

Automatic SOM Compatibility Check & FOM Development

Gary Tan, Yu Hu
School of Computing
National University of Singapore
SINGAPORE, 117543
{gtan, huyu}@comp.nus.edu.sg}

Farshad Moradi
Swedish Defense Research Agency
Stockholm, SWEDEN
SE-172 90
farshad@foi.se

Abstract

The High Level Architecture (HLA) is widely used in defense applications as a common framework for modeling and simulation. The Federation Development and Execution Process (FEDEP) is a generalized process to build HLA federations from scratch. However, simulation model design, implementation, testing and execution defined in the FEDEP are time consuming and expensive. The Model Construction Environment (MCE) is introduced in this paper as a web-based system to enhance the efficiency of the federation development by reusing the existing Simulation Object Models (SOM) to build new Federation Object Models (FOM). To ensure the compatibility of the SOMs in a FOM, a Matching Algorithm is defined and tested. An Extensible Elements scheme is designed to help shape the characteristics of the FOM/SOMs. This study can contribute to more cost-efficient methodologies for the development and execution of simulation models.

1. Introduction

1.1. Overview

Computer simulation is the process of designing and modeling an actual system and executing the model based on a digital computer. With the emergence of the low-cost and high-power computers, computer simulation is developing fast in recent years. For instance, distributed simulation, which is built and executed on distributed systems, could help in situations when the simulation has heavy computation or vary in problem size.

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 is widely used in defense M&S for it facilitates the reusability of the simulations and the interoperability among them.

To support the general goals of the HLA, the HLA Object Model Template (OMT) [1] 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) [2] provides a high-level framework and a comprehensive systems engineering methodology. It divides the federation construction into six basic steps, from federation objective definition, model development to federation execution and results collection. It has also been regarded 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 widespread use of the Internet, more and more resources can be shared online. So, is there an efficient way to reuse existing simulation models and develop new models?

1.2. Objective

In 2001, the project "A net based modeling and simulation platform (NetMas)" [3] was initiated in Sweden, aimed 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 Environment (MCE), in which

a model builder may use 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 a matching algorithm. Also the comparison process has to be done automatically.

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-model used in this platform can be either Simulation Object Model (SOM), the description of a federate according to the OMT, or Federation Object Model (FOM), the description of HLA federations. The model created under this scheme is FOM and could be saved for future reuse.

This paper describes the data flow of the current MCE and the Matching Algorithm for SOM compatibility check. In addition, it examines whether the HLA OMT is sufficient for ensuring the compatibility, interoperability and communicability among federates in the federation. The Extensible Element with priority level scheme is introduced in this paper to strengthen the OMT.

This study is a first attempt to investigate the possibility to build HLA federation automatically. It can be useful to further facilitate the reusability and interoperability of the HLA federate/federations.

The rest of this paper is organized as follow: Section 2 introduces the MCE system and how it works; Section 3 explains the Matching Algorithm; Section 4 introduces an actual example, discusses the feasibility of the scheme and finds out some limitations of the current MCE; Section 5 introduces the extensible elements to extend the HLA OMT; Section 6 concludes this paper and gives some future work aspects.

2. Model Construction Environment

One of the main parts of the NetMas project [3] is the Model Construction Environment (MCE), designed to build new simulation models with existing ones. All the selected SOM/FOMs should be carefully examined through a compatibility checker before a new FOM is created. The compatibility checker is defined as the Matching Algorithm in this paper and will be discussed in detail in Section 3. The MCE also provides the users with a means to save the newly created model for future reuse.

2.1. Infrastructure

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

DIF is built upon a common meta-model that represents the information needed to represent and manage object models.

The Extensible Markup Language (XML) [4] is the universal format for structured documents and data on the Web. It has been recommended to IEEE as the standard for HLA DIF descriptions. Thus, the FOM/SOMs in this study are stored in XML format based on the OMT DIF. It is also worth noting that the OMT used in the MCE currently is the U.S. Department of Defense HLA OMT Specification Version 1.3 (DoD 1.3).

2.2. Framework

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 1 illustrates the relationship and the data flow chart between the client and the server.

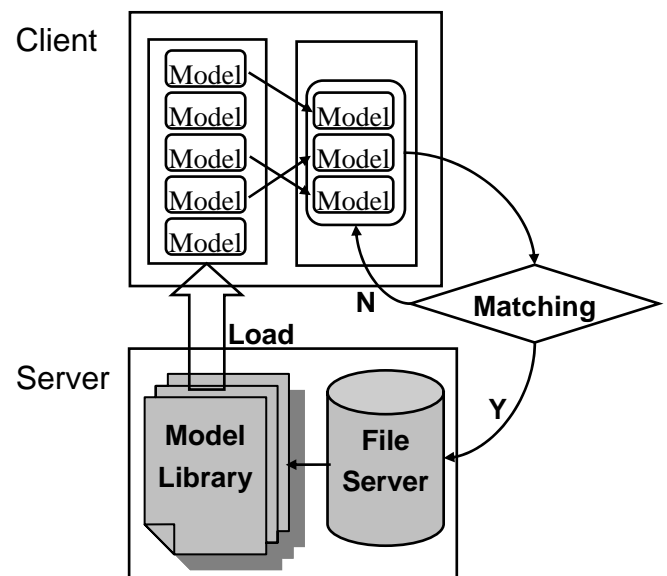


Figure 1. The framework

As one can see from Figure 1, the client interface first loads all the models from the model library and lists them before the user. The graphical interface also enables the user to view the existing FOM/SOMs in either tree-view mode or XML source file mode. Then the user decides which models he needs and puts them together in the model pool. To help the user find appropriate models, a keyword search function is provided. The selected models will be put through a Matching Algorithm to check for compatibility. If the matching result is false, the user is informed the details in which the models are incompatible in the status bar. To continue, the user has to go back to

the model pool, replace the models that are not compatible with some new models or modify the selected model based on the incompatible information. However, if those models are compatible under the Matching Algorithm, a new model will be built up. And lastly, to make the new model reusable in the future, a save request can be sent from the interface to the server application. The interface during execution is illustrated in Figure 2.

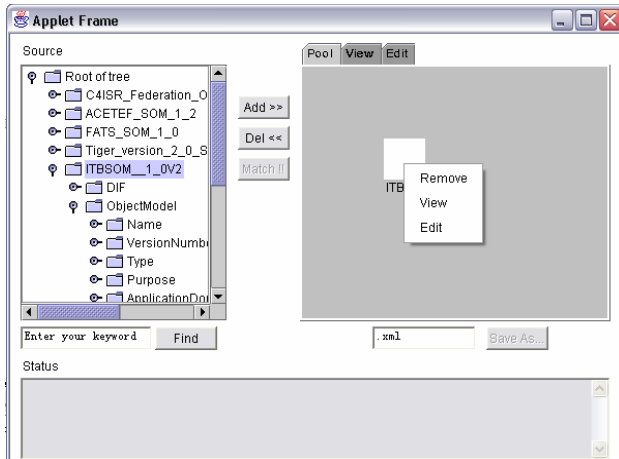


Figure 2. The user interface

The file server listens to the save request from the interface and performs the save action. It is built in Java. Both the file name and the content of the new model are transmitted from the user interface through socket communication. The new model can be a reinforcement of the existing model library.

3. The Matching Algorithm

The Matching Algorithm is used in the MCE to check the compatibility among the models and to present the result of the check. It is made up of four consecutive processes: DataType Check, Routing Space Check, Object Match and Interaction Match. Failure in any of the four processes will lead to a mismatch and the data/elements that led to the mismatch will be reported.

3.1. DataType Check

The DataType defined in HLA OMT is used to reference data types of the OMT elements [1]. It may be chosen from thirteen permissible base data types (e.g. "char", "integer" and "char", etc.), or it may be a user defined DataType.

User defined DataTypes, including the Enumerated DataTypes and the Complex DataTypes, are supplementary formats used to better document the OMT

contents. 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 a 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.

One of the first steps for the matching algorithm is to ensure the consistency in the DataType definition among all federates in the federation. Both the Enumerated DataTypes and the Complex DataTypes are checked. The rule for the checking is that whenever supplementary data types with the same name occurred in the federation, check whether the enumerators and representations of the datatype (for Enumerated DataTypes) or components (for Complex DataTypes) are accordingly the same.

3.2. Routing Space Check

The routing space is defined in the Data Distribution Management (DDM) of the HLA RTI Programmer's Guide [5]. A routing space is a multidimensional coordinate system for federates to express their willing to publish or subscribe data.

During the development of an HLA federation, it is critical that all federation members achieve an agreement of DDM routing spaces and their semantics, and stick to a common set of routing space specifications. These agreements can help federates filter object attribute updates and interaction that they are not interested.

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 DataType 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.

3.3. Object Match

An HLA object class is defined as a collection of objects with certain characteristics or attributes [1]. It is a key component in the HLA OMT. There are several properties in the 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 and their meanings 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.

Each HLA object is also characterized by a fixed set of attribute types [1]. The Attributes are defined as named portions of their object's state whose value can change over time. An HLA object model shall support representation of the characteristics for attributes, such as: Name, DataType, Cardinality, Units, Resolution, Accuracy, Routing Space, etc.

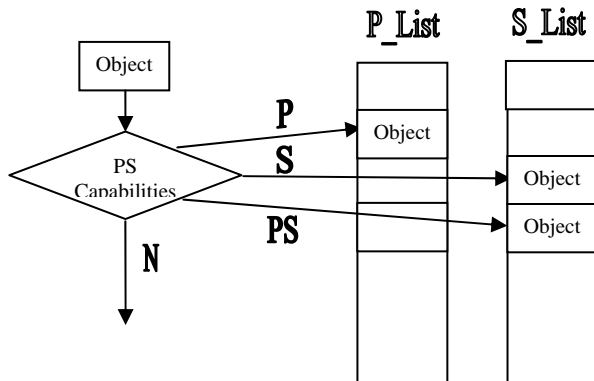


Figure 3. P_List and S_List construction

Whenever an object class is publishable in a federation, there shall be a same class which is subscribable otherwise it is useless to publish the object [1]. As a solution, we create 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 3, objects of which PSCapabilities is <P> are added into the P_List; the objects of <S> are added into the S_List; the objects of <PS> are only added into the S_List because even before matching we can tell that the object itself is a subscriber; objects of <N> are discarded.

After generating the two lists, the next step of the Object Match is to find a subscriber for each object in the P_List from the S_List. The subscriber should also have the same attributes characteristics.

3.4. Interaction Match

In the OMT, an interaction is defined as an explicit action taken by a federate that may have some effect or impact on another federate [1]. Similar to the PSCapabilities designation provided in the object class structure, the interaction class structure also provide a designation of federate/federation capabilities with respect to give classes of information. It is called ISRType and the possible values and the meanings are listed below:

- Initiates (<I>): Indicates a federate is currently able to initiate and send interactions of the given type;
- Senses (<S>): Indicates that a federate is currently able to subscribe to the interaction and utilize the interaction information;
- Reacts (<R>): Indicates that a federate is currently able to subscribe and properly react 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.

Most interaction classes will also be documented with a list of interaction parameters. An HLA object model shall support representation of the following characteristics for each parameter: Name, DataType, Cardinality, Resolution, Accuracy and Accuracy Condition.

As defined in the OMT, at least one federate should sense or react to every interaction class that is initiated in a federation [1]. This is the main concern of the Interaction Match Procedure. As a solution, we use a similar scheme as the Object Match to create the Initiates list (I_List) and Senses/Reacts list (SR_List). Every interaction in the federation is checked by the ISRType: the interactions of <I> are put into I_List; the interactions of <S> or <R> are put into SR_List; the interactions of <IS> or <IR> are put into SR_List only; the interactions of <N> is discarded.

The result of the Interaction Match will be successful if 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. The comparison between two interactions is also similar to the one between objects and we will not discuss in detail here.

4. Results and Discussion

The MCE user interface and server application was built up and tested with several examples. There were altogether 10 FOM/SOMs in our model library. Among the models, 8 were SOMs and 2 were FOMs. All the models in the library were available for federation construction. We chose some of them and developed some new FOMs. As an example, a combat federation FOM was built and illustrated below.

4.1. Combat Federation Example

One of the main application areas of M&S is military simulation. In military institutions, the computer-driven combat simulation has its use in officer training, mission rehearsal and tactics exploration. The need for military simulations such as combat simulation is continually increasing [6]. In this example 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 [7] 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.

The Extended Air Defense Simulation (EADSIM) is sponsored by the 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 gave a mismatch and the elements that led to the mismatch were presented in the status bar. It was found that both models have the definition of a ComplexDataType under the name of RelativePositionStruct, while there were some minor differences between the two definitions of ComplexComponents. Table 1 presents the differences between the two definitions.

However, the two definitions of the Data Type should be deemed as the same as one can see from Table 1. Thus, after converting the definition in EADSIM to that in CGF FOM, a match was reached. This newly created Combat Federation can be a practical implementation of CGF.

Table 1. Differences in definition of Complex DataType RelativePositionStruct between CGF FOM and EADSIM

Element	CGF FOM	EADSIM
FieldName	BodyX, BodyY and BodyZ	BodyXDistance, BodyYDistance and BodyZDistance
Units	metres	Meters

4.2. Feasibility

This study attempted for an alternative way to develop an HLA FOM automatically. It further facilitated the reusability and interoperability of the HLA SOM/FOMs to save time and effort. The MCE produced meaningful FOMs based on the HLA OMT. 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.

We also noticed that there exist other tools for building HLA models, for example, the Aegis OMDT Pro. The model libraries of these commercial products are usually kept confidential. However, the MCE introduced here shares simulation models online, which promotes worldwide accessibility of the models. With more and more model builders/developers sharing their FOM/SOMs through the Internet, the FOM building process will be easier and quicker in the MCE.

Currently all the FOM/SOMs used in this work were 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 [8] based on the DoD 1.3. One major change in the IEEE 1516 version is the removal of routing spaces: there is only one routing space in which all the dimensions exist. We can upgrade our matching algorithm from the current DoD 1.3 to IEEE 1516 by modifying the Routing Space Check. We should check for dimension definitions instead of 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 routing spaces between the object attributes/interactions to that of dimensions.

The matching method described in this paper uses string matching. However, federate developers may use different ways to implement model entities, which in reality can be the same things. To solve the problem, the Agile FOM Framework (AFF) suggests federate developers to build FOM Agile Federates that can be reused in different federations without code modifications [9]. The main idea is to build some converters for

mapping data in the SOM to different FOMs. By developing some AFF converters, the result of the Matching Algorithm may be more accurate and more SOMs in the model library can be reused. This will be considered and implemented in the future.

4.3. Limitations of the OMT

The interoperability and compatibility of the object models in this study were defined following the HLA OMT. Generally, a FOM resulting 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 bring out some credibility problems as defined in [10].

There are also some representation problems of the current HLA OMT scheme. First of all, the HLA OMT does not provide enough information for the models. It is a unified standard rather than a detailed document. The 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. Some complicated objects or attributes cannot be shaped with and identified from the information provided in FOM/SOMs. Secondly, simulations are based on assumptions. Different federates may use different assumptions, which are sometimes not indicated in FOM/SOMs. The critical difference in assumptions may even lead to risky situation.

A good example would be the electricity outlet [11]. In different countries, the shape of the electric outlet is different. If people want to form a federation with some federates from different countries, such information (shape of the electric outlet) cannot be described in the HLA OMT model. The federation may not work even if the match result was successful. And even if the electric outlet is the same, different countries may operate on different voltages. Without noticing the critical differences of the assumptions among the federates, the implementation of the models built can cause damage in the real life.

4.4. Additional Documents

By recognizing the deficiency of the HLA OMT, the Modeling and Simulation industry used different schemes to strengthen the models' representation accuracy and validate them. One trend is to use some additional documents to help in describing simulation models.

The Simulation Conceptual Model (SCM) was initiated in 1999 by Defense Modeling and Simulation Office (DMSO) to help achieve an agreement of the

details of the model between the design and implementation sides [12]. It is widely used in simulation verification and validation. During federation construction, the SCM can provide the characteristics and compatibility requirements for choosing federates [11, 12]. Thus, defining a standardized conceptual model framework would greatly help to avoid substantive interoperability problems from the beginning.

The Synthetic Environment Data Representation and Interchange Specification (SEDRIS) project, sponsored by DMSO, is aimed at providing ways to represent environmental data and promote the unambiguous interchange of environmental data [13]. The SEDRIS Data Representation Model (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. The SEDRIS Data Coding Standard (DCS) provides a mechanism to specify the environmental data which a particular data model is intended to represent. Among these codes, the Classification Codes address what the object is, the Attribute Codes represents the additional characters and the State Codes represents the status of the object. Working together, these three DCS components support the unambiguous description of the environmental data.

Although all these documents and schemes offer help in identifying the objects in the models, they do not fit in with the situation here. This is because in this study, we are trying to build simulation models with the existing ones automatically. All that we compare here is one single model description or document. The additional document is only useful when we build simulation models following the standard FEDEP processes that are time consuming.

4.5. OML & OMDDS

Another trend is to unify the definitions of object models. The Object Model Library (OML) and the Object Model Data Dictionary System (OMDDS) were introduced by DMSO respectively in October 1997 and March 1998 [14] to set up tools for object model development in FEDEP. The OML aimed at providing appropriate SOMs to form new federations. Moreover, the OMDDS provides five types of components to build object model: object classes, interaction classes, Complex DataTypes, Enumerated DataTypes and generic elements.

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 is still needed to strengthen the capability of the HLA OMT itself, but not the way we use and implement it.

5. 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 Elements scheme to extend the OMT DIF.

5.1. Definition

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

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

In order to manage different Extensible Elements and differentiate between them, we define the priority level of the elements. 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 (e.g. the voltage information in the electricity outlet example).
- Medium: Medium level priority elements are used to represent the important factors that can be easily fixed (e.g. the shapes of the outlet in the electricity outlet).
- Low: Low level priority elements are used to represent the unimportant factors that are suggested only. These factors are rather preferences than requirements (e.g. my favorite PC brand is DELL).

In embedding the Extensible Elements in the DIF, some description of the elements are also encouraged. The description information can 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.

An example of the Extensible Elements is illustrated in Figure 4. The element begins with a `<Extensible_Element>` tag and ends with a `</Extensible_Element>` tag. As Figure 4 shows, there is 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>
  <DataType>any</DataType>
  <Cardinality>1</Cardinality>
  <UpdateType>Conditional</UpdateType>
  <UpdateCondition>N/A</UpdateCondition>
  <TransferAccept>N</TransferAccept>
  <UpdateReflect>UR</UpdateReflect>
  <Description>N/A</Description>
</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 4. An example for Extensible Elements

5.2. Match

The Extensible Elements embedded in the FOM/SOMs have to be covered in the Matching Algorithm. The algorithm should remain the same as we defined in Section 3 until it encounters Extensible Elements. 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 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 since the element represents critical factor that cannot be ignored. The element name and value should also 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 can be different. As long as both sides are aware of the factor which can be easily fixed, it would not be a problem.
- Low: No action needs to be done since the element is just a preference.

5.3. Summary

There are several advantages of the Extensible Elements scheme. Firstly, it is an appropriate complement of the OMT defined elements. The user can address more about the characteristics of the model and record important assumptions that would be easily ignored by people who do not know them. Secondly, the Extensible Elements could appear anywhere in the OMT DIF. The freeness of this scheme makes it possible to better shape every part of the components in the model.

Just as XML can improve the variety and effects of HTML, we believe the extensible OMT DIF can provide more liveliness and accuracy than the original one. Meanwhile, it helps to achieve a common definition and description of the simulation models. By suggesting a standard way to use and define the Extensible Elements, we could expect more gains from the scheme.

6. Conclusion

This paper argued for an alternative method to automatically build HLA FOMs with existing HLA SOMs, which further facilitate the reusability of HLA federates and interoperability among them. The Model Construction Environment (MCE) could produce meaningful models. By publishing it on the web, the MCE could be a useful tool for model construction worldwide. The matching algorithm was defined to check the compatibility among the selected SOMs based on the OMT DIF. It ensured the interoperability of the models in the OMT DIF syntax. For most of the cases the MCE with the Matching Algorithm works fine. However, because of the natural limitation of the HLA OMT, some credibility problems as well as representation limitation might occur. The Extensible Elements proposed in this paper served to strengthen the identity and accuracy of the object models based on OMT DIF.

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

OMT standard. Some aspects for future work include: improving the Extensible Element scheme in a more systematic way, investigating other schemes to strengthen the meaning and accuracy of the models, and taking model semantics into consideration during the matching process.

References

- [1] U.S. Department of Defense. *High Level Architecture, Object Model Template Specification*, Version 1.3, 5 February 1998 (27 July 1998 Document Release).
- [2] Defense Modeling and Simulation Office (DMSO). *High Level Architecture, Federation Development and Execution Process (FEDEP) Model*, Version 1.5, December 8, 1999.
- [3] Rassul Ayani. and M. Farshad. A net-based modeling and simulation platform (NetMas), <http://www.imit.kth.se/forskningsprojekt-detalj.html?projektid=39>.
- [4] The World Wide Web Consortium (W3C). Extensible Markup Language (XML), <http://www.w3.org/XML/>.
- [5] 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.
- [6] 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.
- [7] Under Secretary of Defense for Acquisition and Technology. *Department of Defense Modeling and Simulation Master Plan*, DoD 5000.59-P, October 1995.
- [8] 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.
- [9] D Macannuco, B. Dufault, L. Ingraham. "A FOM Agile Framework", Proceedings of 1998 Fall Simulation Interoperability Workshop, September 1998.
- [10] Simone Y. and O. Linda. "Federation Credibility Challenge", Proceedings of the 12th International Training and Education Conference (ITEC), Lille, France, April 2001.
- [11] Dale K. Pace. "Simulation Conceptual Model Role in Determining Compatibility of Candidate Simulations for a HLA Federation", Proceedings of 2001 Spring Simulation Interoperability Workshop, Florida, U.S.A., March 2001.
- [12] Dale K. Pace. "Development and Documentation of a Simulation Conceptual Model," Proceedings of the 1999 Fall Simulation Interoperability Workshop, March 1999.
- [13] The Synthetic Environment Data Representation and Interchange Specification (SEDRIS™), <http://www.sedris.org>.
- [14] Lutz R., R. Scudder and J. Graffagnini, "High Level Architecture Object Model Development And Supporting Tools", SIMULATION, Volume 71:6, December 1998, 401~409.