A METAMODEL FOR THE HLA OBJECT MODEL

Deniz Çetinkaya
Halit Oğuztüzün
Department of Computer Engineering
Middle East Technical University
Inonu Bulvari, 06531, Ankara, Turkey
E-mail: e131263@ceng.metu.edu.tr, oguztuzn@ceng.metu.edu.tr

KEYWORDS
High Level Architecture, Object Model Template, metamodeling, model integrated computing, Generic Modeling Environment

ABSTRACT

The High Level Architecture (HLA), IEEE Std. 1516-2000, provides a general framework for distributed simulation applications, called federations. An HLA object model, be it a simulation object model (SOM), a federation object model (FOM) or the management object model (MOM), describes the data that can be transferred during some federation execution. This paper introduces a metamodel for the HLA Object Model, fully accounting for IEEE Std. 1516.2. The metamodel ("paradigm") is constructed with GME (Generic Modeling Environment), a meta-programmable tool for domain-specific modeling, developed by the Institute for Software Integrated Systems at Vanderbilt University. The paper also describes an application based on the paradigm. This work aims at bringing model-integrated computing to bear on HLA-based distributed simulation.

INTRODUCTION

The High Level Architecture (HLA) provides a standardized, general framework within which distributed simulation developers can structure their applications. HLA puts special emphasis on interoperability and reusability of individual simulations. It has three components: HLA Rules, Object Model Template (OMT) and Federate Interface Specification. OMT, which constitutes the subject of this paper, describes the forms of data involved in any HLA object model, which can be a simulation object model (SOM), a federation object model (FOM) or the management object model (MOM).

Officially designated "IEEE Std. 1516-2000 Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)", the standard was prepared by the HLA Working Group, sponsored by the Simulation Interoperability Standards Committee (SISC) of the IEEE Computer Society [SISC 2000]. The standard consists of four related standards:

- *IEEE Std. 1516-2000: IEEE Standard for M&S HLA Framework and Rules:* Provides an overview of the HLA and defines a set of principles that governs HLA-compliant federations and federates.
- *IEEE Std. 1516.1-2000: IEEE Standard for M&S HLA Federate Interface Specification:* Defines the standard services and interfaces to be provided by the Run Time Infrastructure (RTI) and to be used by the federates in a federation to interact with each other. Federate Interface Specification also introduces the Management Object Model (MOM), which provides facilities for federates to access RTI operating information during federation execution.
- *IEEE Std. 1516.2-2000: IEEE Standard for M&S HLA Object Model Template (OMT) Specification:* We present an overview in the next section.
- *IEEE Std. 1516.3: IEEE Recommended Practice for HLA Federation Development and Execution Process (FEDEP):* Provides guidance in the process of developing HLA federations.

Several OMT tools are available in the marketplace. Object Model Development Toolkit (OMDT) by AEgis Corporation, Calytrix SIMplicity by Calytrix Technologies, Visual OMT by Pitch Technologies AB, and GENESIS by the French National Air and Space Agency (ONERA) are examples. Presumably, each of these tools has some native representation scheme for object models. The point is to make these schemes explicit, in the form of a metamodel, and available to the users.

In this work, we have adopted a domain specific metamodeling approach. A metamodel is a definition of an architectural language in the form of a model [Kleppe et al. 2003]. We believe that tool integration and tool extensibility will be greatly facilitated by the use of metamodels that are accessible to users. Adoption of standardized metamodels will further improve integration and extensibility issues. Metamodeling is at the core of the approaches, such as Model Driven Architecture (MDA) and Model Driven Engineering (MDE) as well as Model Integrated Computing (MIC) [Bezivin et al. 2005].

This paper presents a study of developing an object model tool for HLA Object Model (OM) based on MIC.

Proceedings 20th European Conference on Modelling and Simulation
Wolfgang Borutzky, Alessandra Orsoni, Richard Zobel © ECMS, 2006
ISBN 0-9553018-0-7 / ISBN 0-9553018-1-5 (CD)

The study focuses on constructing a metamodel for HLA OM, expressed in the GME domain specific modeling environment. In GME, metamodels are defined as modeling paradigms, which are instances of MetaGME, the top level metamodel of GME. Upon loading the HLA OM paradigm, the MetaGME interpreter automatically creates an environment for HLA OM, which federate and federation developers can use to design their HLA object models.

As a practical application of the HLA OM paradigm, a "model interpreter" has been developed to generate FOM Document Data (FDD) files, which the RTI inputs prior to federation execution.

The remainder of the paper is organized as follows. The two succeeding sections provide overviews of the HLA Object Model Template and Model Integrated Computing. Then, the HLA OM paradigm is introduced and the resulting Object Model Design Environment (OMDE) is discussed. Next, the "interpreter" built for OMDE to generate FDD files is described. Finally, conclusions are drawn and future work is suggested.

HIGH LEVEL ARCHITECTURE OBJECT MODEL TEMPLATE

OMT introduces two types of object models: Simulation Object Model (SOM), one per federate, which describes the interests and capabilities of the federate, and Federation Object Model (FOM), one per federation, which describes the data that can be exchanged among the members of the federation. An HLA object model is composed of a group of interrelated components. The OMT Specification, IEEE Std. 1516.2, specifies the format, abstract syntax and information contents of HLA object models.

The standard presents object models in OMT tabular format. The OMT consists of 14 components which are presented as tables: Object model identification table, Object class structure table, Interaction class structure table, Attribute table, Parameter table, Dimension table, Time representation table, User-supplied tag table, Synchronization table, Transportation type table, Switches table, Datatype tables, Notes table, FOM/SOM lexicon. We refer the reader to the standard, where the contents of these tables are specified.

Differences between the HLA OM and object models encountered in object-oriented programming languages have been pointed out in the 1516.2. For an extensive discussion of the HLA OM, see [Tolk 2001]. A comparison of HLA and DEVS object models appears in [Sarjoughian and Zeigler 1999].

MODEL INTEGRATED COMPUTING

Model Integrated Computing (MIC) allows the synthesis of application programs from models by using customized domain-specific Model Integrated Program Synthesis (MIPS) environments [Nordstrom et al. 1999]. Once a domain-specific modeling language (in our case, HLA OM metamodel) has been formally defined, a meta-level translation can be performed to synthesize the Domain-Specific MIPS Environment (DSME) from the metamodel [Karsai et al. 2003]. The DSME (in our case, OMDE) is then used by domain experts (in our case, HLA-based distributed simulation developers) to create various models of domain-specific systems (e.g., a FOM). Given domain models, model interpreters perform semantic translations on them to generate executable models and other artifacts (e.g., an FDD file), or perform various types of analyses (e.g. constraint checking).

A MIPS environment operates according to a modeling paradigm. A modeling paradigm is a set of requirements that governs how systems within the domain are to be modeled. In other words, a modeling paradigm is a formal language, i.e. a metamodel, for modeling systems in the domain. In summary, MIC applies the metamodel based approach to domain specific applications.

We have used GME version 5.9.21 in this work. GME (Generic Modeling Environment) is described as a configurable MIPS tool, where configurable means that it can be programmed to work with vastly different domains [ISIS 2005]. It is available as free and open source software.

In GME models can include models, atoms, FCOs (first class objects), references, sets, and connections. The purpose of the GME is to create and manipulate these models. Modeling paradigms may have several kinds of models. MetaGME supports various techniques (hierarchy, multiple aspects, sets, references, and explicit constraints) for building large-scale, complex models.

The proposed metamodel for HLA OM is a GME modeling paradigm, based on MetaGME, the GME metamodel. GME generates a design environment for HLA object models automatically using the defined metamodel. This is achieved by the MetaGME Interpreter and we designate the generated design environment as Object Model Design Environment (OMDE). Figure 1 indicates how MIC process is applied to HLA OM.

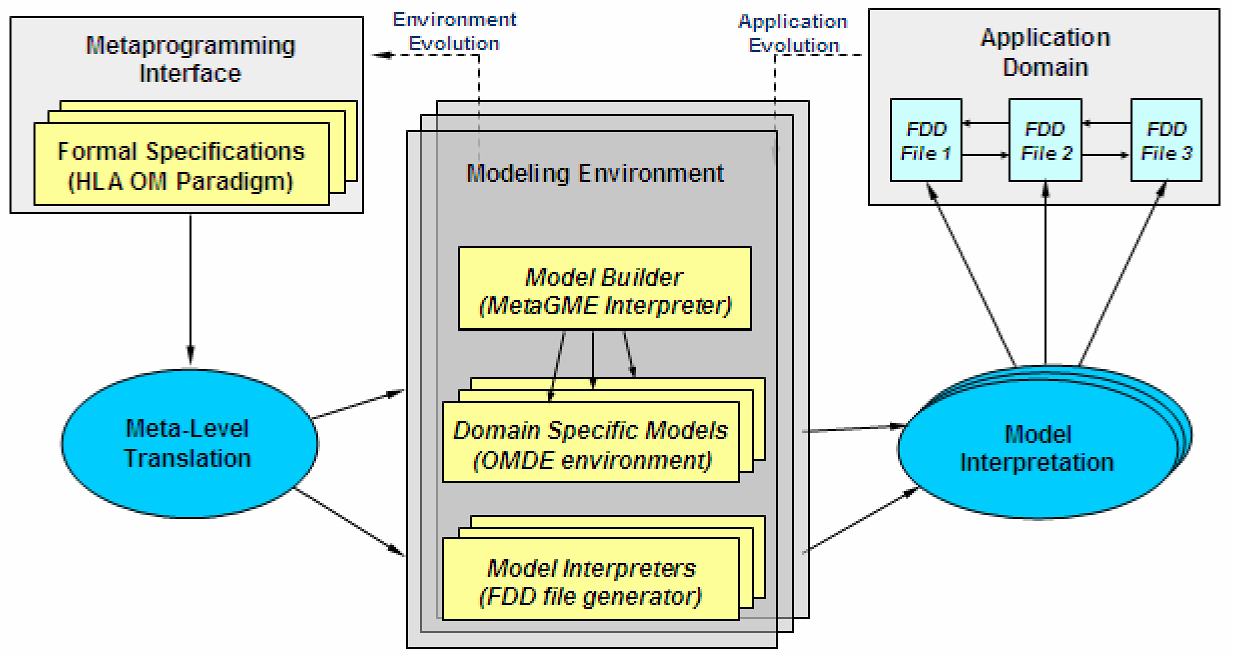
Metaprogramming Interface
Formal Specifications (HLA OM Paradigm)
Environment Evolution
Application Evolution
Application Domain
FDD File 1
FDD File 2
FDD File 3
Modeling Environment
Model Builder (MetaGME Interpreter)
Domain Specific Models (OMDE environment)
Model Interpreters (FDD file generator)
Meta-Level Translation
Model Interpretation

Figure 1: MIC Process applied to HLA OM (adapted from [Nordstrom et al. 1999])

HLA OM PARADIGM

This section introduces the HLA OM paradigm. The class diagrams, aspects, attributes and constraints are briefly discussed for each model. For the missing details the reader is referred to [Çetinkaya 2005], where the complete HLA OM metamodel including MOM, IEEE default library model, a sample federation design model, modeling rationale and a user's guide are presented.

The introduced metamodel fully accounts for IEEE Std. 1516.2. The modeling elements in our metamodel are named consistently with OMT Data Interchange Format (DIF), where DIF is defined with an XML Document Type Definition (DTD). The metamodel is compatible with the HLA OMT tabular format in the sense that every table column in the OMT tabular format has a corresponding, metamodel element. Together with the attached commentary, we regard the HLA OM metamodel as an alternative rendering of the 1516.2.

The HLA OM paradigm includes the `Object Model` paradigm sheet, the `Federation Design` paradigm sheet and the `OMT Core` folder. Paradigm sheets are separate models. `OMT Core` folder includes the definitions for OM identification, classes, data types, dimensions, normalization functions, notes, switches, synchronization points, user supplied tags, time representations and transportations in separate paradigm sheets. To provide a context for the HLA OM paradigm, we have also defined a rudimentary Federation Design metamodel. `Object Model` paradigm sheet and `OMT Core Folder` could be used separately from `Federation Design Model`.

There are some rules which apply to the whole metamodel. One of them is the naming constraint: Names in object models can be constructed from a combination of letters, digits, hyphens, and underscores with no spaces or other breaking characters. Names beginning with the string "hla" or any string that would match (('H'|'h') ('L'|'l') ('A'|'a')), are reserved and also the string "na" or any string that would match (('N'|'n') ('A'|'a')), is reserved. These may not be included as a user-defined name. We check this constraint for object class names, interaction class names, attribute names, parameter names, datatype names, enumerated datatype enumerators, enumerated datatype values, fixed record field names, variant record alternative names, basic data representation names, dimension names, transportation type names, synchronization point names, note identifying labels. Other names could be checked in the same fashion.

Another common constraint is the cardinality constraint. We check the cardinalities in terms of the upper bounds and the lower bounds. Violation of an upper bound is an error, and the user is not allowed to do the operation. Violation of a lower bound generates warnings, allowing the user to add the missing elements later.

Unique name constraints are applicable for various elements. If an element's name has to be unique throughout the model, this constraint causes error messages for duplicate names. We also checked for the null references, in order to detect the omitted elements. A null reference causes a warning.

In our metamodel some elements have standard attributes or names, which can not be changed. For example, the name of the object class HLAobjectRoot cannot be changed. Any violation causes errors or warnings according to the severity of the situation. Some attributes and some model parts are mandatory, and the user is expected to fill in these fields. Some attributes may be "NA", which means "not applicable". Such attributes as date, e-mail, etc., have a common format. We also check the validity of attributes and model parts.

All the constraints are expressed in OCL. Extended OCL support of GME helped us immensely to enhance the precision of the metamodel.

Succeeding sections describe the Federation Design Model, Object Model and OMT Core Folder in some detail.

Federation Design Model

The Federation Design Model (FDM) provides an interface to define a federation and the member federates and to connect them to their respective FOM and SOMs. In FDM, FOM and SOMs are referenced object models. Each design can include one federation and one FOM reference, with any number of federates and their respective SOMs.

There is a "MemberOf" connection between federation and federates. This connection presents the federation execution capabilities. FDM is not intended to be complete; it is constructed simply to illustrate the usage of object models in context.

Object Model

The Object Model paradigm sheet includes the main diagram for object models. There are three types of object models, namely, FOM, SOM and Other. FOM and SOM are as given in the HLA OMT specification. The Other type provides a template for "temporary" object models -not to be included in the Federation Design model. Figure 2 shows a simplified view of the Object Model top level.

GME allows large models to be partitioned into "aspects". A certain set of model elements become visible when that aspect is on. Object models have six aspects, namely Classes, User-Supplied Tags, Synchronization, Switches, Time Represen-tation and OM Identification. In each aspect, the model definitions are taken from the related paradigm sheets with proxy elements, which are packaged under the OMT Core folder.

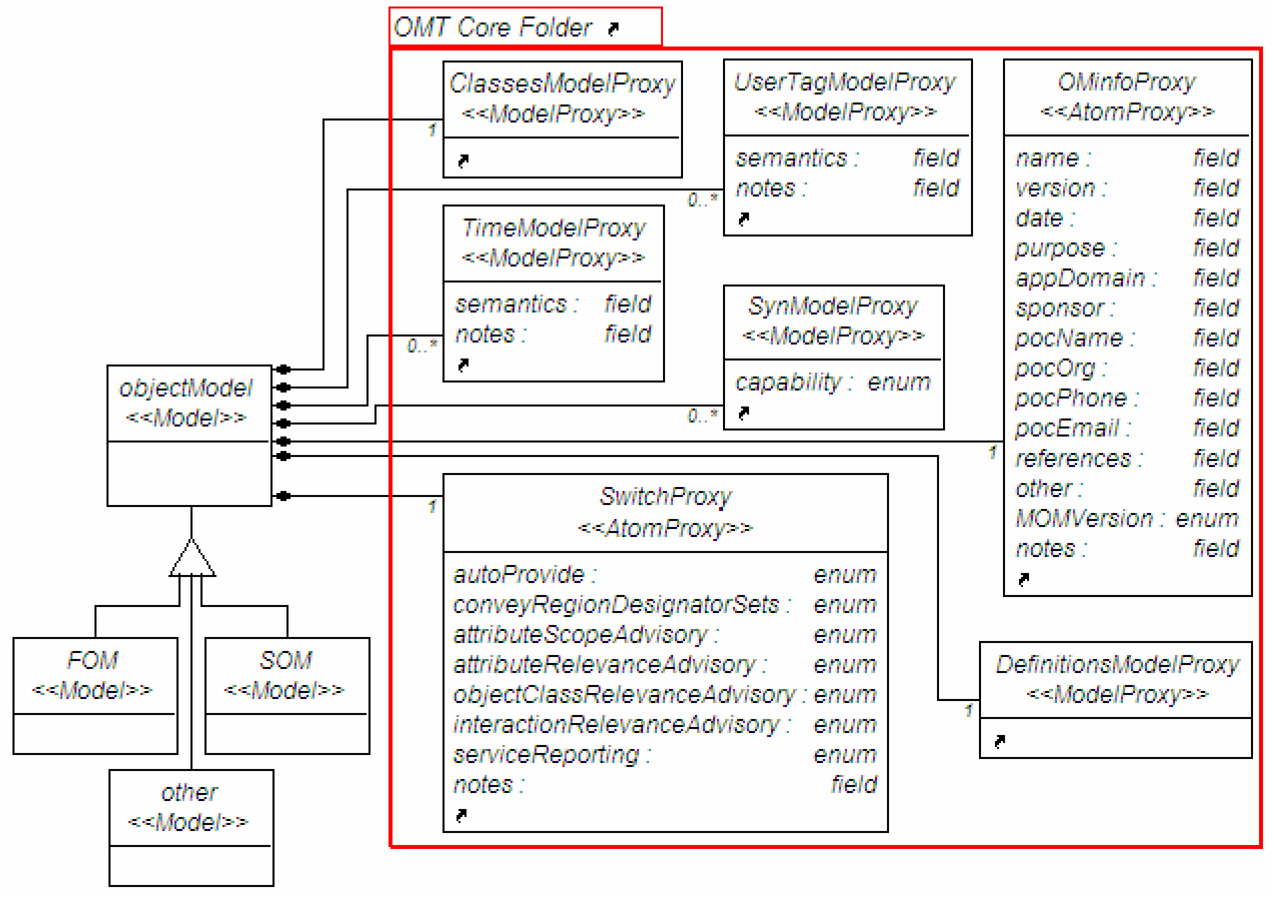

Figure 2: Object Model top view

The `Classes` aspect includes the definitions of object classes, attributes, interaction classes and parameters. It also includes FOM/SOM lexicon definitions. The `User-Supplied Tags` aspect includes the user-supplied tag elements. The `Synchronization` aspect includes the definition of synchronization point models. The `Switches` aspect includes the switches in order to change the initial settings. The `time representation` aspect includes the definitions of lookahead and timestamp models. Proxy elements referring to the related models can be seen in Figure 2.

OMT Core Elements

The `OMT Core` folder includes the basic elements needed to define an object model, namely `OM identification` model, `classes` model, `datatypes` model, `dimensions` model, `normalization functions` atom, `notes` atom, `switches` atom, `synchronization points` model, `user supplied tags` model, `time representations` model, `transportations` model and FOM/SOM lexicon `definitions` model.

The predefined datatypes and transportation types are modeled in IEEE default library. This library can be attached to all federation design models and provide the default entries in the 1516.2 [Çetinkaya 2005].

Figure 3 shows a simplified view of the `Classes Model`.

Management Object Model

Management Object Model provides facilities for joined federates to access RTI operational information during federation execution. MOM specification, given in the 1516.1, employs the OMT tabular format. So we have readily modeled it in our metamodel. MOM provides some default model elements for some services, such as publishing object classes; registering object instances and updating values of attributes of those object instances; subscribing to and receiving some interactions; or publishing and sending other interactions. The classes, attributes, parameters, datatypes and dimensions in MOM are modeled completely. This serves as a test of our metamodel as well.

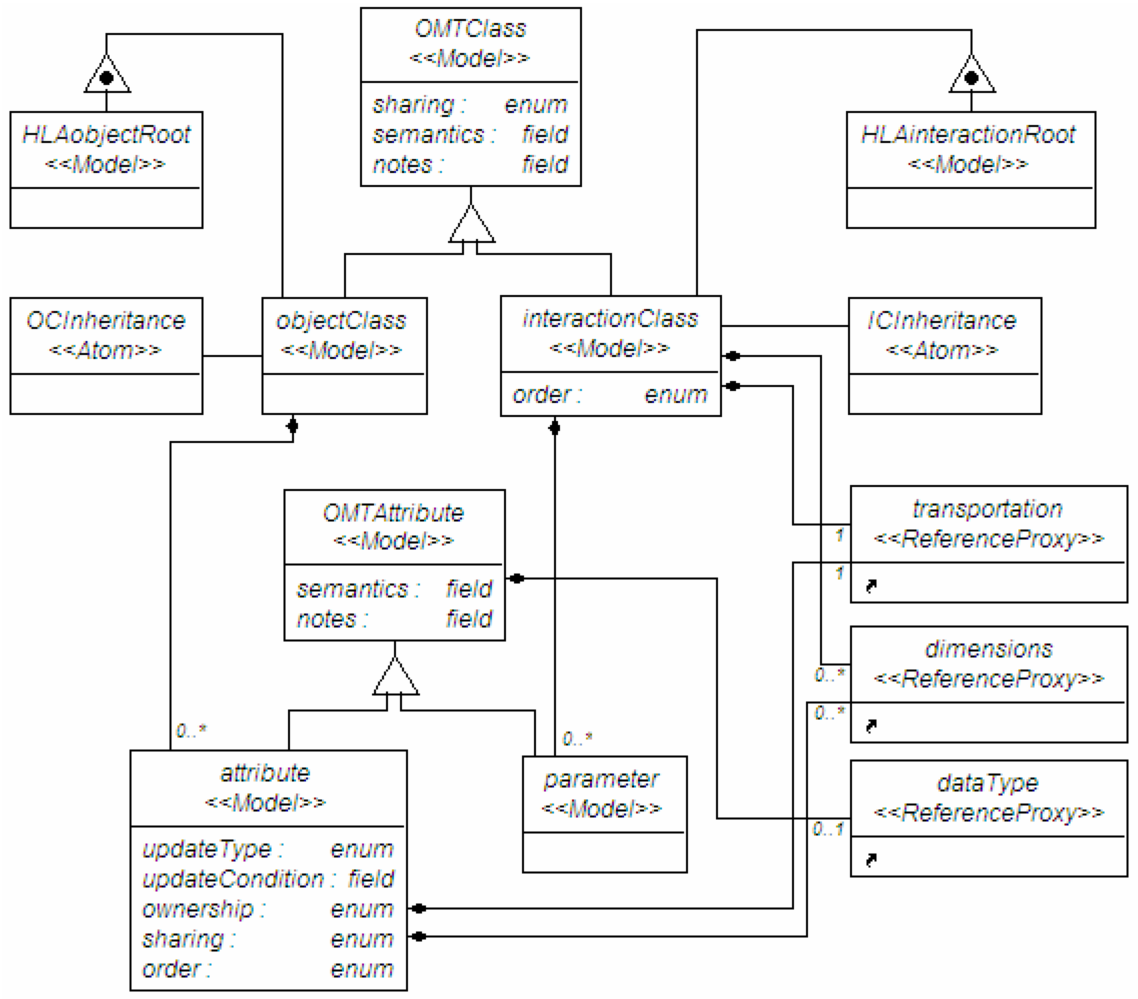

Figure 3: Classes Model (simplified)

OBJECT MODEL DESIGN ENVIRONMENT (OMDE)

From the HLA OM metamodel GME's Meta Interpreter generates the Object Model Design Environment (OMDE) automatically. OMDE serves as a design tool to provide individual federates and federations with SOMs and FOMs, respectively. OMDE also allows modeling simple federation designs for illustrative purposes. The object models defined with OMDE are the domain models of the MIC process. So they can be used in available model transformation tools, such as GReAT [Agrawal et al.], and model interpreters.

We have modeled the sample restaurant federation, which is used throughout the 1516.2 to illustrate the OMT tabular format. We have extended this example with a few additional modeling elements and a simple federation design model to have a more involved test case.

Figure 4 shows a screenshot from OMDE. On the right side of the screen the tree browser shows the datatypes, dimensions and transportations. On the left side, the model shows some of the object classes of Restaurant Federate SOM. The modeling elements for object models can be seen at the bottom of the figure.

GME has a decorator facility for nicer visualization of the models. We used the standard Meta Decorator, although more advanced decorators can be defined and used.

INTERPRETING OMDE MODELS

In this section we discuss the generation of Federation Object Model Document Data (FDD) files based on the HLA OM paradigm.

Object model development tools, such as OMDT, can convert between the OMT tabular representation of the object models and the data interchange format (DIF) representation. With that it is possible to automatically generate the FDD file that is required by the RTI to start a federation execution.

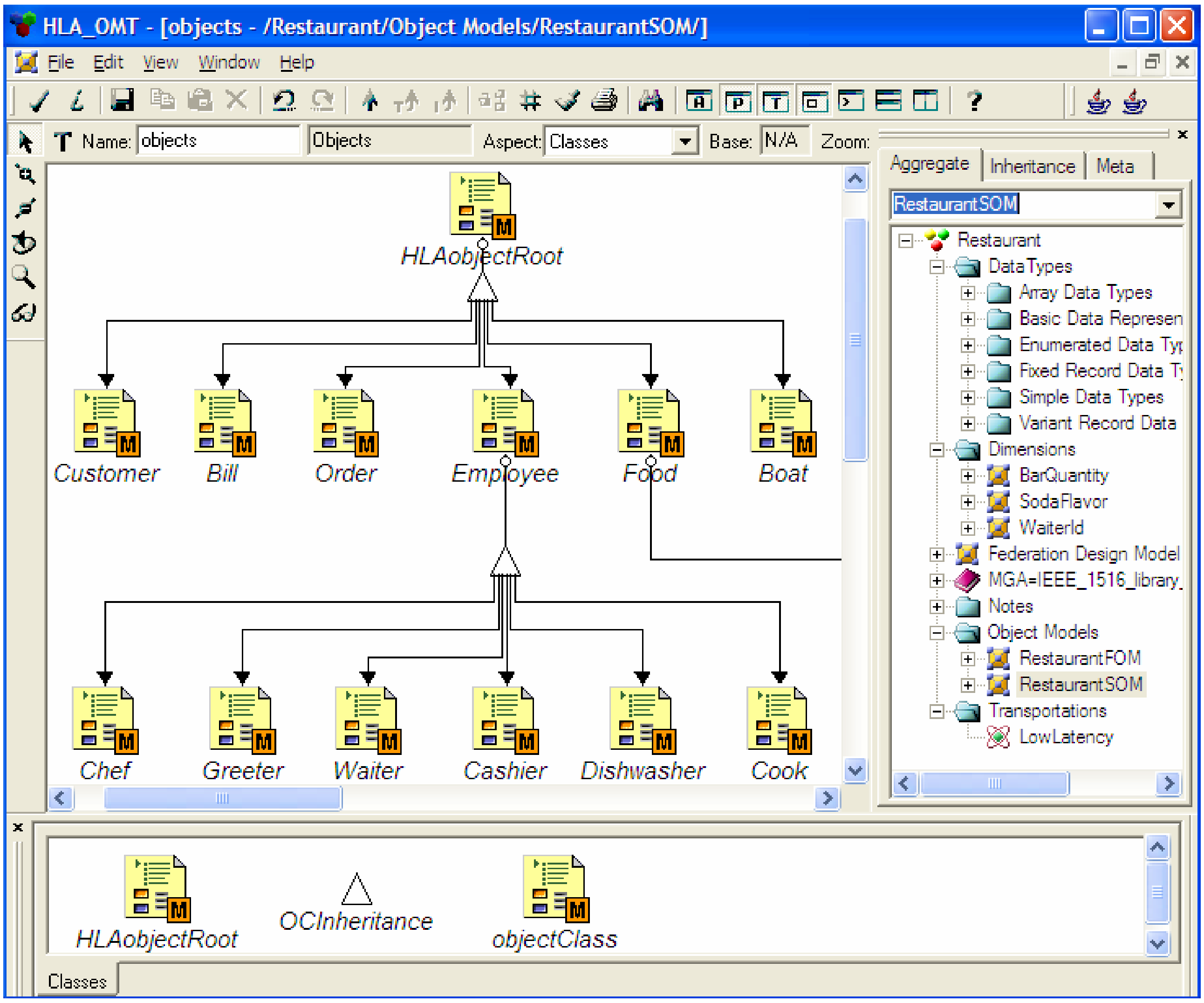

Figure 4: Sample object model overview

In our study, a pluggable component ("model interpreter" in GME terminology) that generates the FDD file for a federation design model or an object model, particularly a FOM, is developed. This interpreter is written in Java, although it could be written in C or C++ as well. It uses the Builder Object Network (BON) API provided with GME.

The file generated is fully compliant with the data interchange format (DIF) that is defined in the 1516.2. So it can be used by any compliant OMT tool to import object models. This is successfully tested with OMDT.

CONCLUSION

Defining a metamodel for HLA OM has been an exercise in domain-specific modeling. The main contribution of this effort is a metamodel, in the form of a GME paradigm, for HLA object models. This forms the foundation for federation object model design tools, which can be a part of a federation development toolkit, by applying model integrated computing to HLA.

The proposed metamodel is in complete agreement with IEEE Std. 1516.2, due, in part, to constraints expressed in OCL. Our metamodel can support virtually any presentation of an object model compliant with the 1516.2. All that is required is to provide a generator based on model traversal, which is supported through an API in GME. The interpreter that generates OMT DIF formatted FDD file serves as an example on how to do it.

Applying MIC to HLA opens up new possibilities considering model transformations and code generation, for which the proposed metamodel can be directly used. The researchers at Vanderbilt University developed a model-to-model transformation language, The Graph Rewriting And Transformation language (GReAT) [Agrawal et al.], and a meta-programmable transformation tool that supports the development of translators based on graph transformations. In a separate but related work we are trying to map, by using GReAT, a particular domain model, unrelated to HLA or any simulation platform, to alternative FOMs reflecting federation design choices. In this application, the HLA OM metamodel serves as the target metamodel for the transformation [Ozhan and Oguztuzun].

We believe that an integrated, open and extensible environment for HLA based distributed simulations, supporting all activities in FEDEP, can be constructed by adopting the metamodeling approach. Having access to the models and model transformations will enable users to customize the environment according to their own needs.

ACKNOWLEDGEMENTS

The authors benefited greatly from discussions with Okan Topçu, Mehmet Adak and Gürkan Özhan.

REFERENCES

Agrawal A.; G. Karsai; Z. Kalmar; S. Neema; F. Shi; A. Vizhanyo.. "The Design of a Language for Model Transformations". In review.

Bezivin J.; C. Brunette; R. Chevrel; F. Jouault; and I. Kurtev. 2005. "Bridging the Generic Modeling Environment and the Eclipse Modeling Framework". In Proceedings of the 4th workshop in Best Practices for Model Driven Software Development, OOPSLA, San Diego, California.

Çetinkaya D. 2005. "A Metamodel for the High Level Architecture Object Model". MS Thesis, Department of Computer Engineering, Middle East Technical University, Ankara, Turkey.

ISIS (Institute for Software Integrated Systems). 2005. GME 5 User's Manual. Vanderbilt University, Tennessee.

Karsai G.; A. Agrawal; F. Shi; and J. Sprinkle. 2003. "On the Use of Graph Transformations for the Formal Specification of Model Interpreters". In Universal Computer Science Journal, vol. 9, no: 11, pp. 1296-1321.

Kleppe A.; S. Warmer; and W. Bast. 2003. MDA Explained - The Model Driven Architecture: Practice and Promise. Addison-Wesley, Boston.

Nordstrom G.; J. Sztipanovits; G. Karsai; and A. Ledeczi. 1999. "Metamodeling - Rapid Design and Evolution of Domain-Specific Modeling Environments". In the Proceedings of the IEEE ECBS'99 Conference, Tennessee.

Ozhan G. and H. Oguztuzun. "Model-Integrated Development of HLA-based Field Artillery Simulation". Submitted.

Sarjoughian H.S. and B.P. Zeigler. 1999. "The Role of Collaborative DEVS Modeler in Federation Development". In the Proceedings of 1999 Fall Simulation Interoperability Workshop.

SISC (Simulation Interoperability Standards Committee). 2000. IEEE Std. 1516.2, IEEE Standard for Modeling and Simulation(M&S) High Level Architecture(HLA) - Object Model Template Specification, IEEE Computer Society.

Tolk A. 2001. "HLA-OMT versus Traditional Data and Object Modeling". In the Proceedings of the 2001 Command and Control Research and Technology Symposium, Maryland.

AUTHOR BIOGRAPHIES

DENİZ ÇETİNKAYA received her M.Sc. in Computer Engineering from the Middle East Technical University in 2005. She received her B.Sc. with honors in Computer Engineering from the Hacettepe University, Ankara, in 2002. Her research interests include design support tools development, metamodeling, modeling and simulation.E-mail: e131263@ceng.metu.edu.tr.

HALİT OĞUZTÜZÜN is an assistant professor in the Department of Computer Engineering at the Middle East Technical University (METU), Ankara, Turkey. He obtained his BS and MS degrees from METU in 1982 and 1984, and PhD from University of Iowa, Iowa City, IA, USA in 1991. His current research interests include distributed simulation, and software and systems modeling. E-mail: oguztuzn@ceng.metu.edu.tr.