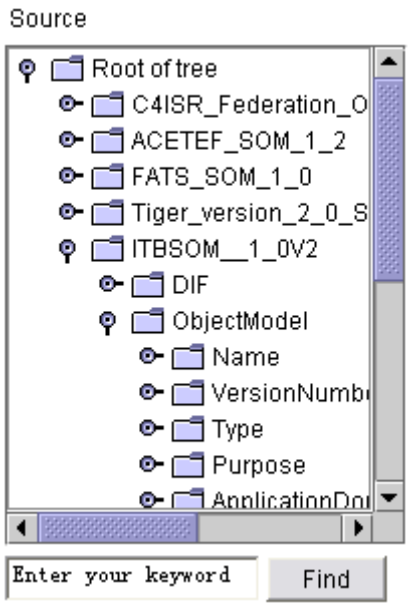


**Figure 3-8 The Source Panel – Model Library Tree View**

Meanwhile, a search function is provided for the user to find the model/component that he needs. This search function allows keyword search; any words that match the keyword will be set on the focus, no matter where it is in the model. By filling in the

edit box with the model/component's name and pressing the Find button, the user can easily select an appropriate model/component.

#### 3.2.1.2. Tabbed Panel

The Tabbed Panel is a container that contains the models selected by the user temporarily. It has three tabs altogether, namely, the Pool tab, the View tab and the Edit tab.

The Pool tab displays the existing models in the container in icon mode. When a model in the pool is clicked by the user with the right mouse button, a popup menu will appear. In the popup menu, the Remove menu item allows the user to remove the model which the user is clicking on; the View menu item allows the user to view the model in XML file format in the View tab; the Edit menu item allows the user to edit every item of the model in the Edit tab. The Pool tab is illustrated in Figure 3-9:

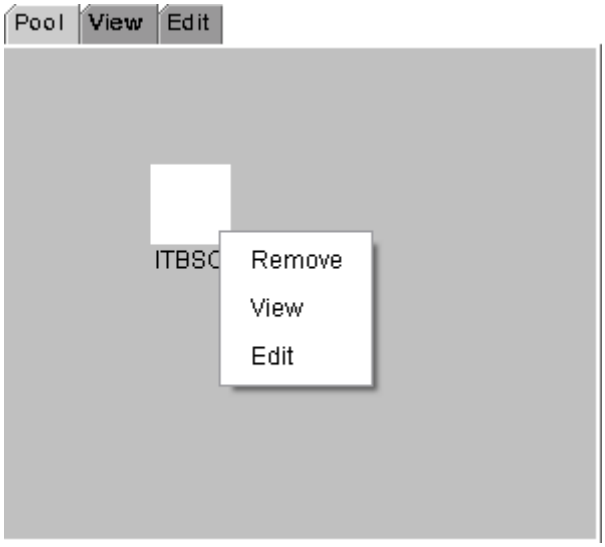


**Figure 3-9 The Tabbed Panel – Add a Model**

The View tab allows the user to view the selected model in XML file format. It is illustrated in Figure 3-10:

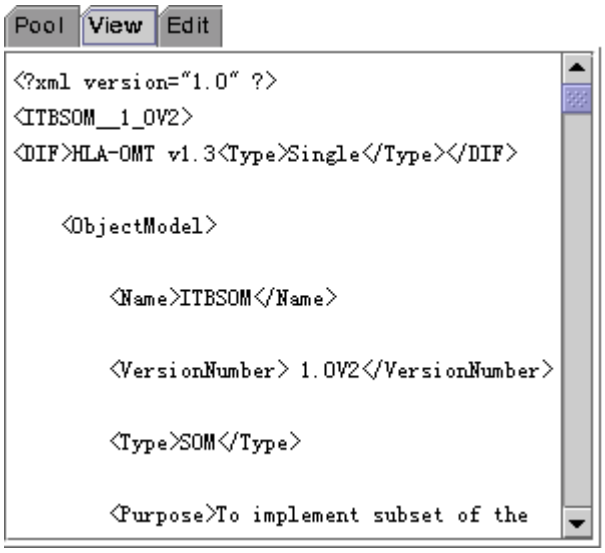


**Figure 3-10 The Tabbed Panel– View XML Format**

The Edit tab allows the user to edit the selected model in tree view mode. It is illustrated in Figure 3-11:

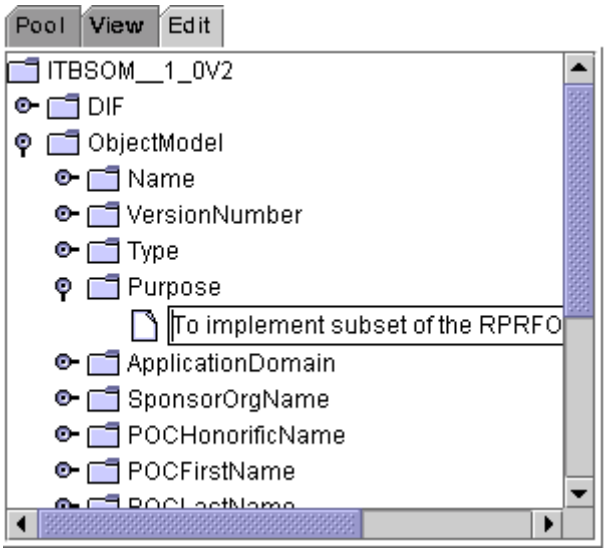


**Figure 3-11 The Tabbed Panel– Edit an Item in the Model**

If the result of the compatibility check is a match, the user shall make the decision to save the newly created model or not with a Save Confirm dialog. Figure 3-12 illustrates the Save Confirm dialog.

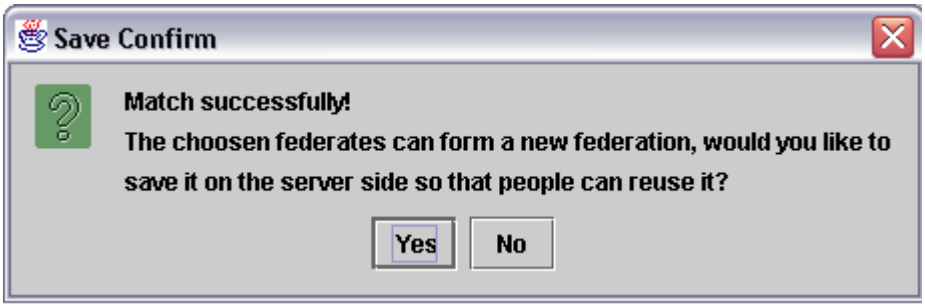


**Figure 3-12 The Save Confirm Dialog**

If the user wants to save the new model, he/she should enter a file name for the new model in the edit box and then send the save request to the server by clicking the Save As button. The edit box and the Save As button is illustrated in Figure 3-13:


**Figure 3-13 The Edit Box and the Save As Button**

#### 3.2.1.3. Function Buttons

There are three function buttons used to manage the components of the new FOM. They are located between the Source Panel and the Tabbed Panel. The Add button enables the user to add one model into the container. It is enabled only after a model in the Source Panel is selected. The Del button enables the user to remove one model from the container. It is enabled only after a model in the Tabbed Panel is selected. As long as there are more than two models in the container, the Match button is enabled and when it is pressed, the models will be checked using the Matching Algorithm.

#### 3.2.1.4. Status Panel

The Status panel reports the match result or mismatch status during the compatibility

check. The match result means the models being checked are compatible with each other, while the mismatch status tells the user with which feature the models are not compatible so that the user can either modify or replace the models. Figure 3-14 shows an example of the failure status.

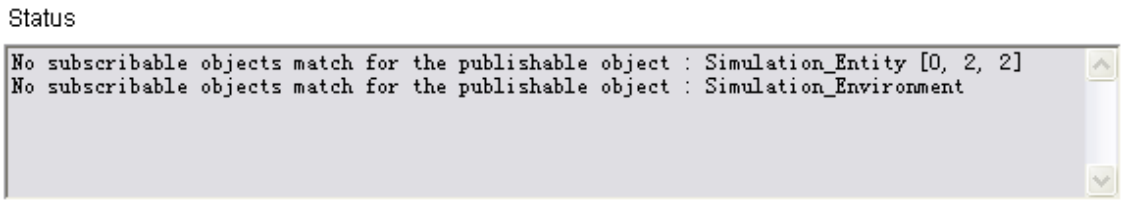


**Figure 3-14 The Status Panel**

### 3.2.2. File Server

The server side application is used to listen to the save request from the interface and to perform the save action. It is designed to run in a loop to capture the request from the interface at any time. Once it captures a request, it first accepts the tree node which stands for the new model sent from the client side, then converts it into the XML DIF format. Because the application is running on the server side, it can easily save the contents of the new model as a file through random file access. If being successfully saved, the new model can be an addition the existing model library.

### 3.2.3. Client-Server Communication

The client and server programs communicate through the socket scheme. They are designed to communicate through a specified port. The client side initiates the socket connection whenever the user wants to save the new model. It first creates an I/O stream, then writes the file name as a string and the model as an object into the

stream, and finally sends out the stream.

Correspondingly, the server side listens to the network and gets one stream at a time. It analyzes the stream and gets the file name and the model object respectively. Then it works as described in section 3.2.2. The accept method of the server socket is put into a circulation which never stops so that the server side application is ready to save a newly created model all the time.

## 3.3. Summary

The Model Construction Environment (MCE) in this work is a platform for users to develop new simulation models with those existing ones in the model library. This chapter introduced the MCE from both the infrastructure and the framework aspects.

The Data Interchange Format (DIF) is selected as data format to store the FOM/SOMs. XML has been suggested to IEEE as the standard format for HLA DIF, XML has noticeable advantages which are explained in this chapter, the XML based DIF is adopted in the MCE as the representation format of FOM/SOMs. To convert the DIF to XML format, a DIF-XML converter was built and tested.

The MCE constitutes a user interface applet and a file server application. The user interface allows users to view and edit models, as well as to check the compatibility among some selected models based on the Matching Algorithm. The file sever performs the save action for newly created models.

The web-based MCE can be a useful tool to develop HLA FOM automatically. Its world wide accessibility can further strengthen the reusability of the simulation

models.

# Chapter 4 The Matching Algorithm & FOM Development

The MCE provides the users with an interface to construct new models with the existing ones. However, a critical condition for this scheme to work smoothly is that all the sub-models in the new model must be compatible. Also the data of different sub-models must be parsed and be ready to be integrated in the new model. This is the job of the Matching Algorithm: it checks the compatibility among the models, presents the result, and then prepares data for new model development. As the models used in this work are FOM/SOMs based on the HLA OMT, the theoretical foundation of the Matching Algorithm is based on the HLA OMT and the FEDEP Model.

This chapter can be divided to two parts: one part for the Macthing Algorithm and the other for FOM development. The Matching Algorithm is made up of four consecutive processes: the DataType Check, the Routing Space Check, the Object Match and the Interaction Match. A match is successful only if all of the four processes results are successful. In other words, failing in any of the four processes will lead to a mismatch and the data/elements that led to the mismatch will be reported. If the result is not a mismatch, a new FOM can be created using the data collected from the Matching Algorithm. This chapter also explains how the Matching Algorithm is enhanced with some ideas from the FOM agility concepts.

## 4.1. DataType Check

### 4.1.1. Base Data Types & User-defined DataTypes

The DataType defined in HLA OMT is used to reference the data type of the OMT elements. The DataType may be chosen from the list of permissible base data types, or it may be a user-defined DataType. There are thirteen base data types defined in the OMT, namely “unsigned short”, “short”, “unsigned long”, “long”, “unsigned long long”, “long long”, “double”, “float”, “boolean”, “any”, “string”, “char” and “octet”.

User-defined DataTypes, including the Enumerated DataType and the Complex DataType, are supplementary formats used to help document the structure and content better. Their names should be different from the names of the base data types. The Enumerated DataTypes describe the data types whose values could only come from a finite discrete set of possible values, for example, the seven days of the week. The data types should be completely documented with every enumerator and its representation of the enumerations. The Complex DataTypes describe those complex data types which aggregate other DataTypes into a structure. They are made up of several complex components in which detailed information such as data type, cardinality, units, accuracy, etc. are documented.

### 4.1.2. DataType Check Procedure

One of the first steps for the matching algorithm is to ensure the consistency in the

data type definition among all federates in the federation. Both the Enumerated DataTypes and the Complex DataTypes are checked during this procedure. The rule for checking is that whenever user-defined data types with the same name occurred in the federation, check whether the enumerators and representations of the data type (for enumerated data types) or components (for complex data types) are accordingly the same.

The Enumerated DataType Check process is organized as follows. At the beginning, an Enumerated DataType array is initialized with the number of Enumerated DataTypes in the models. Then the tree node in the model pool is parsed. Whenever an Enumerated DataType node is found, the check process examines whether this DataType conflicts with those that are already in the array. If a conflict occurs, the check will return false which indicates the result is mismatch. And if no conflict occurs, this DataType will be appended to the end of the array. Figure 4-1 records the pseudo-code of the Enumerated DataType Check Procedure as an example of the DataType Check.

```
int Parse_EDataType (TreeNode node)
   If (node is a EnumeratedDataType)
      For each i in the EDataType record
      //check if any conflict exists
         If (node.Name = EDataType[i].name)
            If !(node equals to EDataType[i]) return -1
            //return error
      EDataType[i] = new EDataType(node)
      //record this Enumerated DataType
```

```
   Else for each child j of node
      Parse_EDataType (node.getChildAt(j))
      //recursion to traverse the entire tree
   Return 0           //successful return
End procedure
```

**Figure 4-1 The DataType Check Procedure**

As the Complex DataType Check procedure is similar to this one, it is skipped from explanation.

## 4.2. Routing Space Check

The Routing Space is defined in the HLA Data Distribution Management (DDM) services [13]. It is a multidimensional coordinate system for federates to either express an interest in receiving data or declare an intention to send data.

### 4.2.1. Dimension, Region & Extent

In DDM, several key conceptions are defined:

- ***Dimension:*** A Dimension is a measure of the routing space. The routing space is a collection of dimensions.
- ***Extent:*** An Extent is a bounded range defined across the dimensions of a routing space. It represents a volume in the multi-dimensional routing space.
- ***Region:*** A Region is a set of extents with the same interest in receiving data or sending data.

Regions can be further divided into two types as

- ***Subscription Regions:*** sets of routing space extents that narrow the scope of interest of the subscribing federate
- ***Update Regions:*** sets of routing space extents that are guaranteed to enclose an object's location in the routing space

In HLA OMT, the Routing Space specifies the particulars of routing space and the dimensions inside. The dimensions shall form the parameter space in which update and subscription regions shall be specified by the federates to the RTI. By having this agreement about the meaning of routing space dimensions, the RTI can calculate the intersections of update and subscription regions efficiently without having to understand the semantics of the dimensions.

### 4.2.2. Routing Space Check Procedure

During the development of an HLA federation, it is critical that all federation members achieve a common understanding of DDM routing spaces and their semantics, and agree to a common set of routing space specifications. These agreements are necessary for federates to filter object attribute updates and interaction in a correct and consistent manner.

Just like the DataType check, we should ensure that the definitions of the routing spaces in different members of the federation are consistent. This is done similarly to the Data Type check that whenever two routing spaces have the same name, check whether all the dimensions and the contents inside are the same for the two correspondingly. The routing spaces are also recorded in the memory in the form of

array.

## 4.3. Object Match

### 4.3.1. HLA Object

In HLA OMT, objects are defined entirely by identifying characteristics that are exchanged between federates during execution. An HLA object class is a collection of objects with certain characteristics or attributes in common [3]. It is the key component in the HLA OMT. The object class structure of an HLA object model shall be defined by a set of relations among classes of objects from the simulation or federation domain.

An HLA class structure shall also be defined in terms of hierarchical relationships among classes of objects. The class hierarchy is used to expand the capability of publication and subscription about broad super classes of objects [3]. Publication and subscription to the values of HLA object attributes are services provided by the RTI. By subscribing to all attributes of a specified object class, a federate is assured of receiving all value updates of attributes defined for that class and all of its super classes.

### 4.3.2. Publishable & Subscribable

There are several properties in the object class definition; one of the most important properties is the PSCapabilities, which indicates the publication and subscription capabilities for each object class. The possible values of this property are listed

below:

- ***Publishable (P):*** The specified object class can be published by a federate;
- ***Subscribable (S):*** A federate is currently capable of utilizing and (potentially) reacting to information on objects in the specified class;
- ***Publishable and Subscribable (PS):*** The object class is publishable, as well as subscribable by a federate;
- ***Neither Publishable nor Subscribable (N):*** The object class is neither publishable nor subscribable by a federate.

The publishable designation of an object is intended to allow federates to distinguish their internal capabilities for modeling objects of the associated classes as well as their ability to share information about such objects in an HLA federation. An object class shall be subscribable by a federate only if the federate can make substantive use of instances of the class when it is notified of them by the RTI. An object class shall not be subscribable by a federate if it always ignores instantiation notices and updates for objects attributes in that class.

### 4.3.3. Attributes

Each HLA object is characterized by a fixed set of attribute types. The Attributes are named portions of their object's state whose value can change over time. An HLA object model shall support representation of the following characteristics for attributes in the attribute table:

- ***Name:*** The name to identify the attribute;
- ***DataType:*** The data type of the attribute;
- ***Cardinality:*** Record the size of an array or sequence;
- ***Units:*** The units (e.g., m, kg, km) used for the attribute;
- ***Resolution:*** Specify the smallest resolvable value separating attribute values;
- ***Accuracy:*** Capture the maximum deviation of the attribute value from its intended value in the federate or federation;
- ***Accuracy condition:*** Contain any conditions required for the given accuracy to hold in a given simulation or federation execution;
- ***Update type:*** Record the update type for the attribute. It shall be specified as "static", "periodic" or "conditional";
- ***Update condition:*** Specify initial conditions for attribute updates;
- ***Transferable/acceptable:*** Record the information about being able to transfer or accept in attribute publication and subscription;
- ***Updateable/reflectable:*** Identify the current capabilities of a federate with respect to attribute updating and reflection;
- ***Routing space:*** Record the association of an object attribute with a routing space.

### 4.3.4. Objects Match Procedure

Whenever an object class is publishable in a federation, there shall be a class which subscribes to the object otherwise it is useless to publish the object. This is the main objective of the Objects Match Procedure.

To ensure every object which is published within the federation has at least one subscriber, we created two lists: the publishing list (P_List) and the subscription list (S_List). The goal is to check that for each object in the P_List, there exists at least one object in the S_List. As illustrated in Figure 4-2, the objects are checked with PSCapabilities and when the PSCapabilities of an object is:

- **<*P*>:** The object will be added into the P_List;
- **<*S*>:** The object will be added into the S_List;
- **<*PS*>:** The object will only be added into the S_List because even before matching we can tell that the object itself is a subscriber;
- **<*N*>:** The object will be discarded.

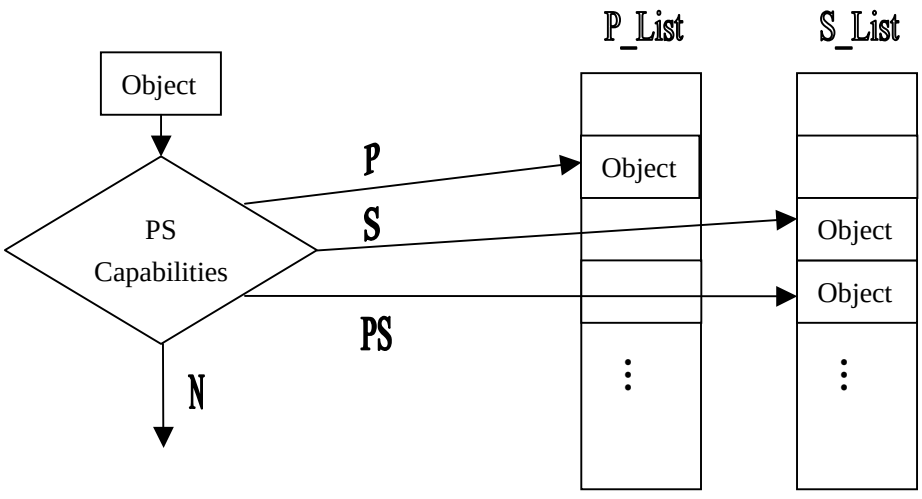

**Figure 4-2 P_List and S_List Construction**

After generating the two lists, the next step of the Object Match is to find at least one subscriber for each object in the P_List from the S_List. The subscriber should also have the same attributes characteristic definitions. Figure 4-3 illustrates the match procedure between two Objects.

```
boolean obj_match (TreeNode A, TreeNode B)
   //A and B stand for the two Object nodes respectively
   if !(A.Name = B.Name) return false
   for each Child in A      //each Child is an Attribute
      attr = getNode(B, Child.Name)
      //get the Attribute node with the same name in B
      for each properties in Child
         if !(properties = attr.properties)
            printErrorStatus(properties)
            //print the error status in the MCE
            return false
   return true           //successful return
End procedure
```

**Figure 4-3 The Match Procedure between Two Objects**

## 4.4. Interaction Match

### 4.4.1. HLA Interaction

An HLA interaction is defined as an explicit action taken by a simulated entity (or aggregation of entities) in one federate that may have some effect or impact on

another federate [3]. The interaction class structure is much the same way like the object class structure described in section 4.3.1.

Interactions are one of the principal determinants of interoperability among simulations. Interoperability ordinarily requires some consistency in the treatment of interactions afforded by the different federates in which they appear. In addition, the publication and subscription of the interactions are also involved in a simulation execution.

### 4.4.2. Initiates, Senses and Reacts

Similar to the PSCapabilities designation provided in the object class structure, the interaction class structure also provides certain designation of federate/federation capabilities with respect to give classes of information. It is called ISRType and the possible values are listed below:

- ***Initiates (I):*** Indicates that a federate is currently capable of initiating and sending interactions of the given type;
- ***Senses (S):*** Indicates that a federate is currently capable of subscribing to the interaction and utilizing the interaction information, without necessarily being able to effect the appropriate changes to affected objects;
- ***Reacts (R):*** Indicates that a federate is currently capable of subscribing and properly reacting to interactions of the type specified by effecting the appropriate changes to any owned instance attributes of affected objects;

- ***Initiates and Senses (IS):*** Both initiates and senses;
- ***Initiates and Reacts (IR):*** Both initiates and reacts;
- ***Neither Initiates, Senses nor Reacts (N):*** Indicates that a federate is not currently capable of initiating, sensing, or reacting to this interaction class.

### 4.4.3. Parameters

Most interaction classes will also be characterized according to a list of one or more interaction parameters which are much similar to the object attributes. They are used to associate relevant and useful information with classes of interactions [3].

An HLA object model shall support representation of the following characteristics for each parameter: Name, DataType, Cardinality, Resolution, Accuracy and Accuracy Condition.

Unlike object attributes, interaction parameters may not be subscribed to on an individual basis. This implies that routing space information shall be specified at the interaction class level rather than at the individual parameter level.

### 4.4.4. Interaction Match Procedure

In a federation, at least one federate should sense or react to every interaction class that is initiated. This is the main concern of the Interaction Match Procedure.

As a solution, we use a similar scheme to the Object Match procedure to create the Initiates list (I_List) and Senses/Reacts list (SR_List). As illustrated in Figure 4-4 below, in this procedure we check every interaction with ISRType and when the

ISRType of an interaction is:

- ***<I>:*** The interaction will be added into the I_List;
- ***<S> or <R>:*** The interaction will be added into the SR_List;
- ***<IS> or <IR>:*** The interaction will only be added into the SR_List because even before matching we can tell that itself is a subscriber;
- ***<N>:*** The interaction will be discarded.

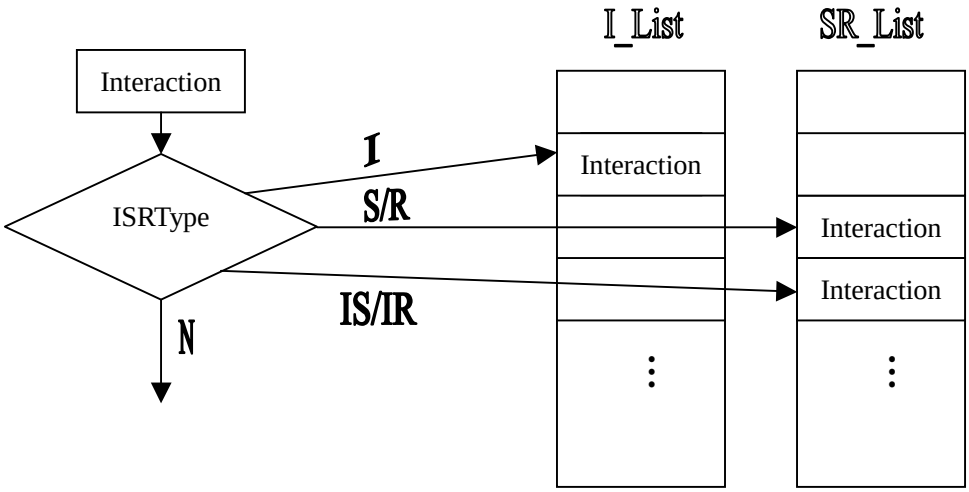


**Figure 4-4 I_List and SR_List Construction**

The result of interaction match procedure will be successful in the case that for each interaction in the I_List, there exists at least one interaction in the SR_List and the two interactions have the same parameter information and other properties details. The comparison between two interactions is also similar to the one between objects and will not be discussed in detail here.

## 4.5. FOM Agility Enhancement

Conceptually an HLA federate is able to participate in several federations. To further facilitate the reusability of federates, the modern trends for HLA federation development is to build several functionality independent federates first and select some of them to build overall federations. However, due to the differences in requirement and implementation of different federations, these federates may need to be modified to be compatible in the federations.

### 4.5.1. Agile FOM Framework (AFF)

To enable the general purpose SOM to participate in multiple FOMs, an Agile FOM Framework (AFF) is introduced in [23]. The AFF provides wide range of feature mappings from the SOM to destination FOM. These feature mappings include name independence, attribute atomicity, unit and coordinate conversions, defaulting and even attribute to object mapping [23]. After the mappings, the FOM developer should provide some converter as the implementation of the mappings.

The beauty of the AFF is that a SOM can be easily plugged into a FOM without any code changes in the SOM. This is achieved by inserting a RTI Interface Layer (RIL) between the RTI and the application. The FOM/SOMs are still running on RTI, but the exchange of data should go through the RIL, then they will be sent to RTI. Figure 4-5 illustrates where the RIL is located and how it works.

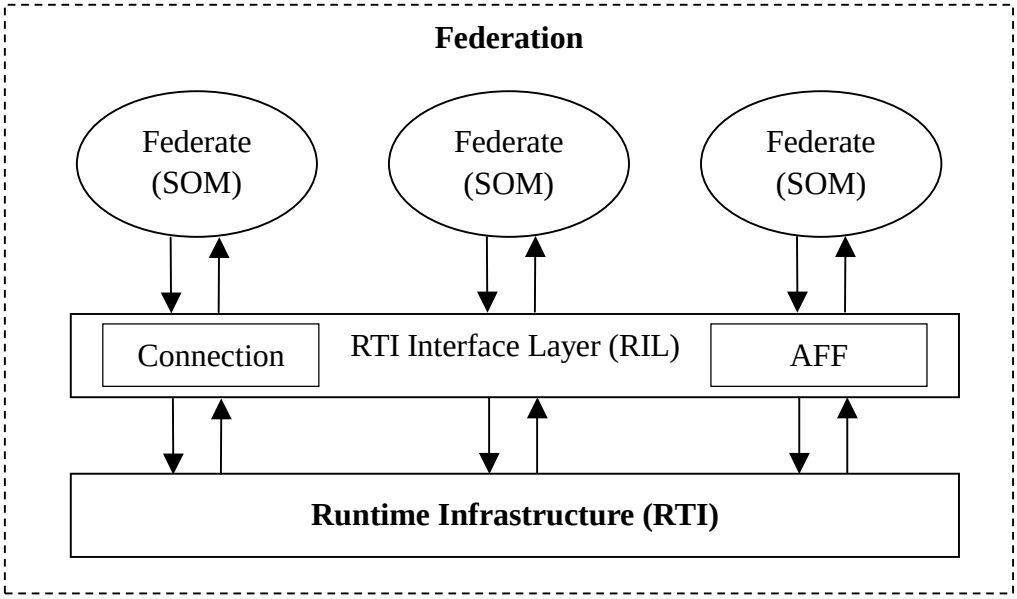


**Figure 4-5 The RIL Role between RTI and Applications**

The RIL consists of a set of C++ classes. There are two major functionalities for the RTI: one is to provide the connection between RTI and applications, the other is to make use of the AFF to support FOM Agility. FOM developers should build some converters which are also C++ classes and deploy them in the RIL. The converter development requirements and processes are introduced in detail in [23].

Apparently, the AFF does not guarantee a SOM can be mapped to any FOM. FOM developers should study carefully the features of the SOM that they want to reuse. It is unacceptable and meaningless to map some conceptually unrelated SOMs into the FOM.

### 4.5.2. Matching Algorithm Enhancement

The AFF defines some mapping capabilities to help FOM developers to reuse the general purpose SOMs. Even a GUI based tool has been developed to define and

configure the mappings [24]. To further facilitate reusability in the MCE, the concepts in the AFF should and can be adopted in the Matching Algorithm. The compatibility checkers inside the four processes of the Matching Algorithm are all involved.

However, differing from the AFF, the Matching Algorithm is an automatic approach. Therefore not all the mappings in the AFF can be implemented in the Matching Algorithm. Some enhancements of the Matching Algorithm based on the AFF are listed below:

1. **Name Independence**

   Naming is not a big issue in modeling the FOM/SOMs. As long as model developers are talking about the same thing, an attribute name of “velocity” in a SOM can be converted to the name of “speed” in the FOM. During the compatibility check, the name check result for the object/interaction match is no longer essential. The differences (if any) of naming are only displayed on the status panel in the user interface.

2. **Attribute Atomicity**

   When modeling complex attribute/parameters in the object/interaction, some developers would like to list them one by one, while others prefer define a complex DataType and include them all in one item. With the AFF, this is no longer a problem: a converter can map the two different scenarios. For example, Figure 4-6 and 4-7 illustrate two scenarios:

```
<Class>
```

```
   <Name>Coordinates</Name>
   <Attribute>
      <Name>Coordinate_X</Name>
      <DataType>double</DataType>
   </Attribute>
   <Attribute>
      <Name>Coordinate_Y</Name>
      <DataType>double</DataType>
   </Attribute>
   <Attribute>
      <Name>Coordinate_Z</Name>
      <DataType>double</DataType>
   </Attribute>
</Class>
```

**Figure 4-6 Scenario 1: Define Three-Dimensional Coordinates Directly**

```
<ComplexDataType>
   <Name>Three_Dimension</Name>
   <ComplexComponent>
      <FieldName>Coordinate_X</FieldName>
      <DataType>double</DataType>
   </ComplexComponent>
   <ComplexComponent>
      <FieldName>Coordinate_Y</FieldName>
      <DataType>double</DataType>
   </ComplexComponent>
   <ComplexComponent>
      <FieldName>Coordinate_Z</FieldName>
      <DataType>double</DataType>
   </ComplexComponent>
</ComplexDataType>
```

```
<Class>
   <Name>Coordinates</Name>
   <Attribute>
      <Name>Coordinate_Attribute</Name>
      <DataType>Three_Dimension<DataType>
   </Attribute>
</Class>
```

**Figure 4-7 Scenario 2: Define Three-Dimensional Coordinates with Complex DataType**

As one can see from these two scenarios, the definitions of the object attributes are different, but the semantics of the two are the same. In this circumstance, the two objects should pass the compatibility check in the Matching Algorithm. A warning of the difference will be displayed in the status panel.

**3. Unit Conversion**

The Units property defined in the attribute/parameter is a measurement of the data that are exchanged during simulation. Since the AFF can provide a wide range of converters mapping data from one unit to another, the unit differences caught in the Matching Algorithm can be ignored. The unit difference should also be displayed in the status panel.

The Matching Algorithm is altered with the three enhancements above. Table 4-1 summarizes these three enhancements and explains what are involved.

| Enhancements | Affected Elements | Matching Algorithm Involvement |
|---|---|---|
| Name Independence | “Name” for Enumerated DataType and Complex DataType | DataType Check |
| | “FieldName” for Complex Component | |
| | “Name” for Routing Space and Dimension | Routing Space Check |

| | | |
|---|---|---|
| | “Name” for Object and Attribute | Object Match |
| | “Name” for Interaction and Parameter | Interaction Match |
| Attribute Atomicness | Attribute definition for Object | Object Match |
| | Parameter definition for Interaction | Interaction Match |
| Unit Conversion | “Units” for Complex Component | DataType Check |
| | “Units” for Attribute | Object Match |
| | “Units” for Parameter | Interaction Match |

**Table 4-1 The Matching Algorithm Enhancement Involvements**

With the differences between the SOMs displayed in the status panel, it is up to FOM developers to implement the converters based on the AFF.

## 4.6. FOM Development

The Matching Algorithm checks the compatibility among the FOM/SOMs. If the result is not a mismatch, the next step would be the FOM development. As all the contents of the FOM/SOMs have been stored in the memory during the check, the FOM development process is rather easy. According to the DIF structure, the process consists of five steps:

1. **Provide the object model identification information**

   The FOM Developer should provide the object model identification information in the FOM. These information include the name, version, date, purpose, application domain, sponsor and POC (Point of Contact) information of the model.

2. **Create DataType information**

   The Enumerated DataTypes and Complex DataTypes that occurred in the models are recorded in the memory. This step takes out the DataType information (in java classes) and form the XML representation of these information.

### 3. Create Object classes

All the object classes have been stored in the P_List and S_List during the Matching Algorithm. The PSCapability of an object in the FOM can only be “PS” or “S” [3]. Since after the Matching Algorithm, each object in the P_List is ensured to have at least one subscriber in the S_List, we only create the object classes in the S_List. The creation of Object classes with the S_List is summarized in Figure 4-8:

```
void createObjectClasses ()
   for each object in S_List
      if (object.PSCapability == “PS”)
      //some “PS” objects only exist in S_List
         ToXML(object)
      else if (P_List contains object)
         object.PSCapability = “PS”
         //the PSCapability of the object should be “PS”
         ToXML(object)
      else if !(P_List contains object)
         ToXML(object)
         //the PSCapability of the object should be “S”
End Procedure
```

**Figure 4-8 The Object Classes Creation Process in FOM Development**

As illustrated in Figure 4-8, the “PS” objects are recorded with no doubt. Some objects which have publisher in the P_List are assigned “PS” and then are recorded as well. The rest of the objects in the S_List are recorded with “S”.

### 4. Create Interactions

The creation of interactions is similar to that of objects. Only the interactions in the SR_List are recorded. All the “IS” and “IR” interactions are recorded. The “S” interactions which have initiator in the I_List are assigned “IS”, the “R” interactions which have initiator in the I_List are assigned “IR”. Then these interactions are recorded. The rest of the interactions are recorded as what they are.

**5. Create Routing Spaces**

The Routing Space information is also stored in the memory. This step is similar to step 2.

After these five steps, a FOM is created. It is then sent to the file server and stored for future reuse.

## 4.7. Summary

This chapter introduced the Matching Algorithm which is used to check the compatibility among models. It was made up of four consecutive processes, namely DataType Check, Routing Space Check, Object Match and Interaction Match. The algorithm was then enhanced with some concepts from the Agile FOM Framework. The enhancements are believed to strengthen the reusability and compatibility of the models.

If the matching result is not mismatch, the user may want to create the FOM with the compatible models. The Matching Algorithm also provided the conveniences for the FOM development. The FOM development was also introduced in the chapter.

# Chapter 5 Results and Discussion

This thesis work built up the MCE system for automatic FOM construction and provided a Matching Algorithm for the compatibility check among the SOMs. We can use the MCE to build new models with existing ones. Because all the FOM/SOMs used in this work are based on the HLA OMT DIF standard, the system and the algorithm are based on the same standard too. This could work fine as they are on the same problem space. However, is the standard an accurate and omnipotent one that never generates wrong results?

This chapter investigates on two case studies with the MCE system and analyzes the results. Later on, the feasibility of the MCE is discussed and some deficiencies of the current scheme are given as well as the reasons behind them. Some limitations of the current HLA OMT are also investigated in this chapter.

## 5.1. Case Studies

There are altogether 10 FOM/SOMs in our model library currently. Among the models, 8 are SOMs and 2 are FOMs. These models are summarized in Table 5-1:

All the models in the library are available for federation construction. We choose some of them to check the compatibility among them. Two case studies are provided and described in the following sub-sections.

| Name | Type | Version | Purpose |
|---|---|---|---|
| ACETEF_SOM | SOM | 1.2 | Simulation Object Model for Air Combat Environment Test And Evaluation Facility. |
| C4ISR Federation Object Model - Combat | SOM | 1.0 | Provide a Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) Simulation. |
| Cobra Ball SOM | SOM | 1.3 | The Object Model for the Cobra Ball Simulator. Based upon the Real-time Platform Reference (RPR) FOM 0.5. |
| Computer Generated Forces (CGF) | FOM | 2.0 | A generic computer generated forces application. |
| EADSIM | SOM | 8.00 | SOM for Extended Air Defense Simulation (EADSIM). |
| FATS SOM | SOM | 1.0 | Reference SOM based on RPR FOM. This SOM has much of the FOM data removed and has a few FATS enhancements “environment interactions”. |
| ITBSOM | SOM | 1.0V2 | To implement subset of the RPR FOM as defined in the Guidance, Rationale and Interoperability Modalities (GRIM). |
| Mission Planning Simulator | SOM | 00.6.1 | Provide space-time location and orientation of simulated test objects and instruments to data collectors. |
| Restaurant FOM | FOM | 1.3 | To provide an example of an object model for a restaurant federate. |
| Tiger version 2.0 SOM | SOM | 1.0 | Tiger SOM entirely based on RPR FOM version 0.5. |

**Table 5-1 Summary of FOM/SOMs in the Model Library**

### 5.1.1. Combat Federation Case Study

One of the main application areas for the M&S is military simulation. In military institutions, the computer-driven combat simulation has the advantage in officer training, mission rehearsal and tactics exploration. The need for military simulations such as combat simulation is continually increasing [25]. In this case study we plan to build a federation for combat simulation. After carefully examining the existing

models in the library, we choose the Computer Generated Force (CGF) FOM and the EADSIM to build the federation.

CGF is defined in the U.S. DoD M&S Master Plan [12] as a generic term used to refer to computer representations of forces in simulations and to model human behavior sufficiently so that the forces will take some actions automatically. DoD programs addressing various levels of computer automation of forces include Command Forces, Intelligent Forces, Modular Semi-Automated Forces, Integrated Tactical Environment Management System, and Close Combat Tactical Trainer Semi-Automated Forces.

Extended Air Defense Simulation (EADSIM) is sponsored by U.S. Army Space & Missile Defense Command. It is a workstation-hosted, system-level simulation which is used by combat developers, materiel developers, and operational commanders to assess the effectiveness of Theater Missile Defense (TMD) and air defense systems against the full spectrum of extended air defense threats. EADSIM provides a many-on-many theater-level simulation of air and missile warfare, an integrated analysis tool to support joint and combined force operations, and a tool to provide realistic air defense training to maneuver force exercises at all echelons.

The compatibility check result of the two models is mismatch in the original Matching Algorithm. It is found that both models have the definition of a ComplexDataType under the name of RelativePositionStruct, while there are some minor differences between the two definitions of ComplexComponents. Table 5-2 presents the differences between the two definitions:

| Element | CGF FOM | EADSIM |
|---|---|---|
| FieldName | BodyX, BodyY and BodyZ | BodyXDistance, BodyYDistance and BodyZDistance |
| Units | Metres | Meters |

**Table 5-2 Differences in Definition of ComplexDataType RelativePositionStruct between CGF FOM and EADSIM**

However, these differences are no longer problems with the enhanced Matching Algorithm. The first difference is handled by Name Independence, while the second one is handled by Unit Conversion. The differences are only presented as warnings in the status bar. Thus, a new Combat Federation FOM is created and the model developer should have converters implemented under the AFF. This newly created FOM can be a practical implementation of CGF.

### 5.1.2. RPR FOM Case Study

The Real-time Platform Reference (RPR) FOM is a reference FOM for the Real-time Platform simulation community. It is a development effort to create a FOM that transforms the widely used real-time simulations based on the IEEE 1278 DIS standard to HLA compliance [26, 27]. This FOM keeps the functionality that exists in DIS simulations while benefits from the scalability and interoperability of the HLA. The FOM was also approved as the SISO (Simulation Interoperability Standards Organization) Standard SISO-STD-001.1-1999 [28] in 1999.

In order to do research based on the RPR FOM, we select three SOMs that are related to the RPR FOM. These three SOMs are: the Tiger version 2.0 SOM, the Cobra Ball SOM and the FATS SOM. The compatibility between any two of them is checked, and the results are summarized below.

#### 5.1.2.1. Tiger Version 2.0 SOM vs Cobra Ball SOM

The match result for the Tiger version 2.0 SOM and the Cobra Ball SOM is mismatch. There exist some differences in definition of ComplexDataType RTIObjectIdStruct, as represented in Table 5-3. Moreover, there is no subscriber for the Publishable Objects EmbeddedSystem, IFF and IffTransponder.

| Element | Tiger version 2.0 SOM | Cobra Ball SOM |
|---|---|---|
| ID.DataType | Octet | String |
| ID.Cardinality | Dynamic | 1 |
| ID.Accuracy | N/A | Perfect |
| ID.Accuracy Condition | N/A | Always |

**Table 5-3 Differences in Definition of ComplexDataType RTIObjectIdStruct between Tiger version 2.0 SOM and Cobra Ball SOM**

#### 5.1.2.2. Cobra Ball SOM vs FATS SOM

The match result for the Cobra Ball SOM and the FATS SOM is mismatch. The differences in definition of ComplexDataType RTIObjectIdStruct are the same as the previous result. In other words, the definition is the same between the FATS SOM and the Tiger version 2.0 SOM. Besides, there exist the Publishable Objects including MilitaryPlatformEntity, MilitaryAirLandPlatform, EmbeddedSystem, IFF and IffTransponder which do not have a subscriber.

#### 5.1.2.3. Tiger Version 2.0 SOM vs FATS SOM

The match result for the Tiger version 2.0 SOM and the FATS SOM is mismatch. There are a number of differences in definitions of ComplexDataTypes between the two SOMs. These differences of definitions are summarized in Table 5-4.

| ComplexDataType | Element | Tiger version 2.0 SOM | FATS SOM |
|---|---|---|---|
| FixedDatumStruct | FixedDatumValue. Resolution | N/A | 1 |
| | FixedDatumID. DataType | DatumIDEnum | DatumIDEnum32 |
| RadioTypeStruct | EntityKind. DataType | EntityKindEnum | Octet |
| | Domain. DataType | EntityDomainEnum | Octet |
| | Category. DataType | EntityCategoryEnum | Octet |
| | Country. DataType | EntityCountryEnum | short |
| SphericalHarmonicAntennaStruct | ReferenceSystem.DataType | ReferenceSystemEnum | ReferenceSystemEnum8 |
| BeamAntennaStruct | ReferenceSystem.DataType | ReferenceSystemEnum | ReferenceSystemEnum8 |
| AttributeValueSetStruct | AttributeSetCount.DataType | Octet | Long |
| | AttributeSetCount.Resolution | N/A | 1 |
| VariableDatumStruct | DatumID.DataType | DatumIDEnum | DatumIDEnum32 |
| EntityTypeStruct | Subcategory.DataType | EntitySubcategoryEnum | Octet |
| | EntityKind.DataType | EntityKindEnum | Octet |
| | Specific.DataType | EntitySpecificEnum | Octet |
| | Domain.DataType | EntityDomainEnum | Octet |
| | Extra.DataType | EntityExtraEnum | Octet |
| | Category.DataType | EntityCategoryEnum | Octet |
| | Country.DataType | EntityCountryEnum | Short |
| RTIObjectIdArrayStruct | ID.DataType | Octet | RTIObjectIdStruct |
| ArticulatedParameterStruct | ArticulatedParameterType.DataType | ArticulatedTypeEnum | ArticulatedTypeEnum32 |
| | ParameterValue.DataType | ParameterValueType | ParameterValueStruct |
| MarkingStruct | CharacterSet.DataType | CharacterSetEnum | CharacterSetEnum8 |
| AntennaPatternStruct | AntennaPatternType | × | √ |

**Table 5-4 Differences in Definitions of ComplexDataTypes between Tiger version 2.0 SOM and FATS SOM**

Although some of the differences can be handled with the enhanced Matching Algorithm, the rest of them cannot be ignored. As a result, we cannot build a FOM based on the RPR FOM with the existing models in the library. More models and investigation are needed to finish this FOM case study.

## 5.2. Feasibility

This work attempted for an alternative way to develop the HLA federation automatically. It further facilitated the reusability and interoperability of the HLA federates/federations and saved time and efforts. The MCE produced real FOMs based on the HLA OMT. This system has several advantages such as:

- ***Robustness and Reliability:*** The composing SOMs were carefully examined and checked through the Matching Algorithm; the compatibility of the resulting FOM was ensured, at least in HLA OMT syntax;
- ***Friendliness:*** The user interface was designed in a user friendly layout, which facilitated the ease of operation;
- ***Platform Independence:*** The MCE was designed with minimal system requirements from the user, and all that the user needs is just a web browser that supports java applet;
- ***World Wide Accessibility:*** By sharing the models online, this work promoted the world wide accessibility of the simulation models. With more and more model builders/developers sharing their FOM/SOMs through the Internet, the building process for FOM will be easier and quicker.

The enhanced Matching Algorithm is proved to be able to provide more reusability of the models. The first case study is an good example of the advantage of the enhancement with AFF concepts.

### 5.2.1. DoD 1.3 vs IEEE 1516

Currently all the FOM/SOMs used in this work are based on the U.S. Department of Defense HLA OMT Specification Version 1.3 (DoD 1.3). However, in September 2000, the Institute for Electrical and Electronic Engineers (IEEE) approved a new standard for HLA, the IEEE 1516 standard [29] based on the DoD 1.3. To make the MCE more useful and adaptive to the new IEEE 1516 standard, we compare the two OMT standards here.

Most parts of the IEEE 1516 standard, such as the form, the functionality and the contents, are generally similar to the DoD 1.3. But, several changes to the Data Distribution Management (DDM) services were made [30]. These changes include:

- ***Removal of Routing Spaces:*** In IEEE 1516, there is only one routing space in which all the dimensions exist. This change facilitates the information publication/subscription and more importantly, it eliminates the risk of misunderstanding in Routing Space definition.
- ***Replacement of Extents and Regions:*** Because the concepts of extents and regions are a little bit confusing, IEEE 1516 also replaced the extents with regions and the regions with region sets.
- ***Addition of Default Regions:*** IEEE 1516 defined the default region as

having one range for all dimensions found in the FOM.

Besides, to implement those changes, part of the OMT is also modified. The Routing Space Table is replaced by the Dimension Table; each object attribute and the interaction are now associated with a set of dimensions instead of routing space. This set is the available dimensions that are used during information publication and subscription.

We can upgrade our matching algorithm from the current DoD 1.3 to IEEE 1516 by modifying the Routing Space Check. Since there is no routing space definition in the IEEE 1516 standard, we should check for the Dimension definitions instead of the Routing Space definitions. This could be done similarly as the Routing Space Check. During the Object/Interaction Match process, we should replace the matches of the routing spaces between the object attributes/interactions to that of the dimensions.

## 5.3. Limitations of the OMT

The interoperability and compatibility of the object models in this work were defined following the HLA OMT. Generally, a FOM built from the MCE could be a representation for a concrete federation. Although for most of the cases this scheme works properly, the limitation of the HLA OMT would still cause some problems.

### 5.3.1. Substantive Interoperability Problems

The substantive interoperability was defined in section 2.3 as the capability of federates to provide adequate, accurate and consistent simulated representations and

address the mission objectives. The current OMT specification does not provide sufficient information to test all of the criteria for substantive interoperability.

First of all, the representational anomaly would occur since the real world is more complicated than people think. There would be some events or states that are ignored by the model designer. Sometimes the order of the events would be different from what in the real world. All these can cause representational anomalies.

Secondly, the FOM/SOM does not provide details about the functional dependencies among models and functions. If one or more of the federates in a federation has stringent timing requirements such as real-time requirements, the federation might have timing incompatibility problems.

Furthermore, the manifold representations may not provide information that is either publishable or subscribable and thus would not be specified in the FOM/SOMs at all. Hiding these details reduces the complexity of federation construction, while it brings more complexity in detecting substantive interoperability problems in the federation.

### 5.3.2. Non-exchange Data Problems

The objective of the HLA OMT is to provide users with means and structure to represent the data that are exchanged during simulation execution. However, the structure is quite limited. For example, objects can only be structured as collections of attributes with no characterization of the properties of the functions that use or change those attributes. The properties of attributes are also limited. The OMT does

not intend to help users to define some properties for object/attribute that are not related to the data exchange.

Meanwhile, simulations are based on assumptions. Different federates may use different assumptions, which is definitely not related to data exchange. Thus, these assumptions are always ignored in the FOM/SOMs. The critical difference in assumptions may even lead to risky situation.

A good example would be the electricity outlet [31]. In different countries, the shape of the electric outlet is different. It might be the 2-way or the 3-way outlets as illustrated in Figure 5-1.

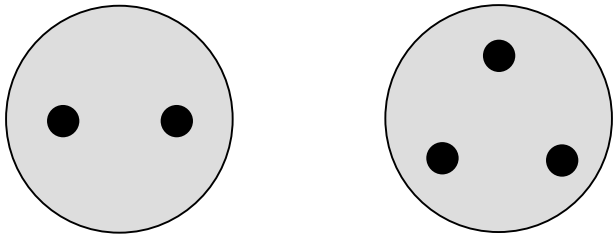

**Figure 5-1 The 2-Way and 3-Way Outlets**

Even 3-way outlets are not always the same; their shapes would vary from country to country. Figure 5-2 shows some different 3-way outlets.

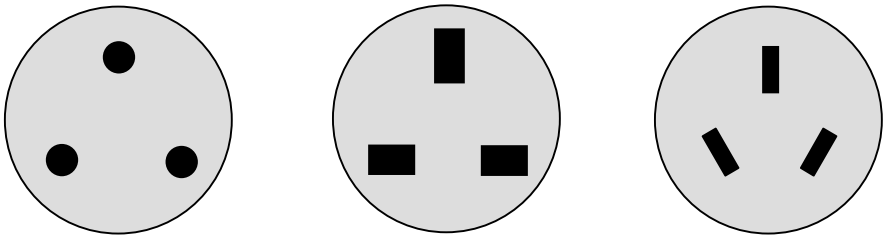

**Figure 5-2 Different 3-Way Outlets**

And even if the electric outlet is the same, there is still an assumption, the voltage supplied in the outlet. Different countries may operate on different voltages.

These information (shape of outlet and voltage) are often not documented in the FOM/SOM. Although model developer may try to record these information, e.g. define them as Enumerated DataTypes, the actual value of them can not be represented in the object/attribute. So, if people want to form a federation with some federates from different countries, the federation can not work in spite of a successful match result. Without noticing the critical differences of the assumptions among the federates, the implementation of the model we built can even cause damage in the real life.

## 5.4. Summary

This chapter examined the feasibility of the MCE with two actual case studies. A combat federation is developed based on the Matching Algorithm and the federation could be a practical implementation of CGF. It was also found that the enhanced Matching Algorithm can provide more compatibility for FOM/SOMs. Currently the OMT version IEEE 1516 is not supported in the MCE.

The OMT itself has some limitations in documenting and describing the models. Basically these limitations include the substantive interoperability problems and non-exchange data problems.

# Chapter 6 Extensible Elements

By recognizing the deficiency of the HLA OMT, the Modeling and Simulation industry has developed different schemes to strengthen the models' representation accuracy and validate them. Some of the schemes use additional documents together with the DIF to better document and shape the models, while others provide standard libraries to ensure the model's credibility. These schemes are discussed in this chapter.

However, these schemes cannot fit in the MCE because the MCE aims to check the model compatibility and build new models automatically. Thus, we propose an Extensible Element scheme to be embedded in the DIF. The elements are documented with priority level and detailed description to help users to recognize the functions and features of the elements in which it was embedded. This Extensible Element scheme is introduced in this chapter and we believe that it can improve the semantics of the current HLA DIF.

## 6.1. Widely Used Schemes

There are some widely used schemes which aim at improving the semantics of the current HLA OMT. Some of them use additional documents besides the HLA OMT DIF to better document and shaped the simulation models. Other schemes use standard ways to develop simulation models. Some of the widely used schemes are introduced below.

### 6.1.1. Simulation Conceptual Model

The Simulation Conceptual Model was initiated in 1999 by the Defense Modeling and Simulation Office (DMSO) to help achieve an agreement of the details of the model between the design and implementation sides [32]. It is widely used in simulation verification (determination that the simulation satisfies its specification) and simulation validation (determination that the simulation is adequate for its intended application).

It has been recommended that the simulation model documentation should include the nine items [31, 32] listed below:

- Conceptual Model Portion Identification;
- Principal Simulation Developer Point(s) of Contact (POCs) for the Conceptual Model (or part of it);
- Requirements and Purpose;
- Overview;
- General Assumptions;
- Identification of Possible States, Tasks, Actions, and Behaviors, Relationships and Interactions, Events, and parameters and Factors for Entities and Processes being described;
- Identification of Algorithms;
- Simulation Development Plans;

- Summary and Synopsis.

During federation construction, the Simulation Conceptual Model can provide the characteristics and compatibility requirements for choosing federates. Thus, defining a standardized conceptual model framework would greatly help to avoid substantive interoperability problems from the beginning. However, the Simulation Conceptual Model cannot be used in the MCE which tries to test the compatibility of models automatically. It is not practicable to analyze and compare the different definitions and descriptions of the Simulation Conceptual Model.

### 6.1.2. SEDRIS Model

The Synthetic Environment Data Representation and Interchange Specification (SEDRIS) project is sponsored by DMSO aimed at providing ways to represent environmental data and promote the unambiguous interchange of environmental data [33].

One of the main challenges of the SEDRIS model is to integrate all the different definitions of environmental data. This is achieved by providing an architecture that relates different representations to each other and attributes them in some categories. The SEDRIS Data Representation Model (DRM) and Data Coding Standard (DCS) are also defined to help unify the representations of environmental data.

The SEDRIS DRM provides not only a specific notation to depict the data (objects) with their attributes but also a graphic representation of the relationship between the data. It gives a means to ensure that the system’s data are defined with a complete

and unambiguous definition as well as their relationships with other data in the system. For example, Figure 6-1 illustrates the “is-a” relationship in the DRM.

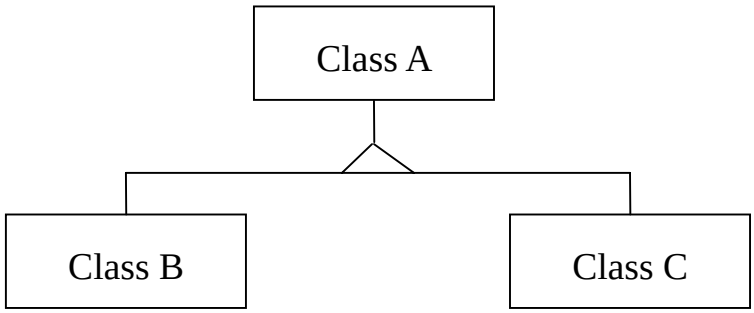


**Figure 6-1 Class B and Class C are both Child Classes of Class A**

The SEDRIS DCS provides a mechanism to specify the environmental data which a particular data model is intended to represent. There are three types of codes used in the DCS:

- ***Classification Codes:*** Address what the object is. For example, “AL015” represents a building, “BH140” represents for a river/stream.
- ***Attribute Codes:*** Represent the additional clarifying characters. Codes are generally assigned based on mnemonic abbreviations, for example, “BFC” for Building Function Category.
- ***State Codes:*** Represent the status of the object. For example, “DGEN” for Damage, General.

Working together, these three DCS components support the unambiguous description of the environmental data. More information on SEDRIS and its DRM and DCS can also be found at the SEDRIS website [33].

Although the SEDRIS model can help to define the relationship between simulation data, we cannot find a way to integrate it in the MCE. As previously claimed, all that need to be compared in the MCE is one single model description or document. The SEDRIS model is only useful when we build simulation models following the standard FEDEP processes which are time consuming.

### 6.1.3. OML & OMDDS

Another trend is to unify the definition of object models. The Object Model Library (OML) was introduced by the DMSO in October 1997 to set up tools for object model development in FEDEP [34]. It provides a central library of reusable HLA object models, as well as tools to identify HLA object models of interest. It also supports browse and search functions so that the users can find appropriate SOMs to form new federations.

The Object Model Data Dictionary System (OMDDS) was introduced in March 1998 as a resource to assist in the construction of FOMs and SOMs [35]. It specifies the components with details to construct FOM or SOM. These components include:

- ***Object classes;***
- ***Interaction classes;***
- ***Generic elements (attributes and parameters);***
- ***Complex data types;***
- ***Enumerated data types.***

By restrictively using the products and models built from the OML or the OMDDS, we can ensure the DMSO standard definitions of (components of) object models among the models. However, the capacity and the completeness of the OML or OMDDS might be a problem, since we cannot expect them to cover all the models we want. Some scheme still needs to be taken to strengthen the capability of the HLA OMT itself, but not the way we use and implement it.

## 6.2. Extensible Elements

The current efforts to help achieve a common understanding among the simulation models could not be used in our work. Thus, we propose a new Extensible Element scheme to extend the OMT DIF.

### 6.2.1. Definition

As the OMT DIF is inefficient in representing the models and their assumptions, an easy solution is to facilitate an Extensible Element scheme of the OMT DIF. Just like the concept of the Extensible Markup Language (XML), the Extensible Element scheme provides the users with maximum variability in documenting the models.

The Extensible Element scheme is defined as a complement of the OMT DIF. The elements can appear where the model builder thinks appropriate. It can address the characteristics of the model component or some assumptions that are made. Because the elements are also used in the MCE, they are documented in XML format too.

An example of the Extensible Elements is illustrated in Figure 6-2. The element

begins with a <Extensible_Element> tag while ends with a </Extensible_Element> tag. As one can see from the figure, there are one Shape element in the Outlet_Shape attribute and one Voltage element in the Electric Outlet class.

```
   <Class>
      <ID>1</ID>
      <Name>Electric Outlet</Name>
      <MOMClass>true</MOMClass>
      <PSCapabilities>PS</PSCapabilities>
      <Description>An example to illustrate the extensible
elements</Description>
      <Attribute>
         <Name>Outlet_Shape</Name>
         <Extensible_Element>
            <Name>Shape</Name>
            <Content>3-way</Content>
            <Priority>Medium</Priority>
            <Description>The shape of the outlet</Description>
         </Extensible_Element>
         ...
      </Attribute>
      <Extensible_Element>
         <Name>Voltage</Name>
         <Priority>High</Priority>
         <Content>220</Content>
         <Description>The voltage of the outlet that is operating
on</Description>
      </Extensible_Element>
   </Class>
```

**Figure 6-2 An Example of Extensible Elements**

The details of the Extensible Elements are introduced in the sub sections below:

### 6.2.2. Priority Level

In order to manage different Extensible Elements and distinguish the significance of them, we define the priority level of the elements. The Priority Level is documented between the <Priority> and </Priority> tags. There are three levels of priority for the elements:

- ***High:*** High level priority elements are used to represent those critical factors that cannot be ignored;
- ***Medium:*** Medium level priority elements are used to represent the important factors that can be easily fixed;
- ***Low:*** Low level priority elements are used to represent the unimportant factors that are suggested only. These factors are rather preferences than requirements.

If we go back to the electricity outlet example in section 5.2.2, the Extensible Element that are involved with the voltage information should be given high level priority because it is a critical factor in the simulation; the element for the shapes of the outlet should be give medium level priority, since the possible problem could be fixed easily with an adaptor. And such information as “my favorite computer brand is DELL” could be documented as a low level priority element.

### 6.2.3. Description

In embedding the Extensible Element in the DIF, some description of the element is also encouraged. The description information should describe the definition, meaning or possible values of the element, which could be helpful for the users to recognize and select during the FOM construction. The description is documented between the <Description> and </Description> tags.

### 6.2.4. Match

The Extensible Elements defined in the FOM/SOMs have to be covered in the Matching Algorithm. The algorithm should work as we defined in chapter 4 until it meets the Extensible Element. Different actions would be taken when the priority level of the elements are different. It does not matter in which of the four consecutive processes the elements could exist, but for the reason of clarity, we assume the Extensible Element we are matching exists in a publishable object. When the priority level of the element is:

- ***High:*** The subscriber object should have the same definition of this extensible element. The element name and value should exactly match the one in the subscriber object.
- ***Medium:*** The subscriber object should at least have the definition of this extensible element. The element name should be the same as the one in the subscriber object, but the values might be different.

- ***Low:*** No action needs to be taken.

The high level priority elements should be exactly matched during the Matching Algorithm, because those elements are critical factors that cannot be ignored. However, the medium priority elements should appear in both sides of the matching objects only since those factors represented by the elements could be easily fixed. As long as both sides are aware of the factors, it would be no problem. For those low level elements, we could ignore them since they are just some preference.

## 6.3. Summary

This chapter first reviewed several means used in the M&S to strengthen the semantics of the models. Unfortunately, the schemes that were explored cannot contribute to the MCE. Thus, an Extensible Element scheme was proposed and used in the MCE. Using the scheme, the FOM/SOM developer can address non-exchange data in the model.

There are several advantages of the Extensible Element scheme. Firstly, it is an efficient complement of the OMT defined elements. The user could address more about the characteristics of the model and record important assumptions that would be easily ignored by people other than the model builder. The description can help the user to better understand the element as well. Secondly, the Extensible Elements could appear in any place of the OMT DIF. The freeness of this scheme makes it possible to better shape every part of the components in the model. Thirdly, it is covered in the Matching Algorithm, too. Thus, the compatibility among different

models can be ensured.

As the XML can improve the variety and effects of the HTML (Hyper-Text Markup Language), we believe the extensible OMT DIF can provide more liveliness and accuracy than the original one. Meanwhile, it is a useful complement for the OMT DIF in this work. It helps to achieve a common definition and description of the simulation models. By suggesting a standard way to use and define the Extensible Element, we could expect to gain more from the scheme.

# Chapter 7 Conclusion

This chapter concludes the thesis work. Some aspects for future work are also suggested in this chapter.

## 7.1. Summary

This thesis work argued for an alternative method to build HLA FOMs with existing HLA SOMs automatically, which further facilitated the reusability of HLA federates and interoperability among them.

The Model Construction Environment (MCE) was built and introduced in this thesis. It provided the user with an interface to select some existing models to construct the new models. A DIF-XML converter was built to transform the HLA OMT DIF format models to XML format files. For the reason of easy operation, the models can be viewed in both tree-node style or in XML file format. Besides, the models are editable in the MCE. To help the selection, a key-word search function was also provided. If the result is a mismatch, users can either replace the incompatible model with a new model or modify the model. However, if the result is not a mismatch, the MCE offered a file server running on the server side to save the new models for future reuse. By publishing it on the web, the MCE could be a useful tool for model development world wide.

The Matching Algorithm was defined to check the compatibility among the selected SOMs based on the OMT DIF. It consisted of four consecutive processes, namely,

the DataType Check, the Routing Space Check, the Object Match and the Interaction Match. These four processes of the algorithm were illustrated in this thesis. The algorithm could ensure the compatibility and interoperability of the models in the OMT DIF syntax. However, sometimes a SOM may be reused in different FOMs. To strengthen the agility of the SOM, the Matching Algorithm was enhanced with some concepts from the Agile FOM Framework. The FOM development process can make use of data analyzed in the Matching Algorithm. The process was also introduced in this thesis.

Two case studies were investigated in this study. The combat Federation case study selected two military models in the library and built a FOM with the enhanced Matching Algorithm. This FOM could be a practical implementation of CGF. The RPR FOM case study tried to build a RPR related FOM, but it failed to find appropriate composing sub-models. With more models available in the model library, it is believed that the FOM case study can be finished.

For most of the cases the MCE with the Matching Algorithm worked fine. However, because of the natural limitation of the HLA OMT, some substantive interoperability problems and non-exchange data problems might occur. The additional documents such as the Simulation Conceptual Model and the SEDRIS model could not help in this situation since we were trying to build a new simulation model automatically with single document/description of the model. The OML and OMDDS scheme which unify the definition of object models were not appropriate either, since the capacity and completeness of them is limited.

To solve the problem, the concept of Extensible Elements for the OMT DIF was introduced in this thesis. The element was embedded in the current DIF models. It was documented with priority level and detailed description to help users to recognize better the functions and features of the elements in which it is embedded. The Extensible Element was covered in the Matching Algorithm too. This scheme was helpful to strengthen the identity and accuracy of the object models based on OMT DIF.

This thesis work can be contributive to more cost-efficient methodologies for automatic development of simulation models and codes. It also brought forward some suggestions to the augmentation of the current HLA OMT standard.

## 7.2. Future Work

This thesis work proposed the SOM compatibility check and FOM development process through the Matching Algorithm and the MCE. It also illustrated the feasibility of the scheme and provided an Extensible Element to strengthen the semantics of the models. However, there is still some interesting future work that worth doing research on based on this work.

Firstly, more case studies can be investigated to test the scheme. Not many FOMs are generated successfully from the MCE, which means the MCE by now is not as contributing as it should be. And, there are only 10 FOM/SOMs available in the model library currently. The number is rather small to find out other possible problems. With more models are available in the library, the feasibility study of the

scheme can be enhanced.

Secondly, it would also be interesting to support models based on the IEEE 1516 standard. Although many FOM/SOMs are stored in both DoD 1.3 and IEEE 1516 format, some of them only support one format. Thus, enhancing the MCE to support both of the formats will be a good future work.

Thirdly, the Extensible Element scheme is still widely open for research. It could be improved in a more systematic way. It is also possible to suggest the scheme to the M&S industry and even establish a standard.

Fourthly, there exist some open source products that can convert XML files to Java classes, e.g. the Digester component under Apache Jakarta project [36]. Some research can be done to investigate how these ideas can be used and whether the ideas can improve performance in the MCE.

# References


[1] Defense Modeling and Simulation Office (DMSO). *High Level Architecture, Federation Development and Execution Process (FEDEP) Model, Version 1.5*, December 8, 1999.

[2] Rassul A. and Farshad M. "A net-based modeling and simulation platform (NetMas)". http://www.imit.kth.se/forskningsprojekt-detalj.html?projektid=39.

[3] U.S. Department of Defense. *High Level Architecture, Object Model Template Specification, Version 1.3*, 5 February 1998 (27 July 1998 Document Release).

[4] Paul A. Fishwick. *Simulation Model Design and Execution: Building Digital Worlds*, Prentice Hall, Englewood Cliffs NJ, 1995.

[5] Fujimoto and Richard M. *Parallel and Distribution Simulation Systems*, John Wiley, New York, 1999.

[6] M. Rodriguez and R. Sanz. "HOMME: A Modeling Environment for Multiple Applications", International Journal of Modeling and Simulation, Vol. 21, No. 3, pp. 171-177, 2001.

[7] The NanoComp Project. http://nanacom.et.tudelft.nl/.

[8] A. Levytsky and EJH. Kerckhoffs. "Integration of Simulation Tools and Models in a Collaborative Environment", Proceedings of the 2001 European Simulation Interoperability Workshop, London, UK, July 2001.

[9] The Dublin Core Metadata Initiative (DCMI). http://dublincore.org/.

[10] S. K. Das and A. Reyes, “An Approach to Integrating HLA Federations and Genetic Algorithms to Support Automatic Design Evaluation for Multi-agent Systems”, Journal of Simulation Practice and Theory, Vol. 9, No. 3-5, pp. 167-192, Apr 2002.

[11] Dahmann, J.S., Kuhl, F. and Weatherly, R. “Standards for Simulation: As Simple as Possible But Not Simpler - The High Level Architecture for Simulation”, SIMULATION, 71:6, pp 378-387, December 1998.

[12] Under Secretary of Defense for Acquisition and Technology. *Department of Defense Modeling and Simulation Master Plan, DoD 5000.59-P*, October 1995.

[13] Defense Modeling and Simulation Office (DMSO). *High Level Architecture, Run-Time Infrastructure 1.3 Next Generation Programmer’s Guide, Version 3.2,* 7 September 2000.

[14] Naur, P. et al. “Report on the Algorithmic Language ALGOL 60”, Communications of the ACM, Vol.6, No. 1, pp 1-17, January 1963.

[15] D.C. Gross et al. “Report from the Fidelity Implementation Study Group”, Proceedings of the 1999 Spring Simulation Interoperability Workshop, Orlando, FL, 14-19 March 1999.

[16] J. Dahmann et al. “HLA and Beyond: Interoperability Challenges”, Proceedings of the 1999 Fall Simulation Interoperability Workshop, Orlando, FL, 12-17 September 1999.

[17] Harmon, S.Y. & Youngblood, S.M. “Leveraging Fidelity to Achieve Substantive Interoperability”, Proceedings of the 2001 Spring Simulation Interoperability Workshop, Orlando, FL, March 2001.

[18] Institute of Electrical and Electronics Engineers (IEEE). *Standard for Information Technology - Protocols for Distributed Interactive Simulation Applications*, IEEE Std 1278-1993, March 1993.

[19] The World Wide Web Consortium (W3C). Extensible Markup Language (XML), http://www.w3.org/XML/.

[20] Lacy Lee, George Stone III and Theodore D. Dugone. “XML Data Interchange Format Standards for HLA-Related Data Interoperability”, Proceedings of the Southeastern Simulation Conference, 1999.

[21] Lacy Lee and Theodore D. Dugone. “Developing International Standards for Interchanging Simulation Data”, Proceedings of the 2001 European Simulation Interoperability Workshop, March 2001.

[22] Haddix Furman, Scrudder Roy and Hopkins Mike. “Unit Order of Battle Toolset”, Proceedings of the 1999 Fall Simulation Interoperability Workshop, Orlando, FL, September 1999.

[23] D. Macannuco, B.Dufault and L. Ingraham. “A FOM Agile Framework”, Proceedings of the 1998 Fall Simulation Interoperability Workshop, Orlando, FL, September 1998.

[24] D. Wilbert, D. Macannuco and W. Civinskas. “A Tool for Configuration FOM

Agility", Proceedings of the 1999 Fall Simulation Interoperability Workshop, Orlando, FL, September 1999.

[25] Ernest H. Page and Roger Smith. "Introduction to military training simulation: a guide for discrete event simulationists", Proceedings of the 30th conference on Winter Simulation, Washington, D.C., United States, December 1998.

[26] G. Shanks. "The PnP FOM A Reference Federation Object Model to Promote Simulation Interoperability between real-time, platform level federates", Proceedings of the 1997 Spring Simulation Interoperability Workshop, Orlando, FL, March 1997.

[27] Joseph J. Berkeley and Pete Muller. "Enhancing The RPR FOM to Include STOW 97 Capabilities", Proceedings of the 1998 Fall Simulation Interoperability Workshop, Florida, U.S.A, September 1998.

[28] Real-time Platform Reference (RPR) FOM Standard Development Group (SDG). SISO-STD-001.1-1999: Real-time Platform Reference Federation Object Model, http://siso.sc.ist.ucf.edu/doclib/doclib.cfm?SISO_CID_59.

[29] Institute of Electrical and Electronics Engineers (IEEE). *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Object Model Template (OMT) Specification*, IEEE Std 1516.2-2000, 9 March 2001.

[30] Morse, K. L. and Petty, M. D. "Data Distribution Management Migration from DoD 1.3 to IEEE 1516," Proceedings of the Fifth IEEE International Workshop on Distributed Simulation and Real-Time Applications, Cincinnati

OH, August 2001.

[31] Dale K. Pace. “Simulation Conceptual Model Role in Determining Compatibility of Candidate Simulations for a HLA Federation”, Proceedings of the 2001 Spring Simulation Interoperability Workshop, Florida, U.S.A, March 2001.

[32] Dale K. Pace. “Development and Documentation of a Simulation Conceptual Model,” Proceedings of the 1999 Fall Simulation Interoperability Workshop, March 1999.

[33] The Synthetic Environment Data Representation and Interchange Specification (SEDRIS™). http://www.sedris.org/.

[34] Lutz R., R. Scrudder and J. Graffagnini. “High Level Architecture Object Model Development and Supporting Tools”, SIMULATION, 71:6, pp 401-409, December 1998.

[35] Hammond, J., M. Dey, R. Scrudder and F. Merkin. “Populating the HLA Object Model Data Dictionary - A Bottom-Up Approach”, Proceedings of the 1998 Spring Simulation Interoperability Workshop, March 1998.

[36] The Digester Component. Apache Jakarta Project, http://jakarta.apache.org/commons/digester.html.

[37] Hamilton, J.A., David A.N. and Udo W.P. *Distributed Simulation*, CRC Press, New York, 1997.