

2014

Simulation Generator (SimGe) – 0.2.6

User Manual



Okan Topçu

7/6/2014



Series

2

SIMULATION GENERATOR

User Manual – D R A F T

Simulation Generator (SimGe)

To my son.

Table of contents

Table of contents.....	1
List of figures.....	4
List of tables.....	7
Foreword.....	8
1. Introduction.....	10
1.1 Overview.....	10
1.2 Typeface Conventions	10
1.3 Summary of Chapters	11
2. Installation	12
2.1 Hardware and Software General Prerequisites	12
2.2 Installation	12
2.3 Uninstallation.....	14
3. Project Management	15
3.1 The Main User Interface	15
3.2 Creating a Project.....	16
3.2.1 Project Home Folder (Project Location).....	16
3.3 Loading an Existing Project.....	16
3.4 Loading a Sample Project	17
3.5 Saving and Closing a Project	17
3.6 Saving as another Project.....	17
3.7 Project Start Page.....	17
3.8 Project Settings	17
3.9 Options.....	18
4. Object Model Constuction	20
4.1 Loading an Existing SimGe Object Model.....	20
4.2 Removing Object Model.....	21
4.3 Creating a Federation Object Model from Scratch	21
4.4 Importing a FED/FDD File.....	21
4.4.1 Mappings	22
4.5 Exporting the FED/FDD Files	23
4.6 Object Model Editor	23
4.6.1 OME Toolbar	24
4.6.1.1 Removing the Current Federation Object Model	25
4.7 Table Editor	25
4.7.1 Object Model Identification.....	27
4.7.1.1 POC.....	28
4.7.1.2 Keywords	29

4.7.1.3	References	29
4.7.1.4	Use History.....	29
4.7.1.5	Glyph.....	30
4.7.2	Objects	30
4.7.3	Interactions	31
4.7.4	Attributes	32
4.7.5	Parameters.....	34
4.7.6	Dimensions	35
4.7.7	Time Representations	36
4.7.8	User-supplied Tags	36
4.7.9	Synchronizations.....	37
4.7.10	Transportations	38
4.7.11	Update Rates	38
4.7.12	Switches	39
4.7.13	Data Types	39
4.7.13.1	Basic Data Representations	39
4.7.13.2	Simple Data Types	41
4.7.13.3	Enumerated Data Types	42
4.7.13.4	Array Data Types	43
4.7.13.5	Fixed Record Data Types	43
4.7.13.6	Variant Record Data Types	44
4.7.14	Notes.....	45
4.7.15	Interface Specification Services.....	46
4.7.16	OMT 1.3 Support.....	46
4.8	Textual View.....	47
4.8.1	Textual View for FED and FDD Files.....	47
4.9	Validating FDD File	50
4.10	MOM Integration.....	50
4.11	OM Explorer	51
4.11.1	Traversing Functionality.....	53
4.11.2	Modifying Functionality.....	53
5.	Federation Architecture Modeling Environment	54
5.1	Creating a FAM	54
5.2	Federation Structure.....	55
5.2.1	Federation Execution Properties	55
5.2.2	Federate Applications	55
5.2.3	Setting Project FOM	56
5.3	FAM Explorer.....	57
6.	Code Generator	58
6.1	User Interface.....	58
6.1.1	Code Explorer	58

6.1.2	Code Viewer	59
6.2	Architecture Style	60
6.2.1	Federate Object Model.....	62
6.2.1.1	Object/Interaction Classes.....	62
6.2.2	Simulation Layer Code Generation	62
6.2.2.1	Application-specific Federate Class.....	62
6.2.3	Management Object Model Code Generation	62
6.3	Code Generator Configuration.....	62
6.3.1	General Settings.....	62
6.3.2	Callback Settings	64
7.	Report Generator.....	66
8.	Case Studies	68
8.1	Chat Federation.....	68
8.1.1	Loading Sample Project.....	68
8.2	Strait Traffic Monitoring Simulation (STMS).....	69
8.2.1	Object Model	70
8.2.2	Federation Architecture	70
8.2.3	Data Distribution Management.....	71
	References.....	72
	List of symbols/abbreviations/acronyms/initialisms.....	74
	Index	75

List of figures

Figure 1. License Agreement Dialog	13
Figure 2. Destination Folder Screen	13
Figure 3. Product Features Dialog	14
Figure 4. Installation Completed Dialog	14
Figure 5. An Empty Project UI	15
Figure 6. Create-Project Wizard	16
Figure 7. Project Start Page	17
Figure 8. Project Settings - General	18
Figure 9. Object Model Create/Import	21
Figure 10. Import of a FED/FDD File	22
Figure 11. Import Validation Report	22
Figure 12. Export Configuration	23
Figure 13. Validation Results	23
Figure 14. Multiple OMEs	24
Figure 15. OME Toolbar	25
Figure 16. Read-only OMT Elements	26
Figure 17. Adding/Removing an OMT Element	26
Figure 18. Object Class Data Input Screen	27
Figure 19. Value Selection	27
Figure 20. Table editor	28
Figure 21. POC Details	29
Figure 22. Keywords	29
Figure 23. References	29
Figure 24. Use History	30
Figure 25. Objects	30
Figure 26. Object Class Data Input Screen	31
Figure 27. Interactions	31
Figure 28. Interaction Class Data Input Screen	32
Figure 29. Attributes	33
Figure 30. Attribute Data Input Screen	34
Figure 31. Parameters	35
Figure 32. Parameter Data Input Screen	35
Figure 33. Dimensions	35
Figure 34. Dimension Data Input Screen	36

Figure 35. Time Representations	36
Figure 36. User-supplied Tags	37
Figure 37. Synchronizations	37
Figure 38. Synchronization Data Input Screen	38
Figure 39. Transportations	38
Figure 40. Transportation Data Input Screen.....	38
Figure 41. Update Rate Data Input Screen	39
Figure 42. Switches.....	39
Figure 43. Basic Data Representations	40
Figure 44. Data Representation Data Input Screen.....	40
Figure 45. Simple Data Types.....	41
Figure 46. Simple Datatype Data Input Screen.....	42
Figure 47. Enumerated Data Types.....	42
Figure 48. Enumerated Datatype Data Input Screen.....	43
Figure 49. Array Data Types	43
Figure 50. Array Datatype Data Input Screen.....	43
Figure 51. Fixed Record Data Types	44
Figure 52. Fixed Record Datatype Data Input Screen	44
Figure 53. Variant Record Data Types	45
Figure 54. Variant Record Datatype Data Input Screen	45
Figure 55. Interface Specification Services	46
Figure 56. OMT 1.3 Support Tab	47
Figure 57. OME FED Viewer.....	48
Figure 58. OME FDDViewer.....	49
Figure 59. Find Dialog.....	50
Figure 60. MOM Suppression.....	50
Figure 61. MOM.....	51
Figure 62. OM Explorer.....	52
Figure 63. OM Explorer Context Menu.....	53
Figure 64. FAM Creation.....	54
Figure 65. FAME User Interface	55
Figure 66. Federate Application Properties	56
Figure 67. Setting Project FOM.....	57
Figure 68. Code Explorer.....	59
Figure 69. Code Viewer and Generated Code Sample	60
Figure 70. Federate Application Design [5].....	60

Figure 71. Class Diagram	61
Figure 72. Application-specific Federate Class	62
Figure 73. Code Generator Settings - General	63
Figure 74. Generated Code Parameters.....	64
Figure 75. Code Generator Settings - Callbacks.....	65
Figure 76. Running Report Generator.....	66
Figure 77. Report Generator Tab	67
Figure 78. Chat Application Conceptual View	68
Figure 79. Loading Sample Project.....	68
Figure 80. Strait Traffic Monitoring Simulation Conceptual View	69
Figure 81. A Part of the STMS FOM.....	70
Figure 82. The Strait Traffic Monitoring Federation Structure Model	71
Figure 83. DDM Example.....	71

List of tables

Table 1. Hardware Requirements.....	12
Table 2. Software Requirements	12
Table 3. Code Generator General Settings.....	18
Table 4. Federation Execution Properties	55
Table 5. Federate Application Properties.....	56
Table 6. Code Generator General Settings.....	63

Foreword

Simulation Generator (SimGe) is a fully-dressed High Level Architecture (HLA) object model editor, simulation design and development environment, and a code generator that is intended to generate code automatically for HLA based distributed simulations. The target platform for code generation is an HLA Runtime infrastructure (RTI) abstraction layer called RTI abstraction component for .NET (RACoN). The architecture of the generated code by SimGe conforms to the layered architectural style.

Current capabilities of SimGe are presented in the following paragraphs.

- Object Model Editor (OME)
 - OME supports both OMT 1.3 specification and OMT 1516.2010 specification.
 - OME allows creation of an empty SimGe project object model.
 - OME allows import of an existing SimGe project object model.
 - OME allows import and export of an HLA 1.3 FED file.
 - OME allows import, export, and validation of an HLA 1516.2010 FDD file. The user interface support for editing and modifying notes element is not provided yet.
 - OME allows modification of the object model using a table-style user interface.
 - OME supports interface specification services usage.
 - OME validates user data input according to HLA 1516-2.2010 specification.
 - OME supports multiple object models. User can work with multiple versions of FOM at the same time with multiple OMEs.
 - Each OME has its own toolbar.
 - Federation Architecture Modeling Environment (FAME)
 - Federation execution properties such as federation execution name can be specified.
 - Federate applications can be added/removed to the Federation Architecture Model (FAM).
 - Code Generator (CodeGen)
 - Code is separately generated for each federate application found in the FAM according to its SOM.
 - The structure of the generated code conforms to the layered architecture.
 - The following code is generated:
 - Application-specific federate class,
 - The skeleton code for federate ambassador callback event handlers for [RACoN](#),
 - A class for each HLA class found in the object model and supports the inherited classes,
 - The federate SOM class,
 - The simulation Manager class.
 - Code generation configuration dialog enables the user to select which management services that the callback handler code will be generated.
 - FED file is automatically exported to the source code folder.
 - Report Generator (RG)
-

- RG fully generates all HLA OMT 1516.2010 tables as well as interface specification services.

This document is intended for the users of Simulation Generator (SimGe) software.

Please note that this is a research and an academic tool that is not intended for a production environment. It is mainly developed as a lab tool for a graduate level distributed simulation course.

No support is guaranteed. Use it at your own risk.

Also note that SimGe versions do not follow backward compatibility policy. This means that a new version of SimGe may not load an older object model, which may cause loss of work.

You can influence this project by sending your feedbacks to otot.support@outlook.com or okantopcu@gmail.com.

1. Introduction

“One doesn’t discover new lands without losing sight of the shore”

Andre Gide

1.1 Overview

SimGe is a simulation design and development environment for High Level Architecture (HLA) based distributed simulations [1] [2]. SimGe includes an object model editor and a code generator. The editor allows the user to manage the object model and enables the creation and modification of HLA object model template and object models and the import and export of the HLA related files (i.e. Federation Execution Details (FED)¹, Federation Object Model (FOM) Document Data (FDD)²), which contains configuration data for the federation execution. The code generator automatically generates code for the target platform, which is an HLA Runtime infrastructure (RTI) abstraction layer called RTI abstraction component for .NET (RACoN)³ [3]. The architecture of the generated code by SimGe conforms to the layered simulation architecture as specified in [4] and [5].

The product highlights:

- Modern User Interface — Easy Use
- Table-style Object Model Editor (see Chapter 4)
- Support for Multiple FOMs
- Project-based Environment (see Chapter 3)
- Sample Project – A sample project is provided for a fast start.
- Step-by-step User Manual (this manual)
- Validation Services (see Section 4.8)
- Compatible with RACON

1.2 Typeface Conventions

This technical document uses the following typeface conventions:

- All code examples/snippets are printed in a Book Antiqua Font.
- At the first introduction or definition of a major term, the term is shown in *italics*.
- All references to classes, attributes, and other elements of a model are shown in Courier New Font.
- General emphasis is shown in *italics*.

¹ FED file is used by RTIs that conform to the HLA 1.3 specification.

² FDD file is used by RTIs that conform to the HLA 1516 specification.

³ <http://www.ceng.metu.edu.tr/~otopcu/racon/>

1.3 Summary of Chapters

The remaining chapters are broken down as follows:

- Chapter 2 gives the hardware and software requirements and the installation steps.
- Chapter 3 explains the simulation project management.
- Chapter 4 focuses on the object model editor.
- Chapter 5 explains the code generator.
- Chapter 6 explains the report generator.
- Chapter 7 gives a case study.

Please, be aware that some screenshots in this document can belong to the older versions of SimGe.

2. Installation

2.1 Hardware and Software General Prerequisites

SimGe does not require a specific or high-end hardware requirement. Table 1 presents a typical configuration.

Table 1. Hardware Requirements

COMPONENT	ABSOLUTE MINIMUM CONFIGURATION	RECOMMENDED MINIMUM CONFIGURATION
RAM	512 MB	1 GB
Hard Disk Space	20 MB	32 MB
Video Graphics Card	-	-

On the software side, the requirements are presented in Table 2.

Table 2. Software Requirements

COMPONENT	ABSOLUTE MINIMUM CONFIGURATION
Operating System	Windows 8.1
Runtime Environment	Microsoft .NET Framework 4.0 Client Profile Package
Installation needs local administration rights.	

2.2 Installation

Extract the zipped file to a folder. Right click to SimGe.msi and select install. Program installer will check that SimGe is already installed. If a current version of SimGe installation is detected, the installer will ask either to remove it or to repair it. If the previous version of SimGe installation is detected, then the installer will uninstall it first.

The installation program is prepared with Windows Installer XML (WiX)¹ toolset. The installation begins with License Agreement screen as depicted in Figure 1.

¹ <http://wix.sourceforge.net/>

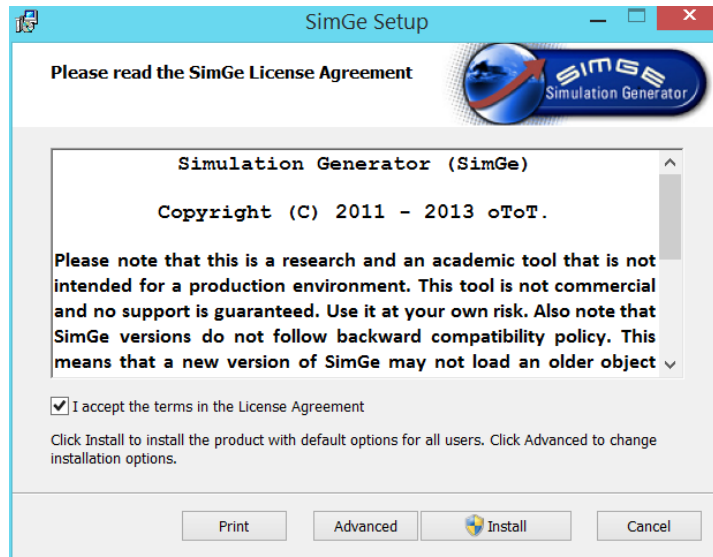


Figure 1. License Agreement Dialog

Here the user must accept the terms in the license agreement and then he/she can proceed with install or select the advanced setup options by clicking advanced button. In case of clicking Install button, the installation will begin and an installation completed dialog will appear as depicted in Figure 4. When Advanced button is clicked, you can select a different installation folder than the default one, if required (see Figure 2).

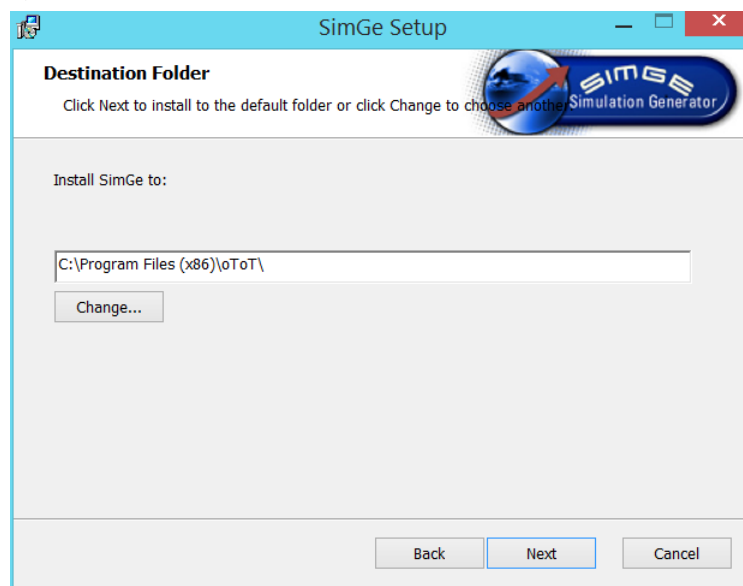


Figure 2. Destination Folder Screen

After clicking Next button, Product Features dialog will appear as in Figure 3. Currently, SimGe is one package. Just click Next in this screen.

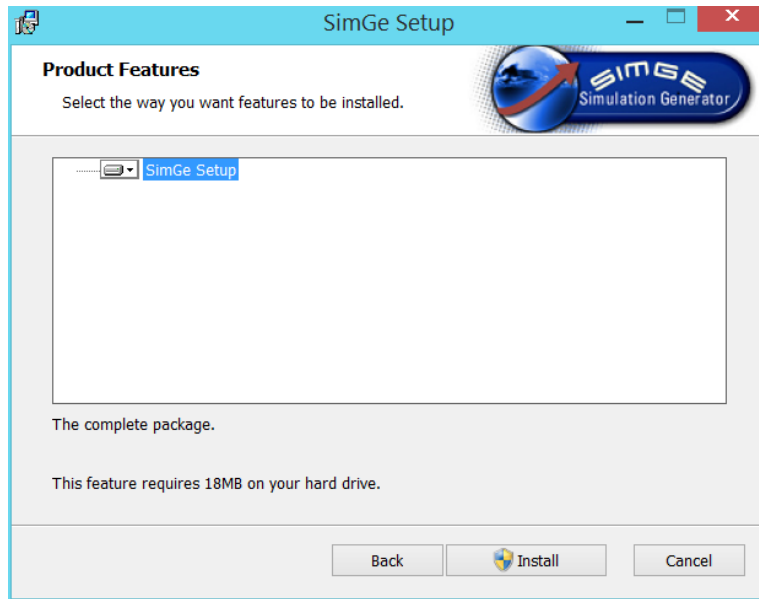


Figure 3. Product Features Dialog

The installation will end with a successful message as seen in Figure 4. The user may continue with the launch of SimGe.

Please note that the installation wizard associates .fap (SimGe Project file extension) files with SimGe.

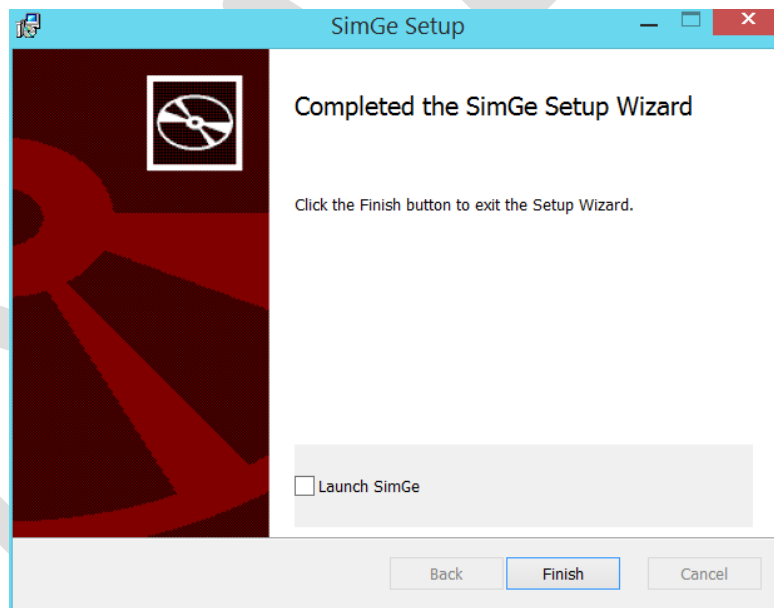


Figure 4. Installation Completed Dialog

2.3 Uninstallation

The user can uninstall SimGe by using Windows Programs and Features or the shortcut to uninstall SimGe at SimGe folder in Programs Menu.

3. Project Management

SimGe organizes all the simulation generation efforts using simulation projects. A *Simulation Project* is the main container and organizer for the simulation design and code generation. A simulation project is composed of the structural and the behavioral parts of a federate architecture. A project file saves all the development data and the projects settings. Therefore, whenever running SimGe, the user must either create a new project or open an existing one.

3.1 The Main User Interface

The project user interface has two main sections divided by a vertical slider. On the right side, the project explorer is located and on the left side the project workspace is displayed (see Figure 5).

The project workspace area includes the various workspaces used in the construction of the simulation in a tab view environment. The workspaces currently supported are (1) New Project Wizard, (2) Start Page, (3) Object Model Editor, and (4) Project Settings.

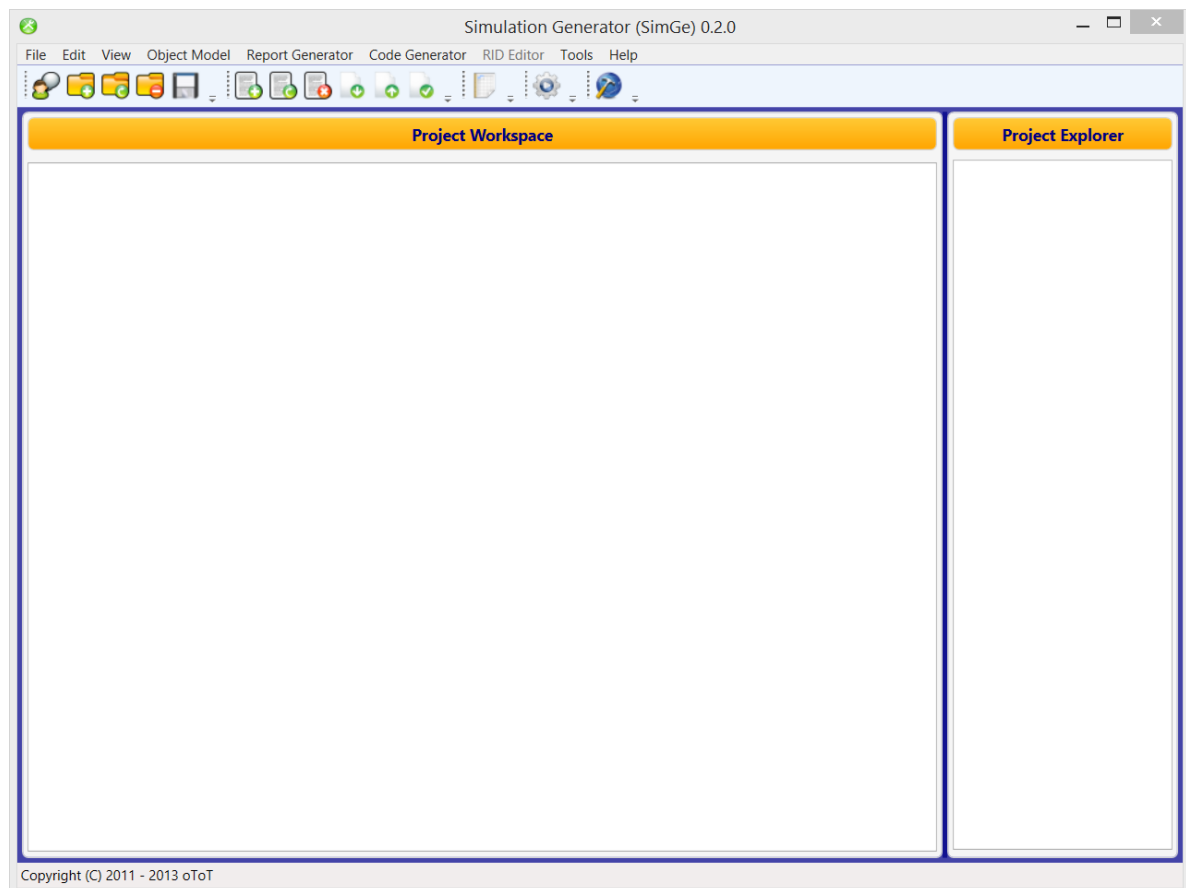


Figure 5. An Empty Project UI

The project explorer area includes the various explorers. An explorer is a tree view of the simulation project areas such that FOM management. Only, FOM explorer is supported.

3.2 Creating a Project

After running the SimGe application, the user can select to create a new project either by using the File menu or by pressing the new project icon at the toolbar menu. The create-project wizard is depicted in Figure 6. Here, the user enters a name and select the root folder, where the project related folders will be created..

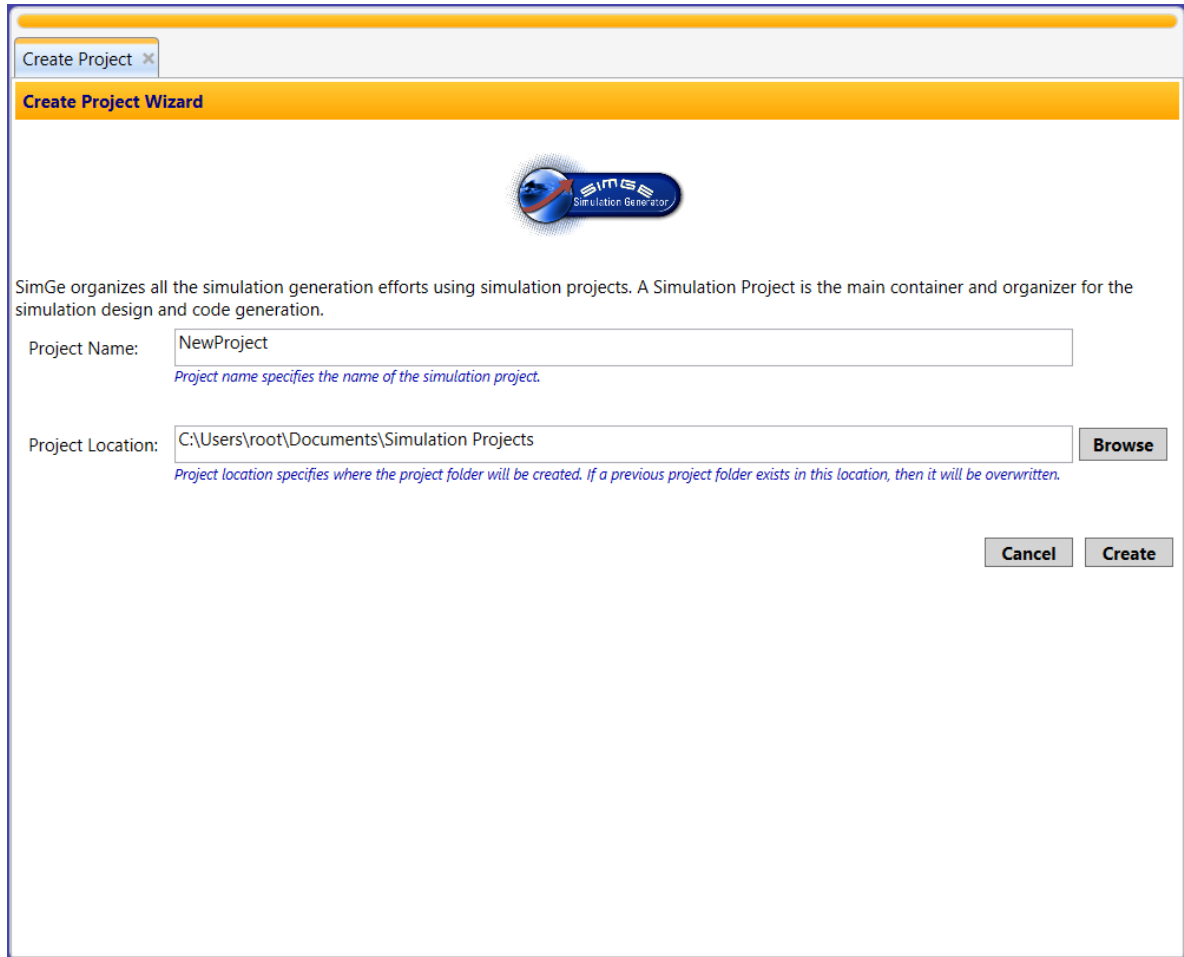


Figure 6. Create-Project Wizard

3.2.1 Project Home Folder (Project Location)

The default *project home folder* takes the same name with the project name and is located under the Simulation Projects folder found in the user's "my documents" folder. The project home folder includes the Object Model and Source Code folders. The FOM folder contains the Object Model Template (OMT) documentation and FOM structure. The Source Code folder contains the generated code source files.

3.3 Loading Project

3.3.1 Loading an Existing Project

In order to load an existing project, press open project button. An open dialog will appear to select the existing project file. The project files are xml files with ".fap" extension. After selecting the project file, SimGe will load all the project files. FAP stands for Federation Architecture Project.

3.3.2 Loading a Sample Project

SimGe provides a default sample project called Chat Federation. See Chapter 7 for the detail explanations of chat federation case study.

3.3.3 Loading Recently Used Project

SimGe keeps the most recently used projects (called Most Recently Used (MRU) list) and displays them in File menu. The MRU list currently is kept in an xml file (i.e. RecentProjects.xml) in the operating system common application data folder (i.e. generally C:\ProgramData\SimGe).

The user may adjust the max number of files to be kept using Options menu.

3.4 Saving and Closing a Project

Use the save or close buttons found in the File menu. SimGe will save the project file in the project home folder. When close button is pressed and there are some open project files that need to be saved, then SimGe will save them before exiting the application. The save process, particularly in the first save, can take a while due to the serialization of the object model in the project.

3.5 Saving as another Project

Use Save As command in File menu and select the folder and a new project name for the project. Note that SimGe only saves the project file and object model as one can always regenerate code and reports.

3.6 Project Start Page

After creating a new project, the project user interface is loaded and the *Project Start Page* is displayed (Figure 7). The project start page currently displays project and FOM information. It shows the user the default folders that are set for the project. All folder paths are clickable links whereas the folder is opened when clicked.

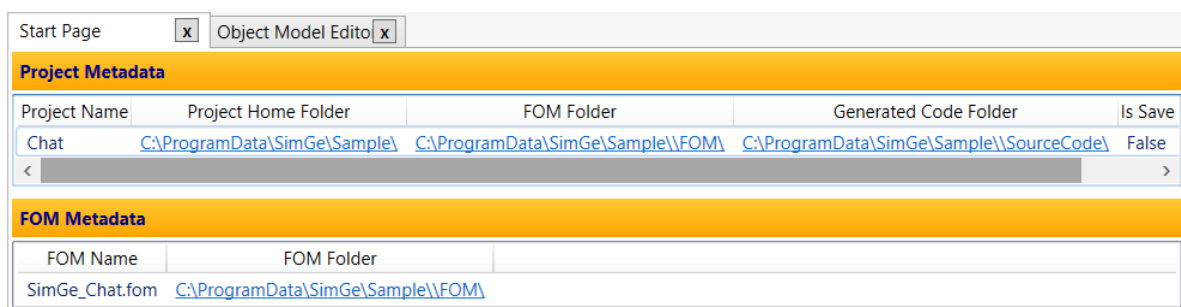


Figure 7. Project Start Page

3.7 Project Settings

Project settings are project-specific and they are not saved. Project settings include the project and the code generator settings. See Section 6.3 for details about code generation configuration parameters.

The project settings contain two groups: (1) the project settings, and (2) folders as seen in Figure 74.

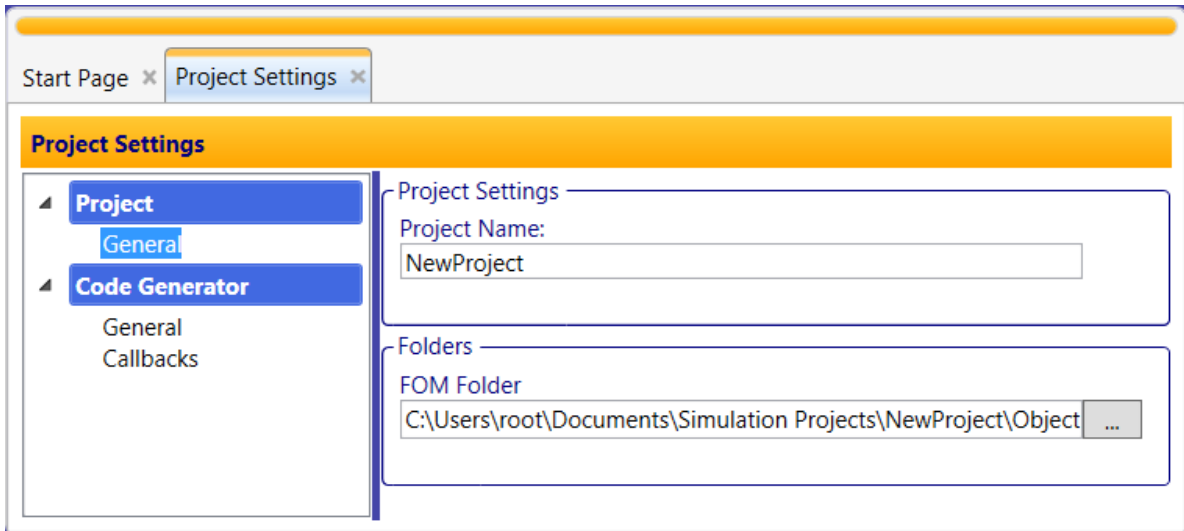


Figure 8. Project Settings - General

The following table summarizes the parameters.

Table 3. Code Generator General Settings

PARAMETER	EXPLANATION
Project Name	The user can rename the project name here. (1) The project name is the default namespace for the code generation. (2) The project name is the file name of the project when saved
Home Folder	This is the home folder where SimGe object model is saved. Not implemented yet.

3.8 Options

Options are application specific parameters such as "Show status bar".-

All the settings done by the user persist. Options are automatically loaded when the program is started and the changes are automatically kept when the program is closed. The options are kept in an XML file (i.e. Options.xml) located in SimGe installation folder.

The options contain only one group for user interface: The MRU options as seen in Figure 74.

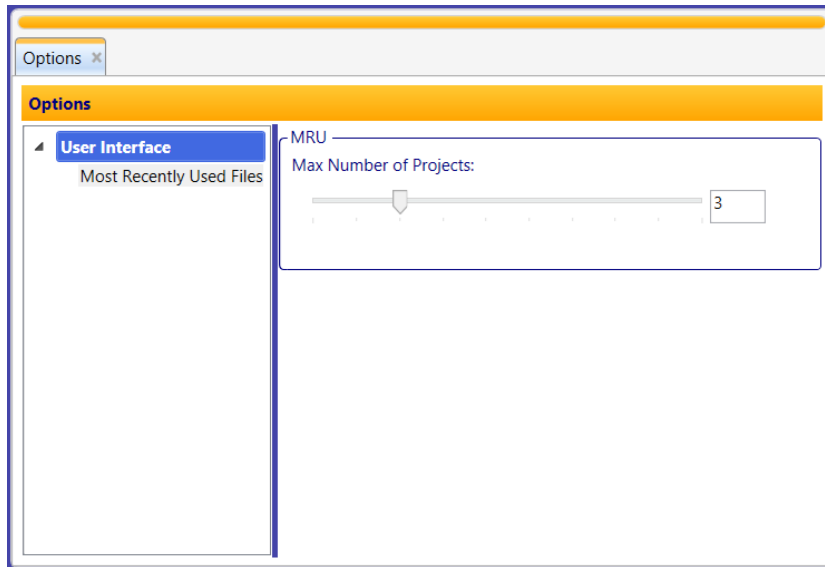


Figure 9. Options – User Interface

The following table summarizes the setting parameters.

Table 4. MRU Options

PARAMETER	EXPLANATION
Max Number of Projects	SimGe keeps track of the recently used projects. The user can adjust the max number of files to be kept. Default value is 5.

4. Object Model Construction

SimGe Object Model Editor (OME) is a fully-dressed HLA object model editor that enables to work with HLA13 and HLA1516-2010 object models. OME support to work with multiple federation object models¹ at the same time. SimGe object model can be saved as a separate file to increase its reusability. It can be imported by any SimGe project.

SimGe provides three options to begin construction of an object model (i.e. FOM and SOM) for the simulation project. The user can opt to construct an object model (OM) by:

- (1) Developing from scratch (i.e. by creating an empty object model),
- (2) Loading a previously saved SimGe object model, or
- (3) Importing either an existing FED file or Federation Document Data (FDD) file (for example, importing a generic FOM – RPR-FOM FED file).

The following subsections explain how to begin construction an object model using those options.

The object model of a SimGe simulation project is kept in an XML file with extension “.fom” in the *project FOM folder*, which is the folder named “ObjectModel” in the project home folder. The name of the file is the name of the project added by suffix “OM”.

When object model is loaded, SimGe allows the user editing the object model content in an OMT table view style.

The user can suppress or reveal the Management Object Model (MOM). When MOM is suppressed, MOM data is taken account neither in the view nor in code generation.

The user interface of OME always uses the local culture format of the operating system, where SimGe is installed at. That said, if the operating system culture information is set to a European country (e.g. Turkiye), the floating point values are separated by a comma as opposed to United States where the floating point format uses a dot. The culture format of the generated artifacts (e.g. FDD file) by SimGe conforms to their related specifications. For instance, the values for the update rates are always separated by a dot.

4.1 Loading an Existing SimGe Object Model

If the user wants to use a previously saved object model file, he/she can import the existing file by using the “Load Object Model” menu item. The imported OM file is copied to the project FOM folder. SimGe overwrites the file if a file with the same name exists in the project FOM folder.

SimGe object model is an internal representation specific to SimGe. It does not conform to HLA 1516-2010 OMT or Data Interchange Format (DIF) schemas.

¹ Federation object model states both Federation Object Model (FOM) and Simulation Object Model (SOM) through this chapter.

Please note that SimGe versions do not follow backward compatibility policy. New versions sometimes can't load an older object model.

4.2 Removing Object Model

The user can remove the current OM file and re-import another one. When the user removes the object model, all opened OMEs and their related federation object models will be discarded.

4.3 Creating a Federation Object Model from Scratch

After creating a new project, a user can either create an empty object model or import a SimGe object model, FED, or FDD file. Figure 10 depicts the options. When the user selects the "Create New FOM/SOM", an empty object model is created and initialized using HLA Standard Management and Initialization Module (MIM) file¹.

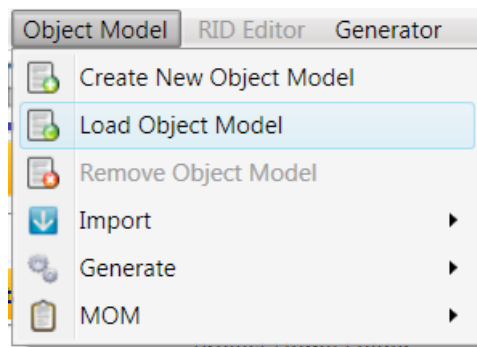
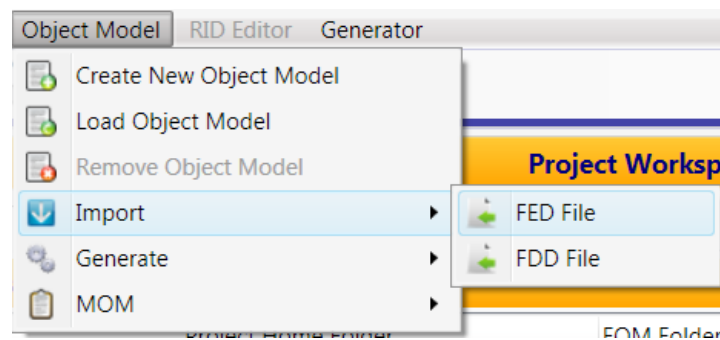


Figure 10. Object Model Create/Import

The user can create multiple federation object models (i.e. FOM and SOM). A separate OME will be dedicated to each federation object model for editing.

4.4 Importing a FED/FDD File

After creating a new project, the user can import an existing FED/FDD file by selecting Import FED/FDD menu item from object model menu or by pressing Import FED/FDD button in toolbar. The user can select the FED/FDD file by browsing folders on the open file dialog. The file extension is ".fed" for FED files and ".xml" for FDD files. Please note that SimGe does not copy the FED/FDD file to the project folder.



¹ <http://standards.ieee.org/downloads/1516/1516.1-2010/>, last accessed at July 8, 2012.

Figure 11. Import of a FED/FDD File

The FED file format must conform to the HLA FED file specification [6] and the FDD file must conform to the HLA1516-2010 FDD schema [7]. FDD importer validates the source FDD file using the HLA1516-2010 FDD schema and reports the validation results. The validation results show the inconsistencies found on the source import file. Although SimGe reports the warnings, it tries to import the source import file.

As an example, when you import the sample RestaurantFOMmodule.xml, you got two warnings as depicted in the following figure. The first error indicates that `HLAinteractionRoot` class has missing mandatory elements, which are transportation and order. The second error indicates that the `HLAreliable` transportation not found in the transportations. Although there are inconsistencies SimGe successfully imports and, if possible, corrects the errors.

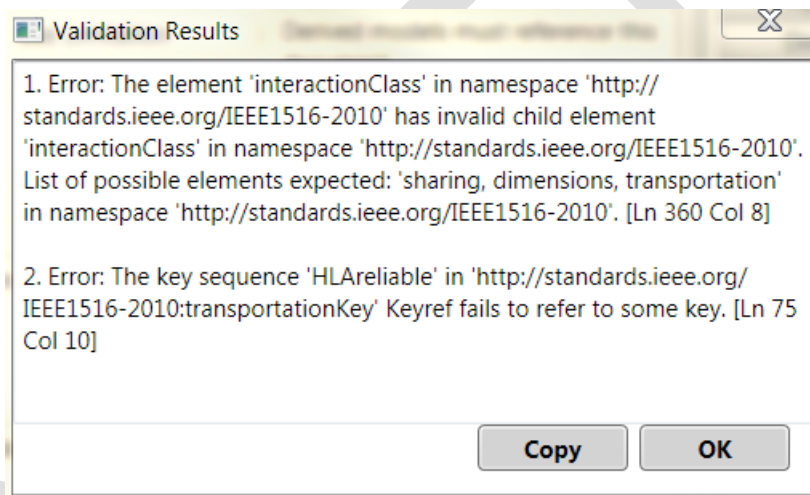


Figure 12. Import Validation Report

The user can copy the errors found to paste it to another program.

After importing a FED/FDD file, the user can edit the OM and the code generator is ready to generate SOM code for application federate.

4.4.1 Mappings

In HLA 1516, the P/S attribute in interaction class structure table, is considered in code generation. The transportation, ordering, and dimensions are added to the interaction class structure table instead of parameters table. Hence, those properties are related to the interaction rather than parameter.

In HLA 1.3 OMT Attribute table, the Transferable/Acceptable (T/A) column corresponds to Divest/Acquire (D/A) in SimGe attribute table. In the same way, Updateable/Reflectable (U/R) column is mapped to P/S. In interaction class structure table, the Capability column (i.e. Initiate/Respond/Sense) is matched to the Publish/Subscribe (P/S) attribute found in the OMT 1516 interaction structure table. P/S attribute is considered in code generation. Dimensions are presented in an additional table.

4.5 Exporting the FED/FDD Files

Export of FED/FDD file is also selected using generate submenu of the object model menu. Using export command from the file menu displays the following user screen. As SimGe supports multiple object models, export utility asks for the source object model and target formats.

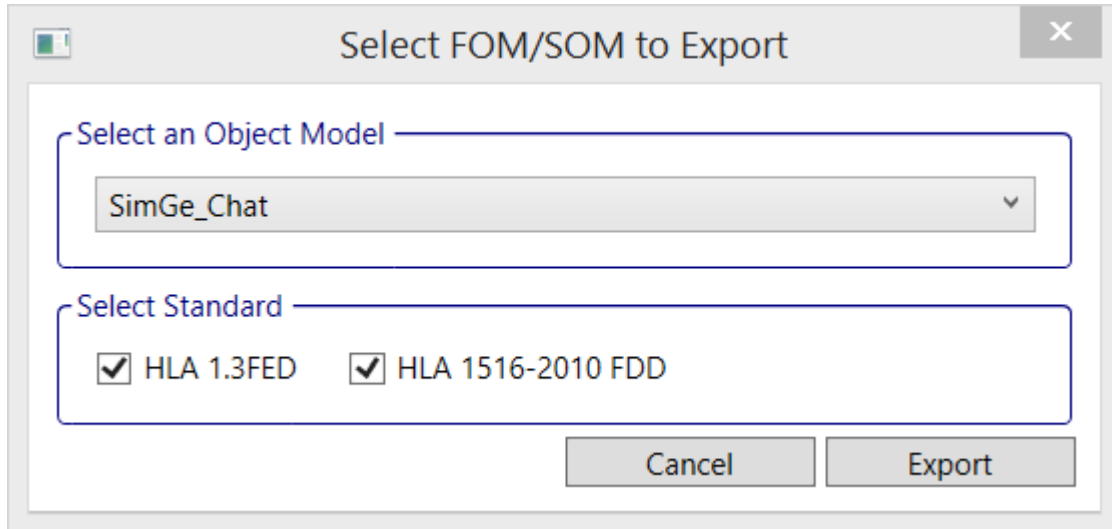


Figure 13. Export Configuration

Each object model also can be exported by using their OME toolbar. See Section 4.6.1.

FDD exporter validates the generated FDD file and reports the result to the user. See Figure 14.

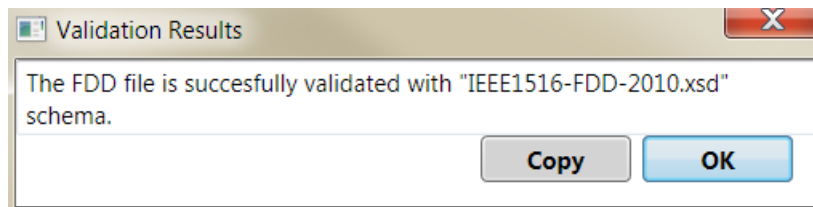


Figure 14. Validation Results

4.6 Object Model Editor

OME supports multiple object models. User can work with multiple versions of FOM at the same time. The user can add many FOMs to the project. Each FOM can be worked in its own OME.

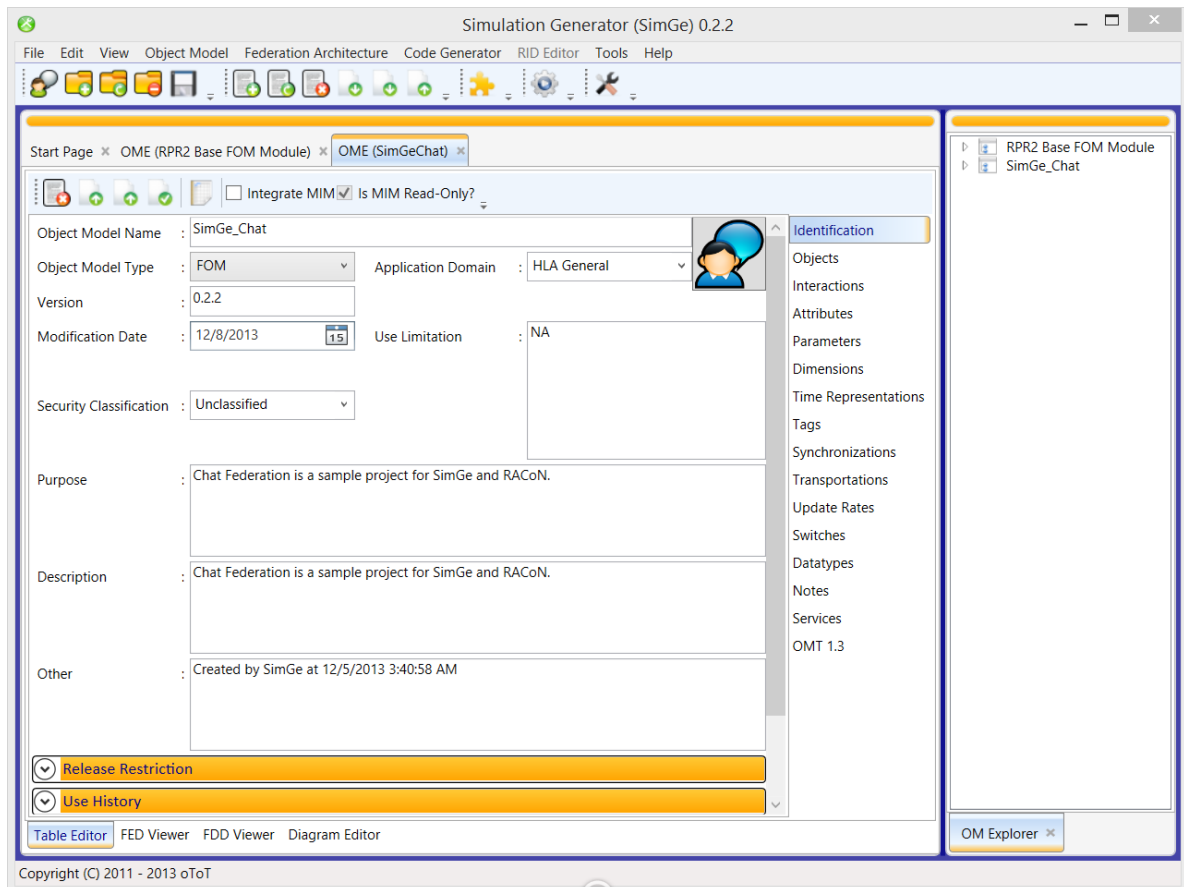


Figure 15. Multiple OMEs

4.6.1 OME Toolbar

Each OME has its own toolbar. Currently, the toolbar commands: removing current federation object model, Export to FED/FDD, validating of FDD, MIM integration, OMT report generation, MIM integration, and setting MIM as read-only are supported. All those commands are specific to the object model opened in that OME.

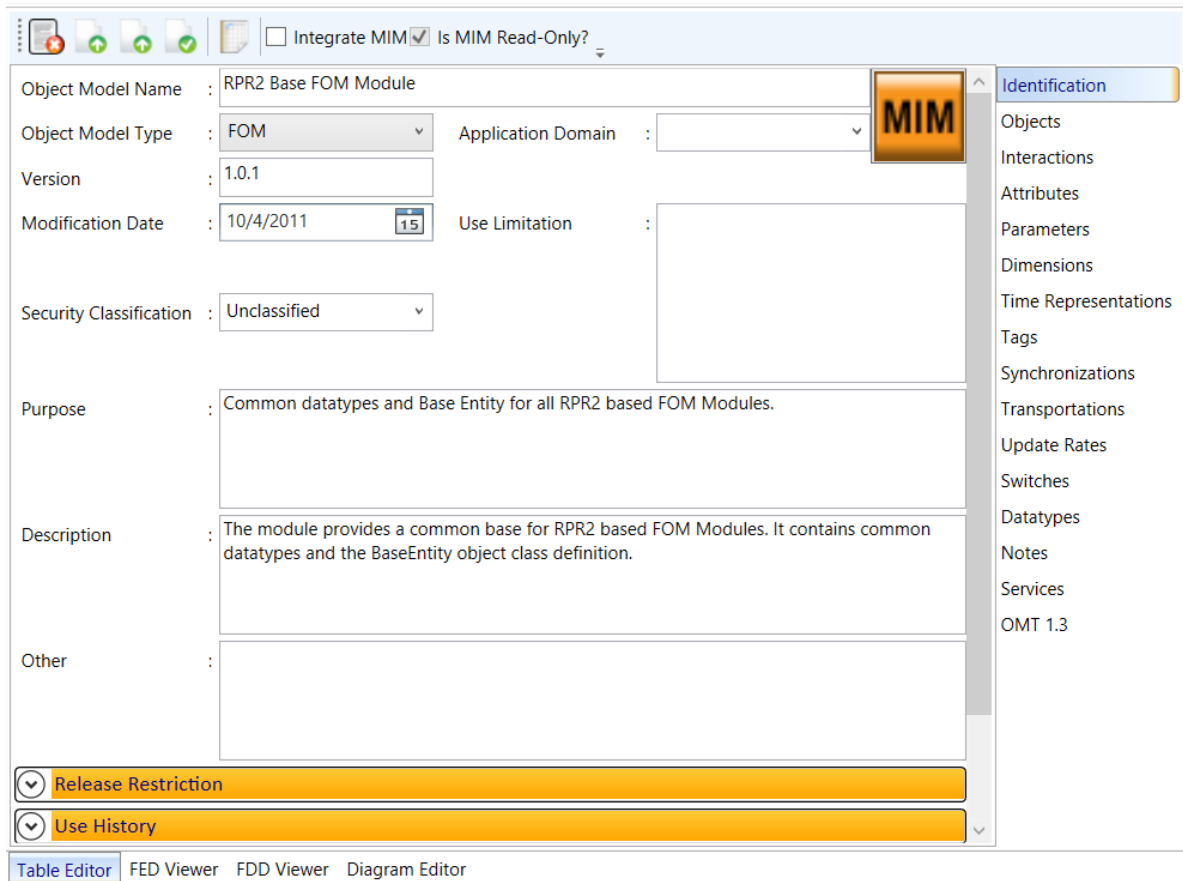


Figure 16. OME Toolbar

4.6.1.1 Removing the Current Federation Object Model

The first button on the OME toolbar is used to remove the current federation object model from the project. Please note that this is an irreversible action.

Another way to remove the current object model is to use the commands on OM explorer, see Section 4.11.

4.7 Table Editor

The Table editor presents a working area to edit the Object Model Template (OMT) tables. It allows the user to do well-known table operations- create, read, update, delete operations, called CRUD. The Table editor is composed of all the OMT tables found in the HLA 1516.2010 OMT specification tables [2]. Moreover, it also supports the HLA OMT 1.3 specification tables [8] for routing spaces. A screen shot of the GUI is depicted in Figure 26.

Some rows in OMT table are read-only rows and they are specified with a lock symbol in the header of the row and the row is grayed. The read-only rows are the compulsory built-in OMT elements (e.g. ObjectRoot, RTIprivate) or the MOM elements (e.g. Manager, Federation, Federate). The white rows are eligible for CRUD operations.

	Name	Size in bits	Interpretation	Endian	Encoding
	HLAinteger16BE	16	Integer in the range [-2 ¹⁵ , 2 ¹⁵ - 1]	Big	16-bit two's complement
	HLAinteger32BE	32	Integer in the range [-2 ³¹ , 2 ³¹ - 1]	Big	32-bit two's complement

READ-ONLY

Figure 17. Read-only OMT Elements

The user can create a new OMT Element by clicking the “add new item” button at the end of the each table (see Figure 18).

Parameter Table

	Parameter	Interaction	Data Type
	TimlinessOk	MainCourseServed	HLAboolean
	AccuracyOk	MainCourseServed	ServiceStat
	TemperatureOk	MainCourseServed	ServiceStat

REMOVE

Add New Item
 CREATE

Figure 18. Adding/Removing an OMT Element

A data input wizard for each element is popped up as seen in Figure 27. All data input is validated according to [2]. In case of a validation error, the user is warned about the error possible cause (e.g. “this field cannot be empty”).

Create New OMT Element Wizard

Hints:

1. Naming must conform to HLA1516.2-2010 OMT specification.

Object Class

Class Name :

This field cannot be empty.

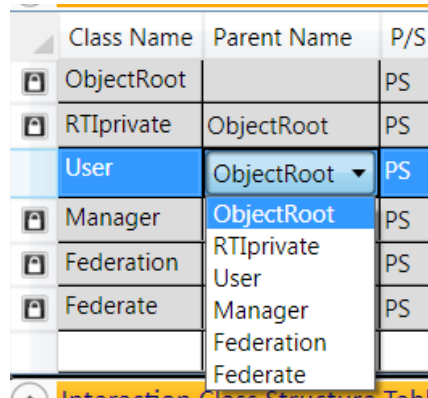
Parent Name :

P/S :

Figure 19. Object Class Data Input Screen

This wizard is only used for the creation of an OMT element. To modify an existing one, double click the cell where modification is required. In order to delete an OMT element, simply select the row and then click the remove icon (found in each row) or press delete key on keyboard.

Mostly, the columns in an OMT table provide the suitable value that the user can select. For example, when setting the `Parent` of an object class, only the object classes defined in the OMT is shown to the user.



Class Name	Parent Name	P/S
ObjectRoot		PS
RTIprivate	ObjectRoot	PS
User	ObjectRoot	PS
Manager	ObjectRoot	PS
Federation	RTIprivate	PS
Federate	Manager	PS
	Federation	
	Federate	

Figure 20. Value Selection

Whenever the user saves the project, the object model (without MOM elements) is also saved.

For the semantics of tables and columns, refer to [8] [9].

4.7.1 Object Model Identification

OM identification interface consists of an area for object model metadata and explorers for point of contacts (POCs), use history, keywords, and references of the object model. A screen shot of the GUI is depicted in Figure 21.

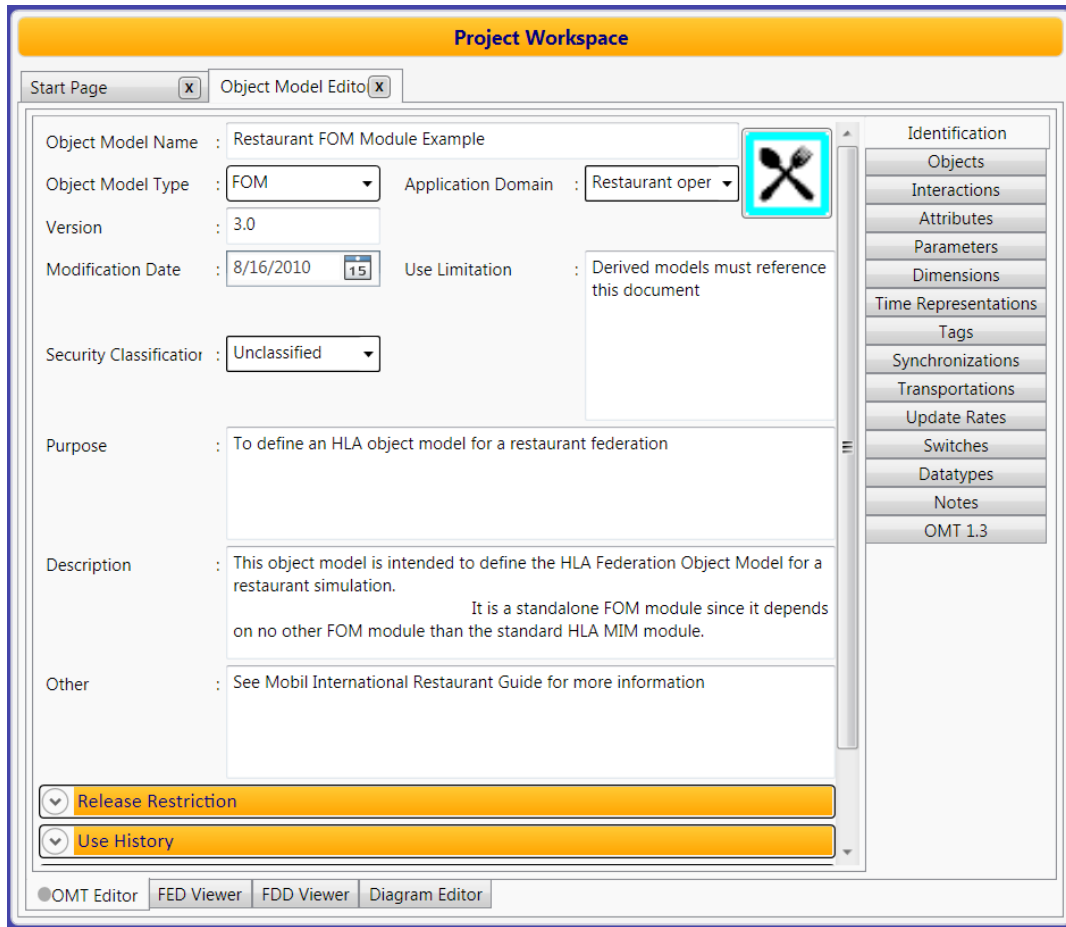


Figure 21. Table editor

4.7.1.1 POC

The POCs explorer displays all the POCs in a table view. When the user clicks a user row, additional details about the user showing the phone numbers and e-mails are displayed (see Figure 22). The user can add, delete, or modify. The predefined POC types are provided as the user can define new ones.

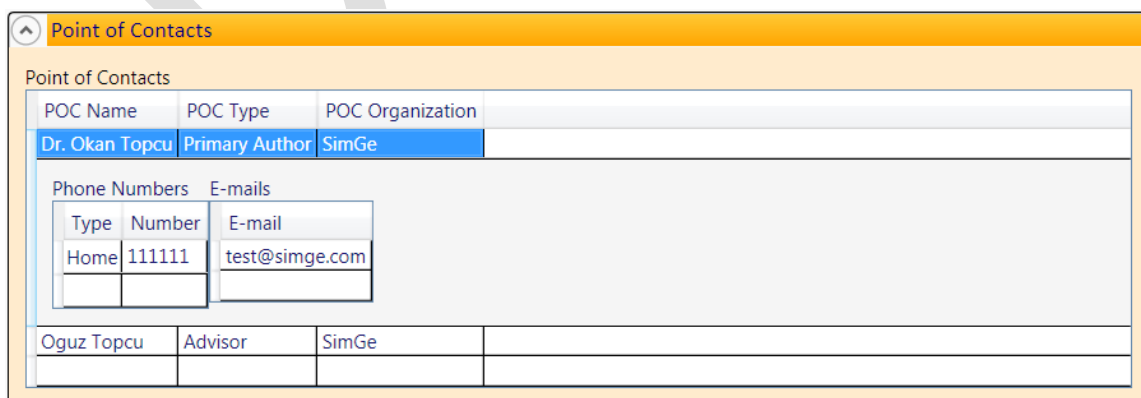
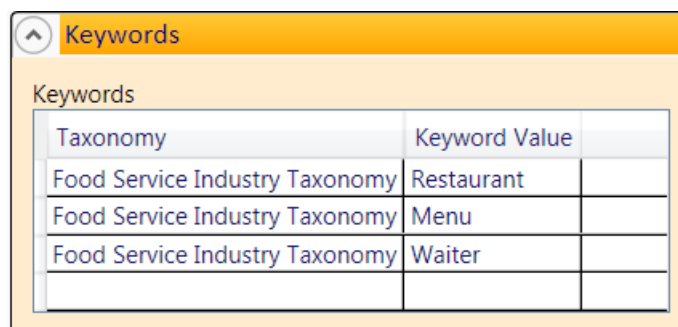


Figure 22. POC Details

4.7.1.2 Keywords

The keywords table displays the taxonomy and keyword value (see Figure 23). The user can add, delete, or modify.

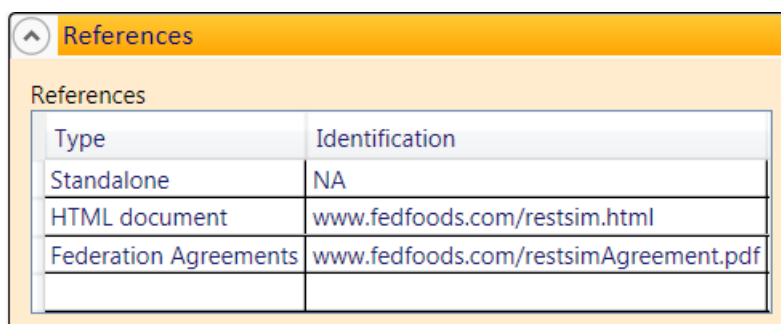


Taxonomy	Keyword Value
Food Service Industry Taxonomy	Restaurant
Food Service Industry Taxonomy	Menu
Food Service Industry Taxonomy	Waiter

Figure 23. Keywords

4.7.1.3 References

The references table displays the reference type and identification (Figure 24). The user can add, delete, or modify.

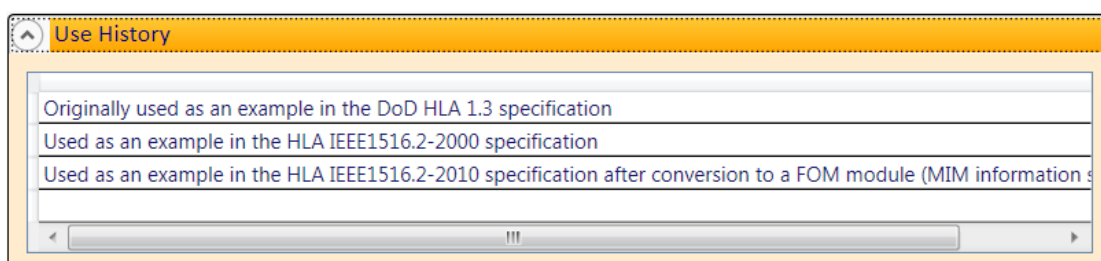


Type	Identification
Standalone	NA
HTML document	www.fedfoods.com/restsim.html
Federation Agreements	www.fedfoods.com/restsimAgreement.pdf

Figure 24. References

4.7.1.4 Use History

The use history table displays the history of the object model (see Figure 25). The user can add, delete, or modify.



Originally used as an example in the DoD HLA 1.3 specification
Used as an example in the HLA IEEE1516.2-2000 specification
Used as an example in the HLA IEEE1516.2-2010 specification after conversion to a FOM module (MIM information s

Figure 25. Use History

4.7.1.5 Glyph

SimGe fully supports glyph import and export from an FDD file. At the top-right, the user also can set a new glyph for the object model by clicking the image area. The glyph attributes; height, width, and type is automatically set.

4.7.2 Objects

The objects tab is used to edit HLA object classes. A screenshot is presented in Figure 26.

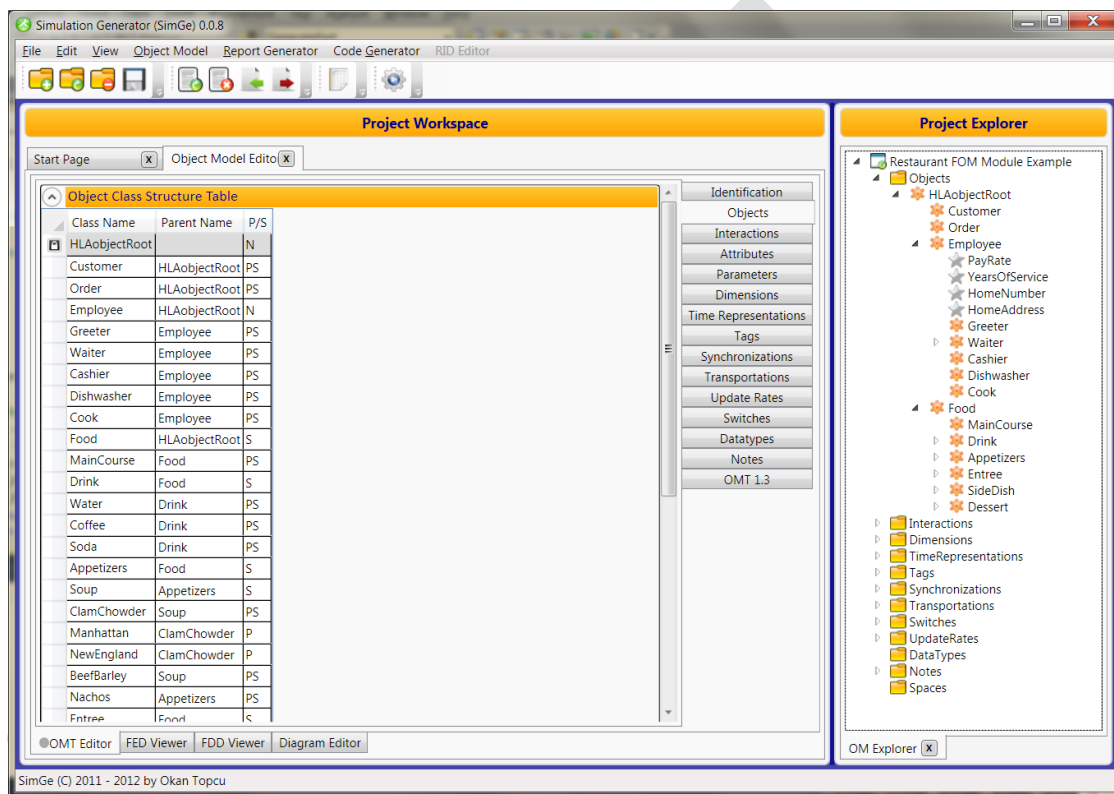


Figure 26. Objects

The class hierarchical relationships (class-subclass relation) among the HLA classes in the object class structure table is defined differently from the OMT specification counterparts. In SimGe, the hierarchical relationships are defined by simply specifying the parent of each class, whereas in the OMT specifications, the hierarchical relationships are defined by column order.

The user can create a new object class by clicking add-new-item button at the end of the each table. The parent of the newly created class must always be set (if it is a root class, then set parent as the HLAobjectRoot). The publish and subscribe status is set to Neither by default, the user can change it according to his/her needs. Data input screen is seen in Figure 27.

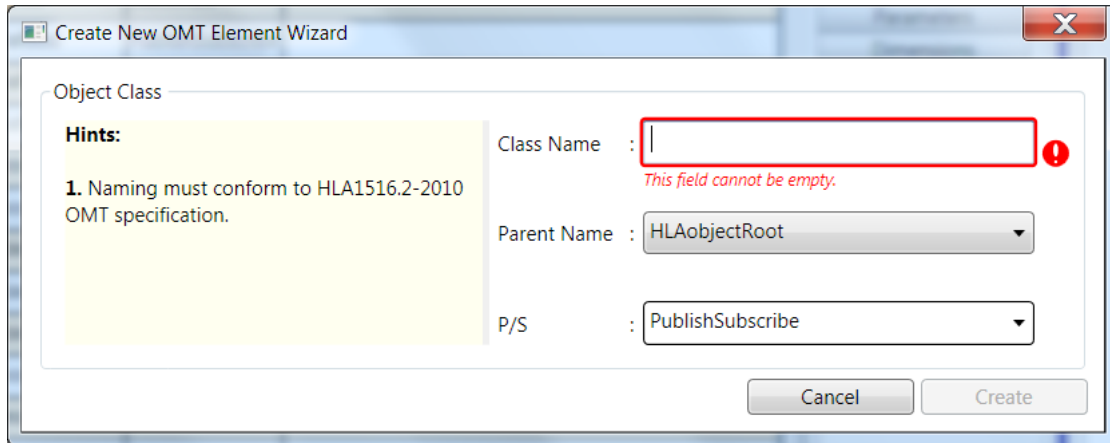


Figure 27. Object Class Data Input Screen

4.7.3 Interactions

The interactions tab is used to create/edit/delete HLA interaction classes. A screenshot is presented in Figure 28. The usage is the same for objects table.

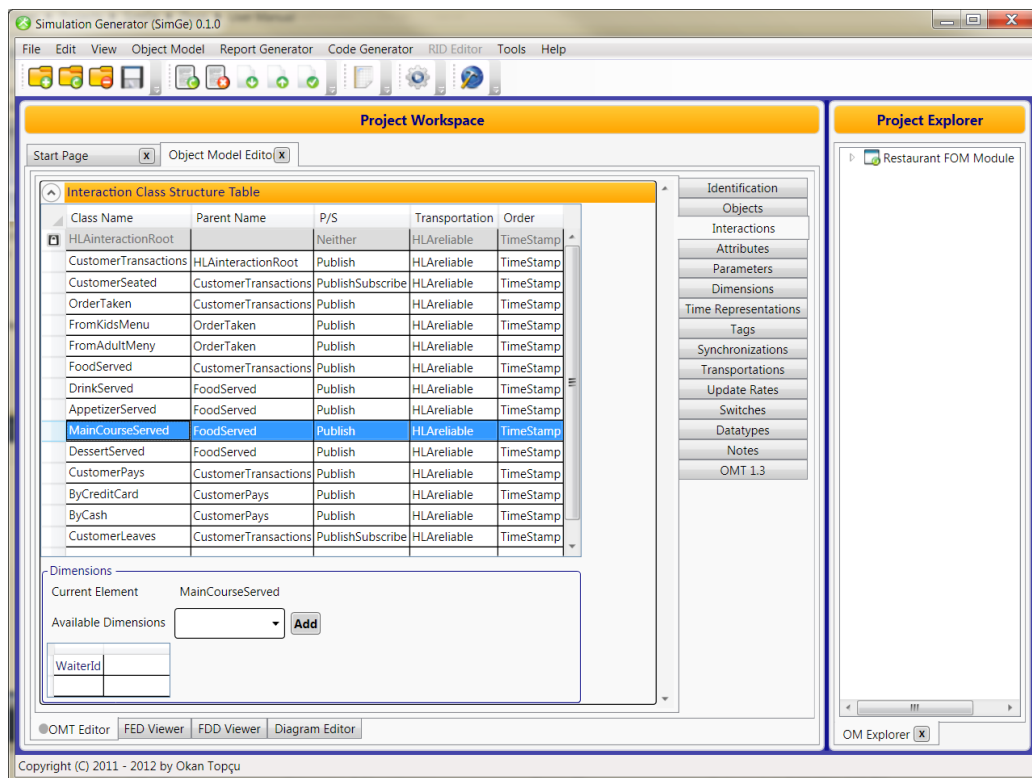


Figure 28. Interactions

The class hierarchical relationships (class-subclass relation) among the HLA classes in the interaction class structure table is defined differently from the OMT specification counterparts. In SimGe, the hierarchical relationships are defined by simply specifying the parent of each class, whereas in the OMT specifications, the hierarchical relationships are defined by column order.

Data input screen is seen in Figure 27.

Figure 29. Interaction Class Data Input Screen

Each attribute and interaction class has a dimension set. In the table view, the dimensions are provided in a detail table when the related row of attribute or interaction is selected as seen in Figure 30. The user can add dimensions to the selected attribute/class from the available dimensions presented in a drop down menu. You cannot add if a dimension is already added to the dimension list of the interaction. If you want to remove a dimension, then just select the dimension you want to delete in the table and then press delete key. The dimensions of attributes and interactions can also be added to the dimension set while creating the attribute/interaction class element using the data input screen as seen in Figure 29.

4.7.4 Attributes

The attributes tab is used to edit HLA object attributes. A screenshot is presented in Figure 30. Dimensions for the selected attribute are provided in the detail table where the user can add or remove a dimension.

Attribute Table									
Attribute	Object	Data Type	Update Type	Update Condition	D/A	P/S	Data Type	Transportation	Order
HLAprivilegeToDeleteObject	HLAobjectRoot	HLAtoken	Static	NA	DivestAcquire	PublishSubscribe	HLAtoken	HLAreliable	TimeStam
HomeAddress	Employee	AddressType	Conditional	Employee request	DivestAcquire	PublishSubscribe	AddressType	HLAreliable	TimeStam
HomeNumber	Employee	HLAASCIIstring	Conditional	Employee request	DivestAcquire	PublishSubscribe	HLAASCIIstring	HLAreliable	TimeStam
YearsOfService	Employee	Years	Periodic	1/year	DivestAcquire	PublishSubscribe	Years	HLAreliable	TimeStam
PayRate	Employee	DollarRate	Conditional	Merit increase	DivestAcquire	PublishSubscribe	DollarRate	HLAreliable	TimeStam
State	Waiter	WaiterTasks	Conditional	Work flow	DivestAcquire	PublishSubscribe	WaiterTasks	HLAreliable	TimeStam
Cheerfulness	Waiter	WaiterValue	Conditional	Performance review	DivestAcquire	PublishSubscribe	WaiterValue	HLAreliable	TimeStam
Efficiency	Waiter	WaiterValue	Conditional	Performance review	DivestAcquire	PublishSubscribe	WaiterValue	HLAreliable	TimeStam
NumberCups	Drink	DrinkCount	Conditional	Customer request	NoTransfer	PublishSubscribe	DrinkCount	HLAreliable	TimeStam
Flavor	Soda	FlavorType	Conditional	Customer request	NoTransfer	PublishSubscribe	FlavorType	HLAreliable	TimeStam

Dimensions

Current Element Flavor

Available Dimensions SodaFlavor **Add**

SodaFlavor	
BarQuantity	

Figure 30. Attributes

Data input screen is seen in Figure 27.

Create New OMT Element Wizard

Attribute

Hints:
 1. Naming must conform to HLA1516.2-2010 OMT specification.

Attribute : !
This field cannot be empty.

Object : HLAobjectRoot

Data Type :

Update Type : Static

Update Condition :

Ownership : NoTransfer

P/S : PublishSubscribe

Transportation : !
This field cannot be empty.

Order : Receive

Dimensions

Available Dimensions: WaiterId

Dimensions Related To This OMT Element	
SodaFlavor	
BarQuantity	
WaiterId	

Figure 31. Attribute Data Input Screen

4.7.5 Parameters

The parameters tab is used to edit HLA interaction parameters. A screenshot is presented in Figure 32.

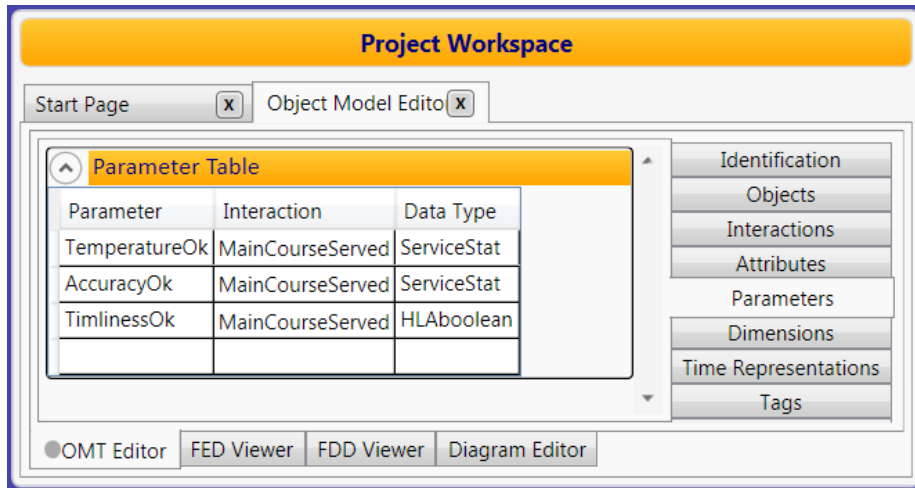


Figure 32. Parameters

Data input screen is seen in Figure 27.

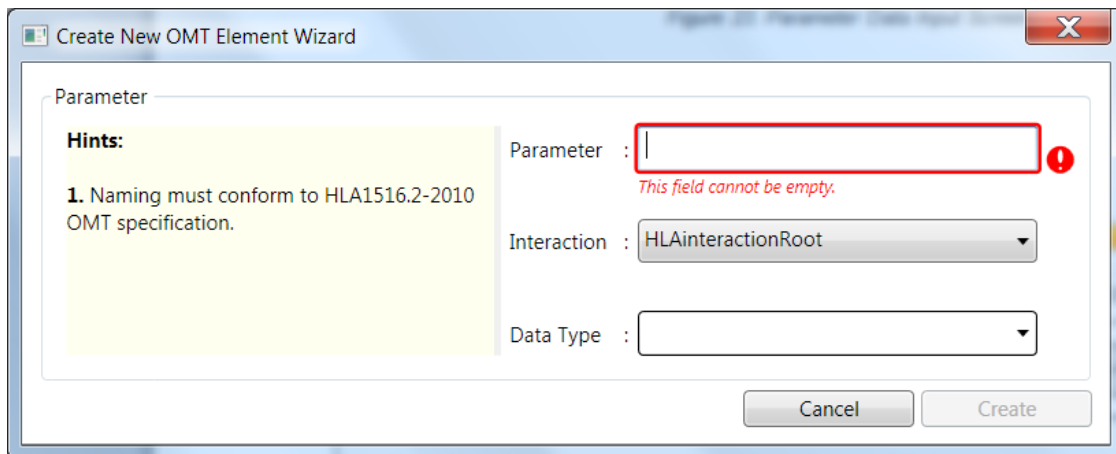


Figure 33. Parameter Data Input Screen

4.7.6 Dimensions

The dimensions tab is used to create and edit dimensions. See hints in Figure 34.

Name	Data Type	Dimension Upper Bound	Normalization Function	Value When Unspecified
SodaFlavor	FlavorType	3	linearEnumerated (Flavor, [Cola, Orange, RootBeer])	[0..3]
BarQuantity	DrinkCount	25	linear (NumberCups, 1, 25)	{0}
WaiterId	EmplId	20	linear (WaiterId, 1, 20)	Excluded
HLAfederate	HLAnormalizedFederateHandle	0	Normalize Federate Handle service	Excluded
HLAServiceGroup	HLAnormalizedServiceGroup	7	Normalize Service Group service	Excluded

Hints:

- Upper Bound value must be a positive integer greater than zero. A zero value shows an N/A value.
- The last column (value when unspecified) specifies a default range that the value can be a nonnegative integer (can be zero or higher), a subrange of [0, dimension upper bound), or Excluded value.

Figure 34. Dimensions

Data input screen is seen in Figure 27.

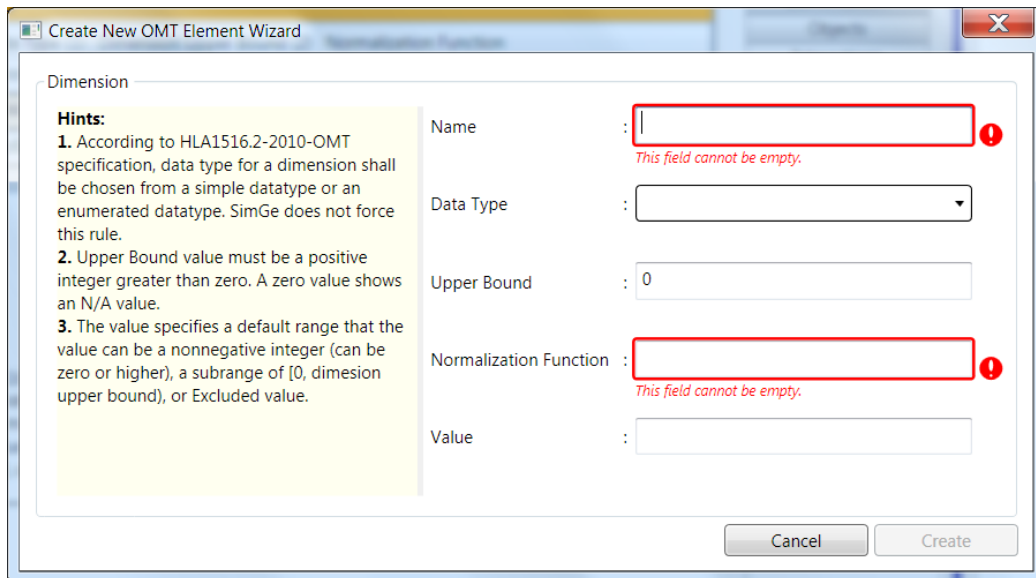


Figure 35. Dimension Data Input Screen

4.7.7 Time Representations

The time representations table provides two time representation categories (i.e. timestamp and lookahead) by default. The user can not add new ones or delete the existing ones. Furthermore, only the data type and semantics cells of two time representations are allowed for modification.

Time Representation Table		
Category	Datatype	Semantics
timeStamp		
lookahead		

Figure 36. Time Representations

4.7.8 User-supplied Tags

The user-supplied tags table provides all default tag categories. The user can not add new ones or delete the existing ones. Furthermore, only the data type and semantics cells of the present ones are allowed for modification.

User-supplied Tag Table		
Category	Datatype	Semantics
updateReflectTag		
sendReceiveTag		
deleteRemoveTag	HLAASCIIstring	Reason for deletion
divestitureRequestTag		
divestitureCompletionTag		
acquisitionRequestTag		
requestUpdateTag		

Figure 37. User-supplied Tags

4.7.9 Synchronizations

The synchronizations tab is used to edit synchronizations. You can add, modify, or delete synchronization.

Synchronization Table			
Label	Tag Datatype	Capability	Semantics
InitialPublish		Achieve	Achieved when all classes are published and subscribed, and all initially present objects are registered
InitialUpdate		Achieve	Achieved when instance attribute values for all initially present objects are updated
BeginTimeAdvance		Achieve	Achieved when time management services are invoked
PauseExecution	TimeType	RegisterAchieve	Achieved when the time advance after the time in the user-supplied tag is attained; time advance requests should then cease

Figure 38. Synchronizations

Data input screen is seen in Figure 27.

Figure 39. Synchronization Data Input Screen

4.7.10 Transportations

Two transportations are added by default. The user cannot delete or modify those. The user is allowed to add new transportations.

Transportations Table		
Name	Reliability	Semantics
LowLatency	BestEffort	Choose the delivery mechanism that results in the lowest latency from service initiation to callback invocation at the receiving federate
HLAreliable	Reliable	Provide reliable delivery of data in the sense that TCP/IP delivers its data reliably
HLAbestEffort	BestEffort	Make an effort to deliver data in the sense that UDP provides best-effort delivery

Figure 40. Transportations

Data input screen is seen in Figure 27.

Transportation

Hints:

1. Naming must conform to HLA1516.2-2010 OMT specification.

Name : !
This field cannot be empty.

Reliability :

Semantics :

Cancel Create

Figure 41. Transportation Data Input Screen

4.7.11 Update Rates

The update rates tab is used to edit update rates.

Data input screen is seen in Figure 27.

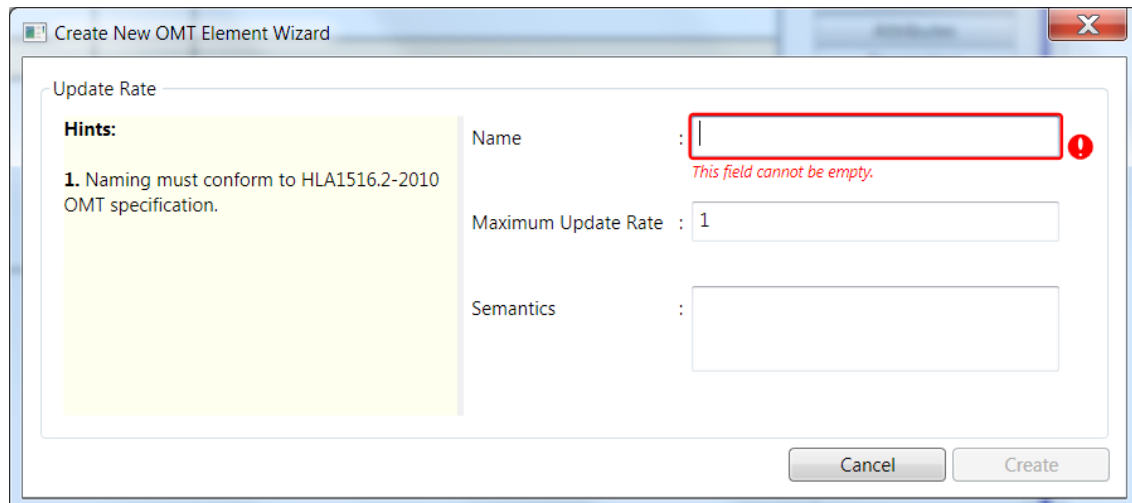


Figure 42. Update Rate Data Input Screen

4.7.12 Switches

The switches table provides all default switches categorized as either federation or federate basis. The user can set a switch by clicking the checkbox. The user can select the desired value for Automatic Resign Action switch using a dropdown menu.

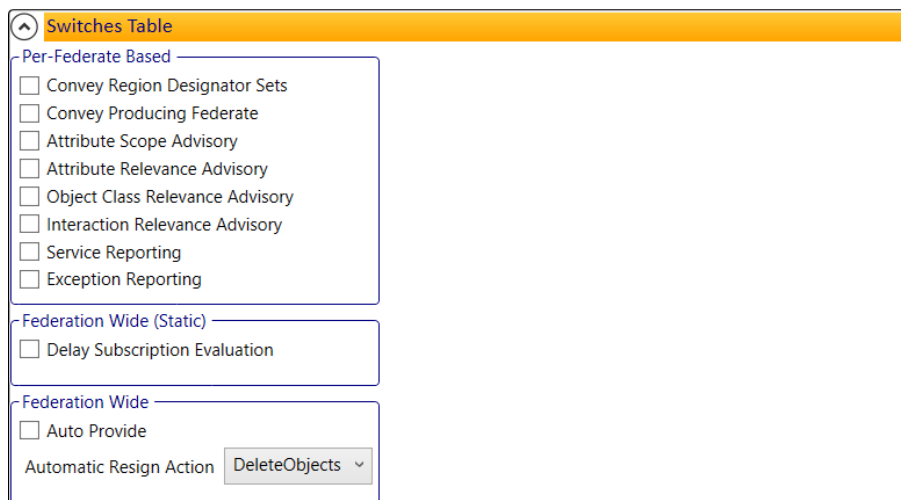


Figure 43. Switches

4.7.13 Data Types

SimGe OME supports all data type tables and data representations.

4.7.13.1 Basic Data Representations

SimGe OME loads all the predefined data representations and does not let the user change them. The user can create or remove new ones.

Basic Data Representation Table				
Name	Size in bits	Interpretation	Endian	Encoding
<input type="checkbox"/> HLAinteger16BE	16	Integer in the range $[-2^{15}, 2^{15} - 1]$	Big	16-bit two's complement sign
<input type="checkbox"/> HLAinteger32BE	32	Integer in the range $[-2^{31}, 2^{31} - 1]$	Big	32-bit two's complement sign
<input type="checkbox"/> HLAinteger64BE	64	Integer in the range $[-2^{63}, 2^{63} - 1]$	Big	64-bit two's complement sign
<input type="checkbox"/> HLAfloat32BE	32	Single-precision floating point number	Big	32-bit IEEE normalized single-
<input type="checkbox"/> HLAfloat64BE	64	Double-precision floating point number	Big	64-bit IEEE normalized double
<input type="checkbox"/> HLAoctetPairBE	16	16-bit value	Big	Assumed to be portable amor
<input type="checkbox"/> HLAinteger16LE	16	Integer in the range $[-2^{15}, 2^{15} - 1]$	Little	16-bit two's complement sign
<input type="checkbox"/> HLAinteger32LE	32	Integer in the range $[-2^{31}, 2^{31} - 1]$	Little	32-bit two's complement sign
<input type="checkbox"/> HLAinteger64LE	64	Integer in the range $[-2^{63}, 2^{63} - 1]$	Little	64-bit two's complement sign
<input type="checkbox"/> HLAfloat32LE	32	Single-precision floating point number	Little	32-bit IEEE normalized single-
<input type="checkbox"/> HLAfloat64LE	64	Double-precision floating point number	Little	64-bit IEEE normalized double
<input type="checkbox"/> HLAoctetPairLE	16	16-bit value	Little	Assumed to be portable amor
<input type="checkbox"/> HLAoctet	8	8-bit value	Big	Assumed to be portable amor
UnsignedShort	16	Integer in the range $[0, 2^{16} - 1]$	Big	16-bit unsigned integer.

Figure 44. Basic Data Representations

Data input screen is seen in Figure 27.

Create New OMT Element Wizard

Basic Data Representation

Hints:

1. Naming must conform to HLA1516.2-2010 OMT specification.

Name :

Size in Bits :

Interpretation :

Endian :

Encoding :

!
This field cannot be empty.

Figure 45. Data Representation Data Input Screen

4.7.13.2 Simple Data Types

The representations can be selected from the basic data representations table. SimGe OME automatically provides the available representations in a dropdown list. Whenever the user changes a data representation in the data representations table, the change is automatically applied here.

Simple Datatype Table					
Name	Representation	Units	Resolution	Accuracy	Semantics
HLAASCIIchar	HLAoctet	NA	NA	NA	Standard ASCII character (see ANSI Std x
HLAunicodeChar	HLAoctetPairBE	NA	NA	NA	Unicode UTF-16 character (see The Unic
HLAbyte	HLAoctet	NA	NA	NA	Uninterpreted 8-bit byte
HLAinteger64Time	HLAinteger64BE	NA	1	NA	Standardized 64 bit integer time
HLAfloat64Time	HLAfloat64BE	NA	4.9E-308	NA	Standardized 64 bit float time
TimeType	HLAfloat32BE	Minutes	0.01667	NA	Time representation
LAtype	HLAfloat32BE	Minutes	0.01667	NA	Time interval (non-negative)
DollarRate	HLAfloat32BE	\$/hour	0.01	Perfect	Cost per hour
Years	HLAinteger32BE	Years	1	Perfect	Elapsed time in years
DrinkCount	UnsignedShort	Cups	1	Perfect	Measure of number of drinks
EmpId	HLAinteger32BE	NA	1	Perfect	Employee identifier
RateScale	HLAinteger32BE	NA	1	Perfect	Evaluation of staff where 1 = best

Figure 46. Simple Data Types

Data input screen is seen in Figure 27.

Simple Datatype

Hints:

1. Naming must conform to HLA1516.2-2010 OMT specification.

Name :

Representation :

Units :

Resolution :

Accuracy :

Semantics :

Cancel Create

Figure 47. Simple Datatype Data Input Screen

4.7.13.3 Enumerated Data Types

When the user clicks a row in the enumerated data type table, then enumerations related to that data type is shown as a nested table. The user can create new enumerations by entering the last row in the table.

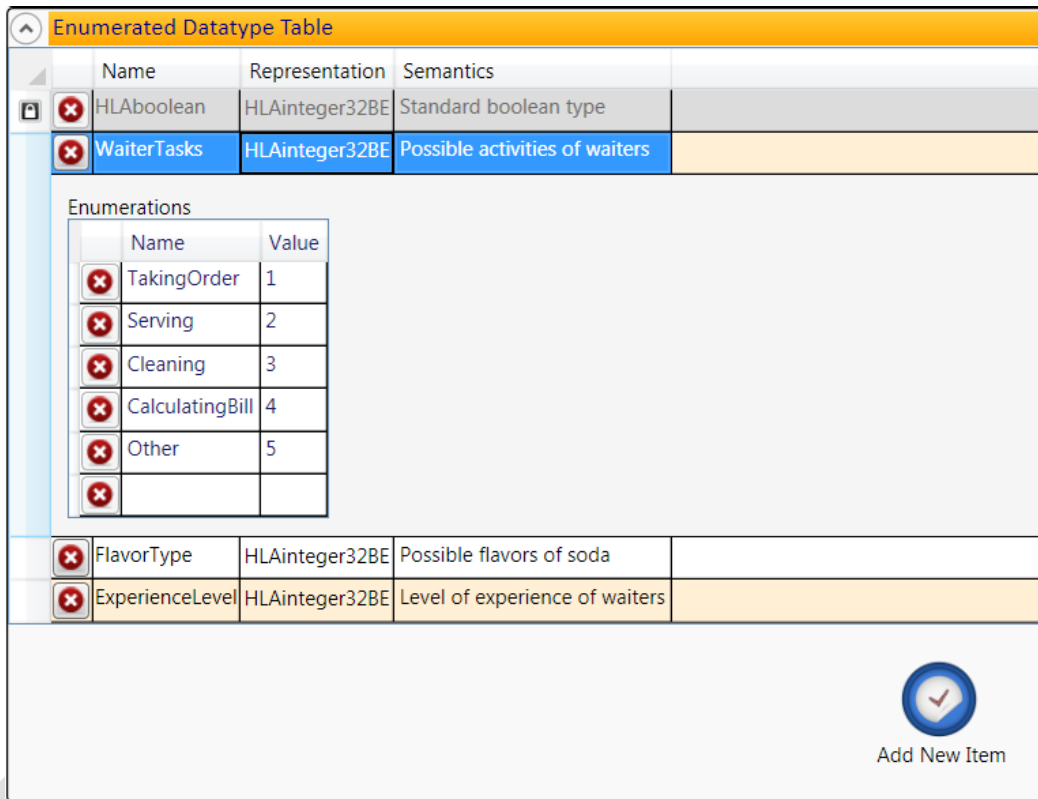


Figure 48. Enumerated Data Types

Data input screen for creating an enumerated datatype is seen in Figure 27.

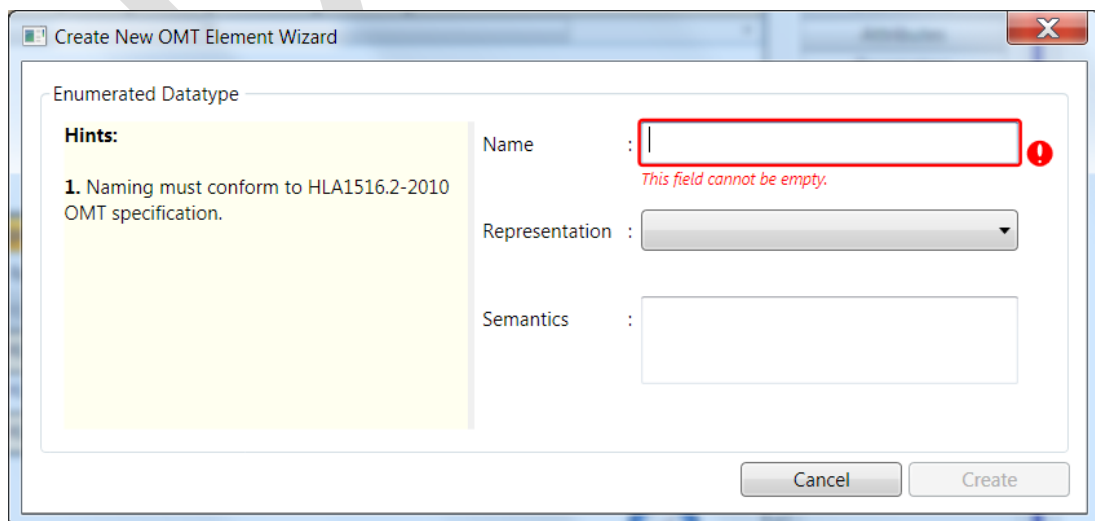


Figure 49. Enumerated Datatype Data Input Screen

4.7.13.4 Array Data Types

Element type is linked to the other types defined.

Array Datatype Table			
Name	Element Type	Cardinality	Encoding
<input checked="" type="checkbox"/> HLAASCIIstring	HLAASCIIchar	Dynamic	HLAvariableArray
<input checked="" type="checkbox"/> HLAunicodeString	HLAunicodeChar	Dynamic	HLAvariableArray
<input checked="" type="checkbox"/> HLAopaqueData	HLAbyte	Dynamic	HLAvariableArray
<input checked="" type="checkbox"/> HLAtoken	HLAbyte	0	HLAfixedArray
Employees	EmplId	10	HLAfixedArray
AddressBook	AddressType	Dynamic	An HLAinteger32BE followed by a set of index-value tuples. E

Figure 50. Array Data Types

Data input screen for creating an array datatype is seen in Figure 27.

Array Datatype

Hints:

1. Naming must conform to HLA1516.2-2010 OMT specification.

Name : !
This field cannot be empty.

Element Type :

Units :

Cardinality :

Encoding :

Semantics :

Cancel Create

Figure 51. Array Datatype Data Input Screen

4.7.13.5 Fixed Record Data Types

Encoding column shows the predefined encodings for fixed record data type. The user can add new encodings by editing this field.

When the user selects a fixed record data type in the table, then a detailed table is shown in order to depict the record fields. The user can choose the data type for the record field.

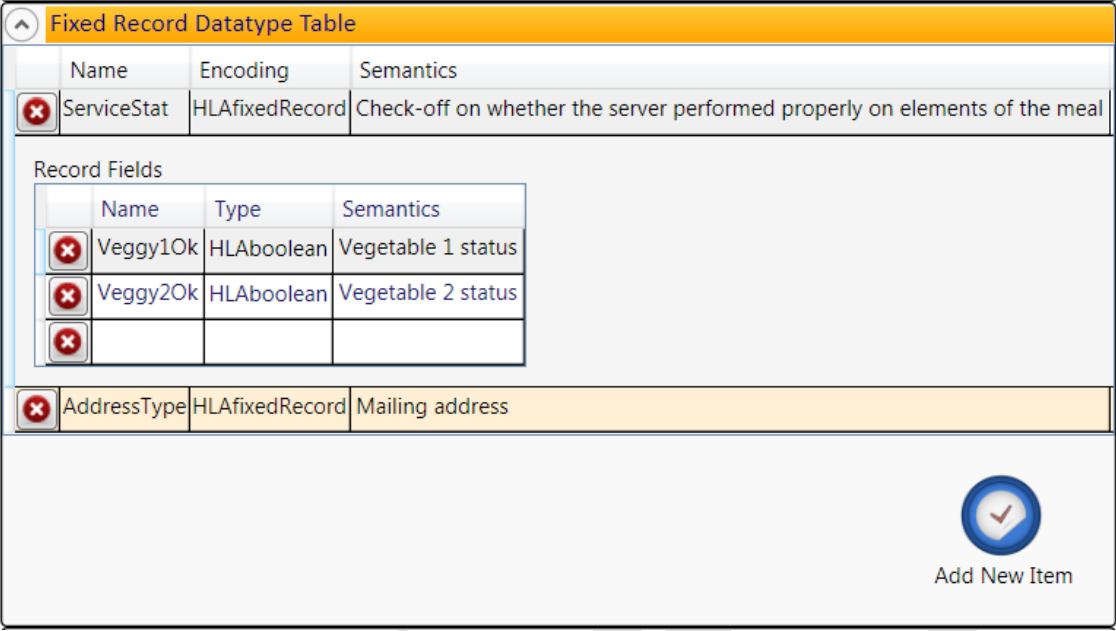


Figure 52. Fixed Record Data Types

Data input screen for creating a fixed array datatype is seen in Figure 27.

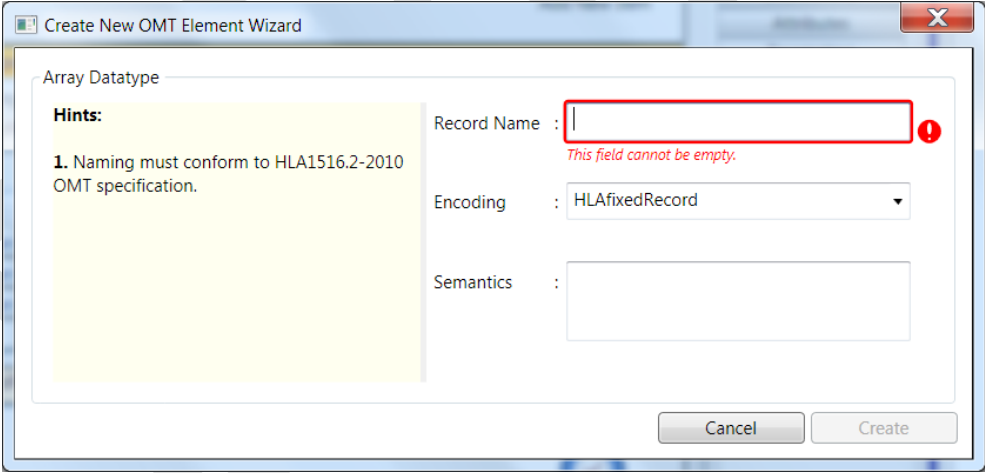



Figure 53. Fixed Record Datatype Data Input Screen

4.7.13.6 Variant Record Data Types

The data type of a variant record can be only an enumerated data type defined in the enumerated data type table.

Variant Record Datatype Table				
Name	Discriminant	Type	Encoding	Semantics
✘ WaiterValue	ValIndex	ExperienceLevel	HLAvariantRecord	Datatype for waiter performance rating value
Alternatives				
Enumerator	Name	Type	Semantics	
✘ Trainee	CoursePassed	HLAboolean	Ratings scale for employees under training	
✘ [Apprentice .. Senior], Master	Rating	RateScale	Ratings scale for permanent employees	
✘ HLAother	NA		All others	
✘				



Add New Item

Figure 54. Variant Record Data Types

Data input screen for creating a variant array datatype is seen in Figure 27.

Create New OMT Element Wizard

Array Datatype

Hints:

1. Record name and discriminant name must conform to HLA1516.2-2010 OMT specification.

Record Name	:	<input style="border: 1px solid red;" type="text"/>	!
		This field cannot be empty.	
Discriminant Name	:	<input style="border: 1px solid red;" type="text"/>	!
		This field cannot be empty.	
Data Type	:	<input type="text" value=""/>	
Encoding	:	<input type="text" value="HLAvariantRecord"/>	
Semantics	:	<input type="text" value=""/>	

Figure 55. Variant Record Datatype Data Input Screen

4.7.14 Notes

The user can add new notes or modify the existing ones.

Normally notes table holds the notes that are referenced by OMT elements. Currently, SimGe does not support referencing notes by OMT elements. In the current status, notes are only related to OMT not to its elements.

4.7.15 Interface Specification Services

Interface specification services usage table is used to track which services are employed for the federation or federate. The table is depicted in Figure 56. The service name, clause of the service, and it-is-a-callback-or-not information cannot be edited by the user, where all are defined in specification [2]. All are loaded by default. The only editable column is the usage column. The user can select which services are used by clicking the checkbox of the related service at usage column.

Some services must always be selected as used according to the object model type (i.e. FOM and SOM). If object model type is FOM, then services: create/destroy federation execution and join/resign federation execution must be used. In case of SOM, only the join/resign federation execution services must be used. According to the user selection, SimGe marks the service row as read only.

Please note that the callback column not specified in [9].

IEEE Std 1516.1-2010 Clause	Service	Usage	Callback
4.2	connect	<input type="checkbox"/>	<input type="checkbox"/>
4.3	disconnect	<input type="checkbox"/>	<input type="checkbox"/>
4.4	connectionLost	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4.5	createFederationExecution	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4.6	destroyFederationExecution	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4.7	listFederationExecutions	<input type="checkbox"/>	<input type="checkbox"/>
4.8	reportFederationExecutions	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4.9	joinFederationExecution	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4.10	resignFederationExecution	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4.11	registerFederationSynchronizationPoint	<input type="checkbox"/>	<input type="checkbox"/>
4.12	confirmSynchronizationPointRegistration	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4.13	announceSynchronizationPoint	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4.14	synchronizationPointAchieved	<input type="checkbox"/>	<input type="checkbox"/>
4.15	federationSynchronized	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4.16	requestFederationSave	<input type="checkbox"/>	<input type="checkbox"/>
4.17	initiateFederateSave	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4.18	federateSaveBegun	<input type="checkbox"/>	<input type="checkbox"/>
4.19	federateSaveComplete	<input type="checkbox"/>	<input type="checkbox"/>
4.20	federationSaved	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4.21	abortFederationSave	<input type="checkbox"/>	<input type="checkbox"/>
4.22	queryFederationSaveStatus	<input type="checkbox"/>	<input type="checkbox"/>
4.23	federationSaveStatusResponse	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4.24	requestFederationRestore	<input type="checkbox"/>	<input type="checkbox"/>
4.25	confirmFederationRestorationRequest	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Figure 56. Interface Specification Services

4.7.16 OMT 1.3 Support

SimGe supports HLA 1.3 OMT files. SimGe Table Editor contains a tab for OMT 1.3 as seen in Figure 57. OMT 1.3 specification incorporates *Routing Spaces* whereas they are omitted in IEEE 1516 specification. The user can make necessary routing space operations in this tab. For this purpose, the routing spaces table is added. The user can create new routing spaces here. Each routing space contains a collection of dimensions. So, the user interface allows adding dimensions to the related routing space. To do this, first select the routing space row in table then select a dimension from the

available dimensions combo box and press add button. To remove a dimension from the routing space list, click the red icon button in the head of each dimension row.

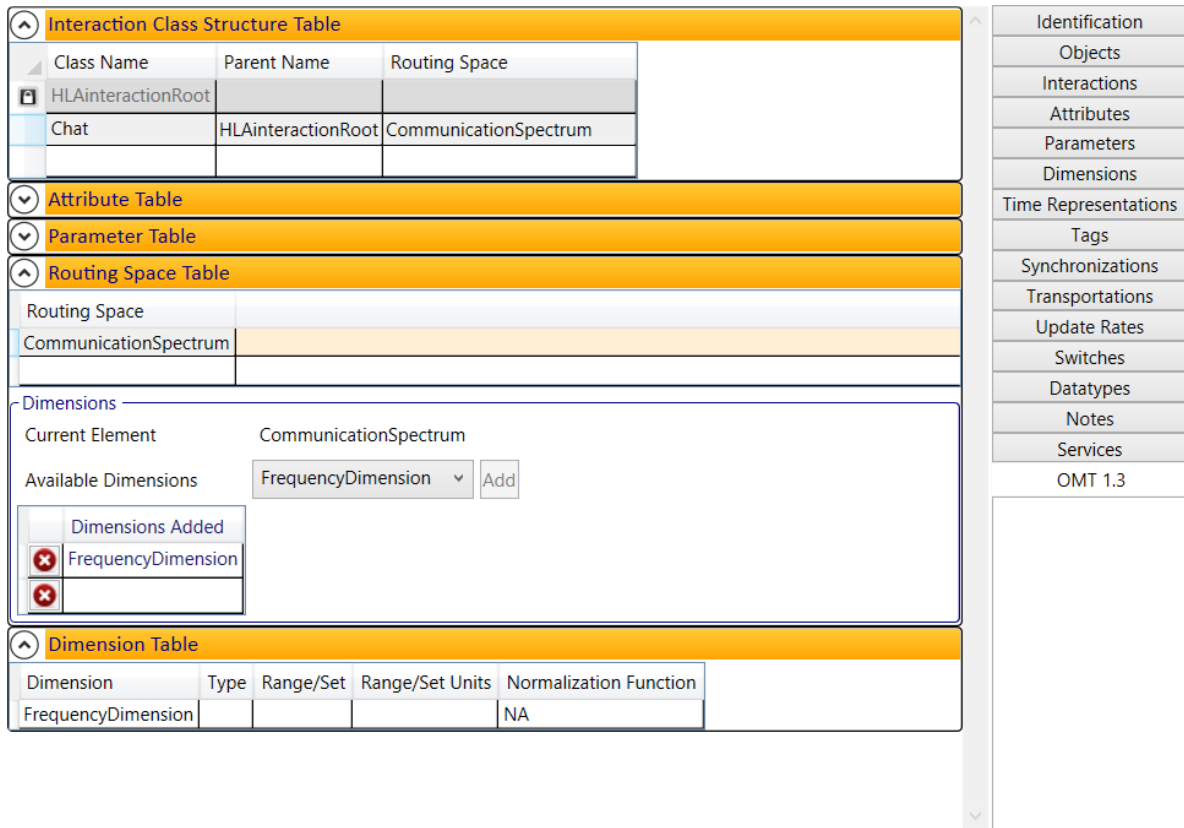


Figure 57. OMT 1.3 Support Tab

Moreover, additional tables are displayed in order to make relations between elements that can be related to the routing space, such as interaction class, attribute, and parameter tables.

4.8 Textual View

The FOM workspace is added to the project UI as a new tab. The object model content is displayed in three forms: (i) Table editor, (ii) text view (FED viewer and FDD viewer), and (iii) diagram editor (not implemented).

4.8.1 Textual View for FED and FDD Files

The textual views provide a read-only look to the FED or FDD files generated from the object model. **Error! Reference source not found.** depicts the textual view of a FED file content. The textual view is read-only and shows the FED file generated by SimGe. The interface for Table editor is depicted in Figure 26 and explained in the following section.

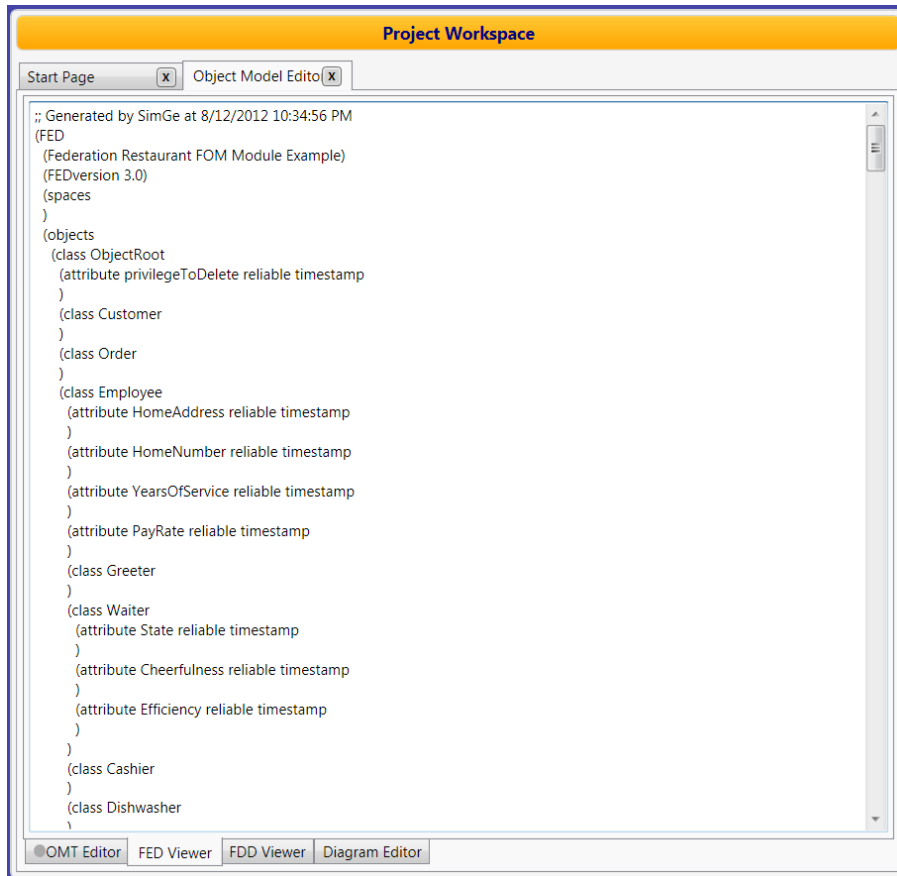


Figure 58. OME FED Viewer

The FDD viewer depicts the FDD file in an XML-style view as shown in **Error! Reference source not found.**

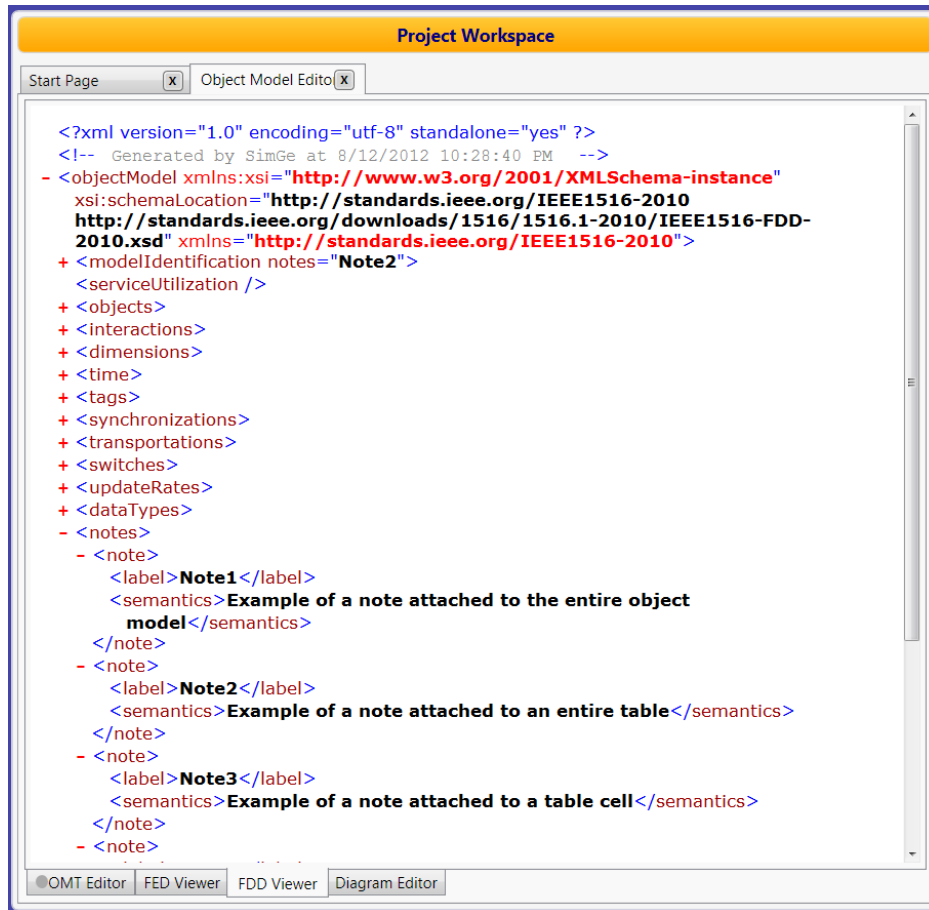


Figure 59. OME FDDViewer

The FDD viewer is implemented as a web browser, when you right click in the area; you have many actions to choose. When you press Ctrl+F, a find dialog appears and you can find any element you want.

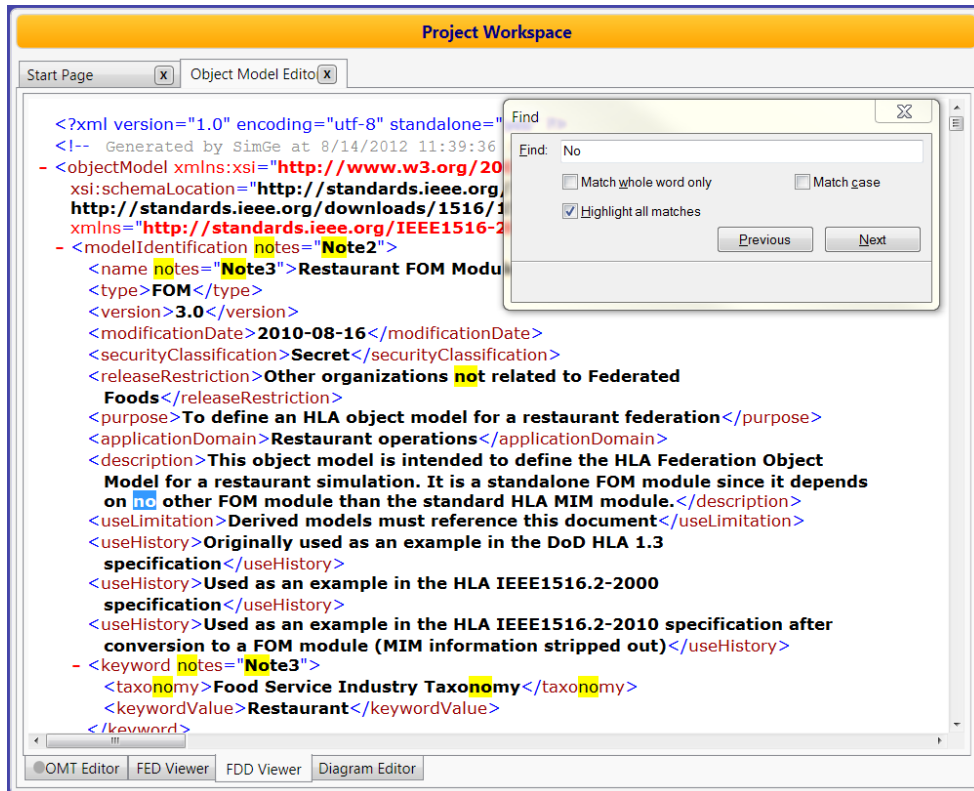


Figure 60. Find Dialog

4.9 Validating FDD File

SimGe OME validates the imported or exported FDD files when export and import is carried out. Moreover, the user can validate the FDD file generated from object model anytime using the validation menu.

The validation results dialog reports the result. The errors found are reported with the line number in the FDD file. SimGe FDD Viewer does not show the line numbers.

4.10 MOM Integration

In HLA 1.3, the MOM is a compulsory part of the FED file. Whenever the user imports a FED file, the MOM is imported too. Most users do not use the MOM. Therefore, SimGe supports suppression of MOM classes. When MOM is suppressed, (1) the table and explorer views do not show the MOM classes, and (2) the code generator does not generate code for MOM classes. MOM suppression can be switched from the OME toolbar as shown in Figure 61.

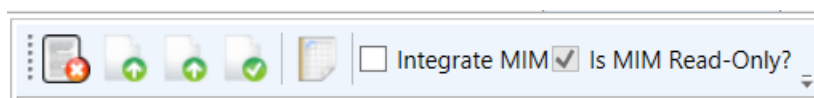


Figure 61. MOM Suppression

When the user wants to work with MOM, then it can disable the MOM suppression switch. MOM will show up in the table and explorers view, but it will be read-only (see Figure 62). When the user wants

to modify the properties of the MOM classes (e.g. Publish / Subscribe (P/S) status of a class), then he/she can disable the read-only switch from the same menu.

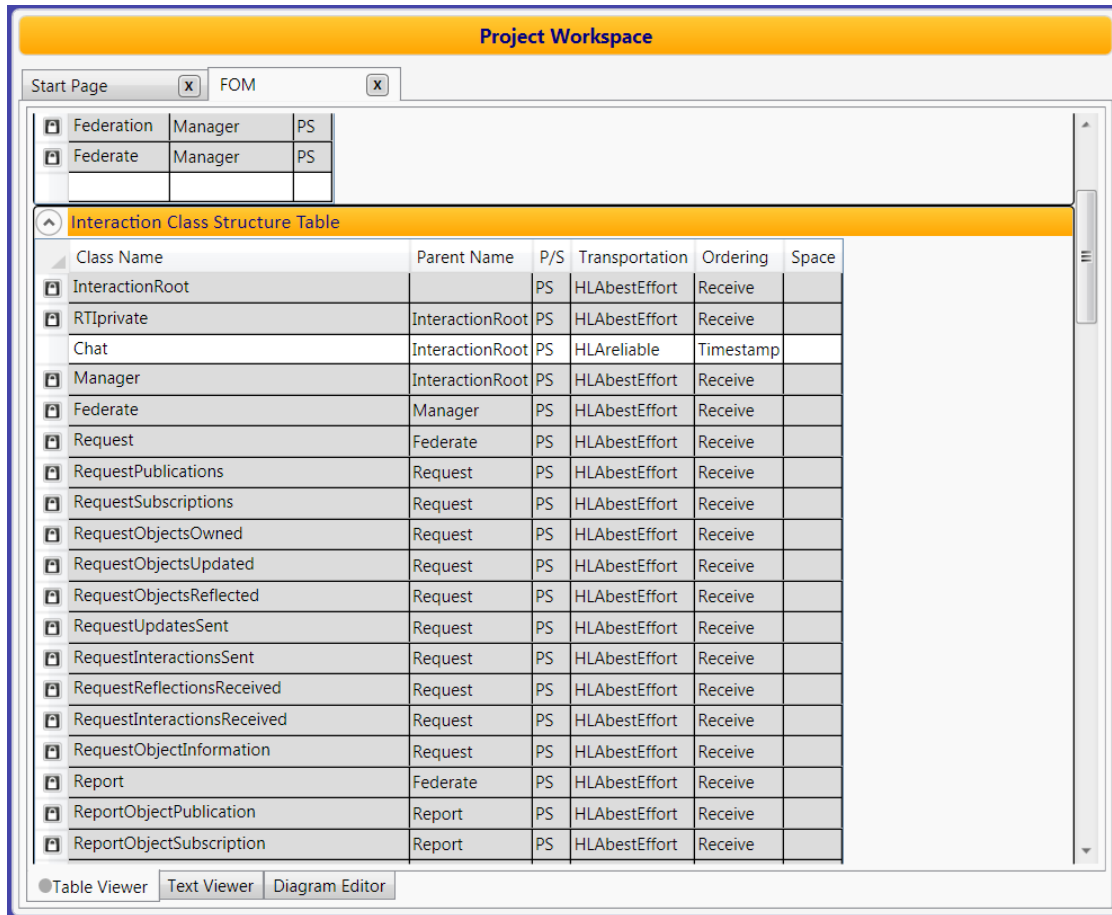


Figure 62. MOM

4.11 OM Explorer

The OM explorer presents a tree view of the OMT presented in the table view. The user can browse the nodes. A blue-colored icon indicates a user-defined element as a red one indicates a built-in element (i.e. a SimGe added element or a MIM/MOM element)

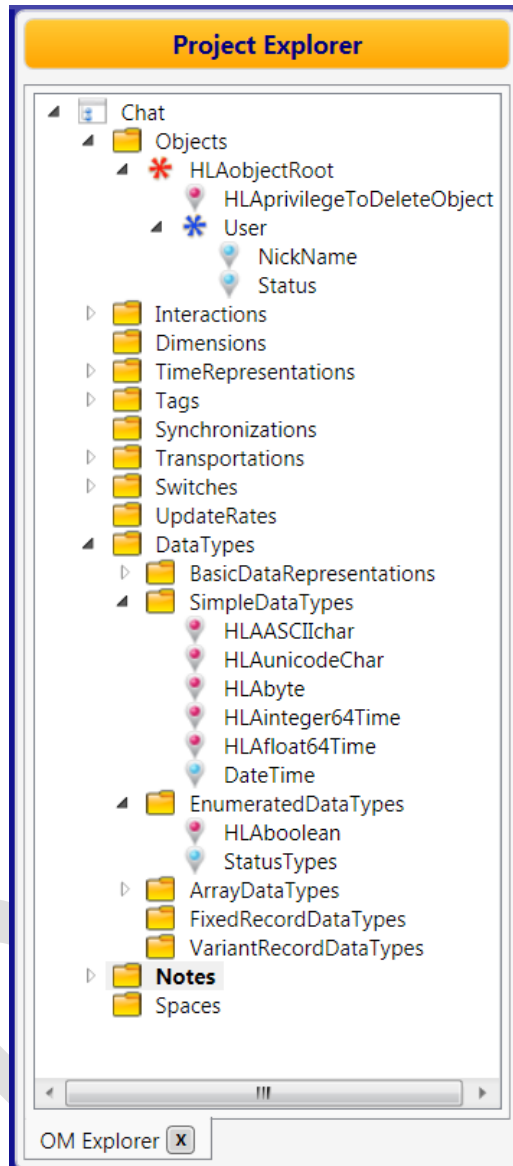


Figure 63. OM Explorer

Currently, the OM explorer supports the following functions via a context menu (i.e. related to the right mouse click):

- Traversing (expand/collapse all, expand/collapse this)
- Modifying (rename and remove)

Doing a right mouse click is required to use most of those functions. The place where you do a right click is important. For instance, if you select a node and then do a right click, then only the related commands will appear.

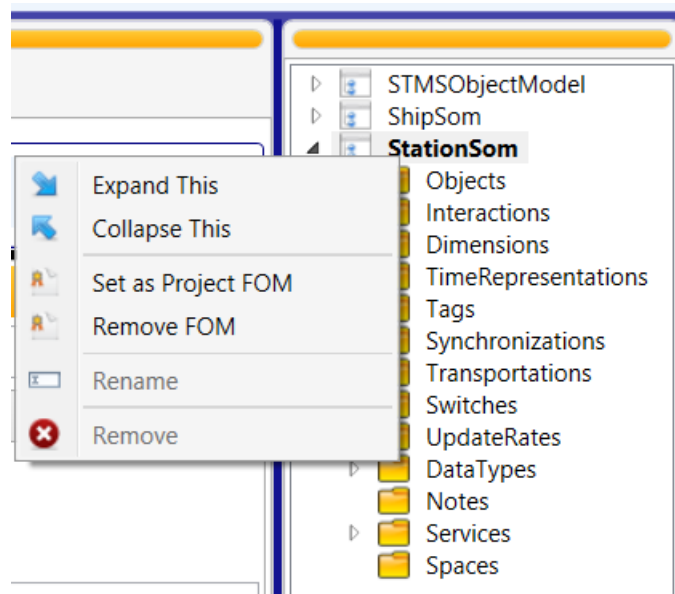


Figure 64. OM Explorer Context Menu

The OM Explorer also is linked to the table editor. When you select a node, then the table tab (e.g. objects) that contains the related item will appear in the table editor.

4.11.1 Traversing Functionality

Expand all and Collapse all: this command provides the ability to expand or collapse all nodes at once in the tree.

Expand this / Collapse this: this command expands or collapses the nodes under the selected node.

4.11.2 Modifying Functionality

Rename: This command provides to rename the selected node. When rename is completed the user must press the enter key.

Remove: This command deletes the selected node and its children (the nodes connected to the selected node).

Remove FOM: This command removes the current federation object model from the project.

5. Federation Architecture Modeling Environment

Federation Architecture Modeling Environment (FAME) is the design environment for the Federation Architecture Model (FAM), or shortly Federation Architecture. For the detailed definition of FAM, please, refer to [10] [11].

FAME enables to design a Federation Structure, where the user can specify the Federation Execution properties and federate properties.

Currently, FAME is under development.

5.1 Creating a FAM

The user can create a new FAM using the “Federation Architecture” menu.

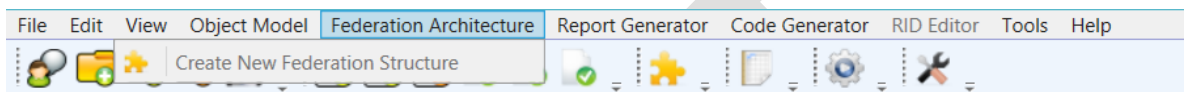
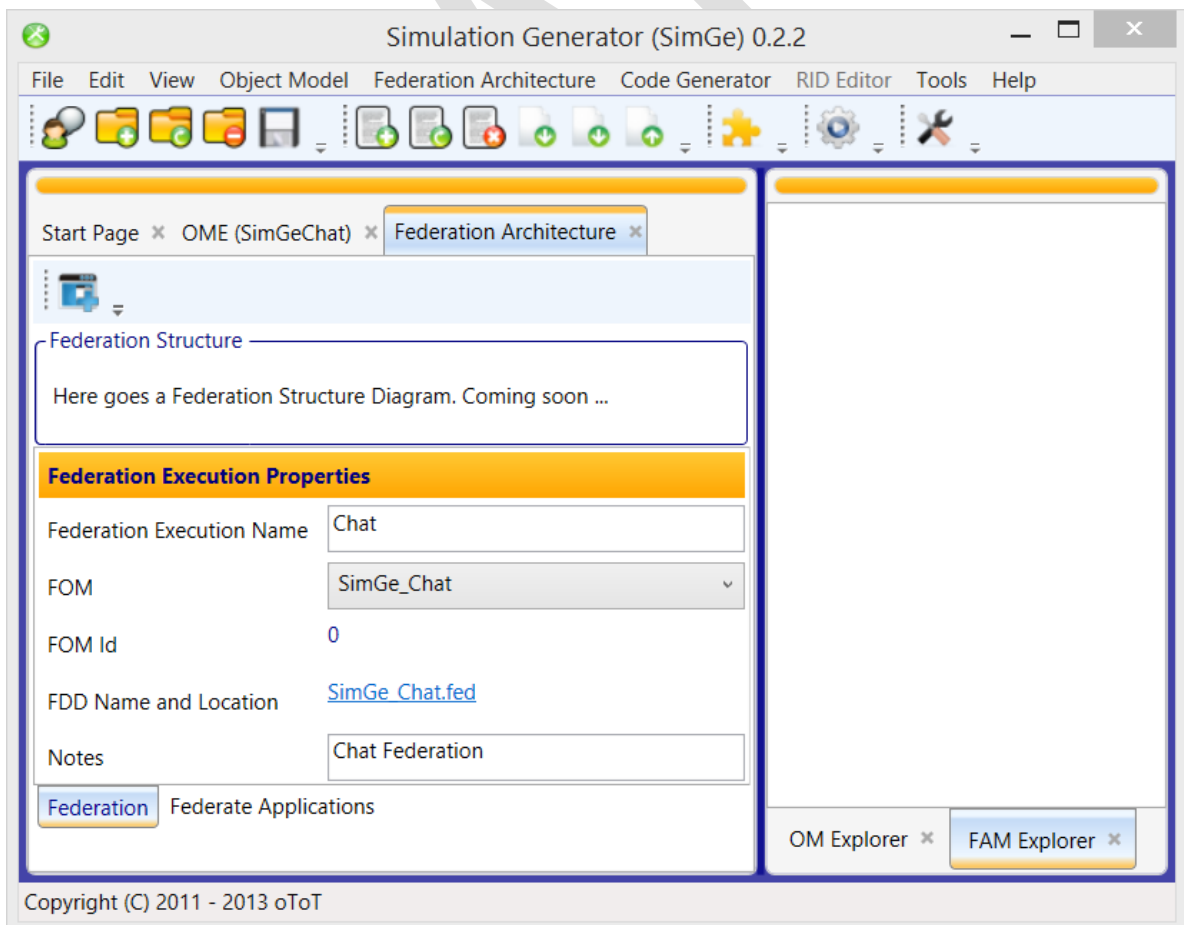


Figure 65. FAM Creation

SimGe will adjust a modeling environment, called FAME, which consists of a new workspace (Federation Architecture) and a new explorer (FAM explorer) to model and to design a federation architecture as seen Figure 66.



5.2 Federation Structure

Federation Structure is the place where the static structure of the federation is designed. The federation structure contains information about the federation execution, such as the location of the FOM Document Data file, the link for the related FOM, and the structure of the federation, where the participating federate applications and their corresponding Simulation Object Models are linked.

5.2.1 Federation Execution Properties

The following table summarizes the Federation Execution Properties.

Table 5. Federation Execution Properties

PROPERTY	EXPLANATION
Federation Execution Name	Federation execution name is the name of the federation registered to RTI when the federation is created.
FOM	FOM property links the federation execution with the object model. When code is generated, the FDD and FED files will automatically generated using this FOM.
FOM Id	This is internally used property. It is read-only. The user cannot set it.
FDD Name and Location	When code is generated, a FED file is generated and placed to the source code folder. When the user clicks the link, a folder explorer will be opened.
Notes	Notes about the federation execution.

5.2.2 Federate Applications

The user can add and remove federate applications to the Federation Structure. The user interface is depicted in the following figure. FAME toolbar consists a button (the only button) to add a new federate application. Each added application will appear in its own tab. In order to remove a federate application, press the close button in its Tab name.

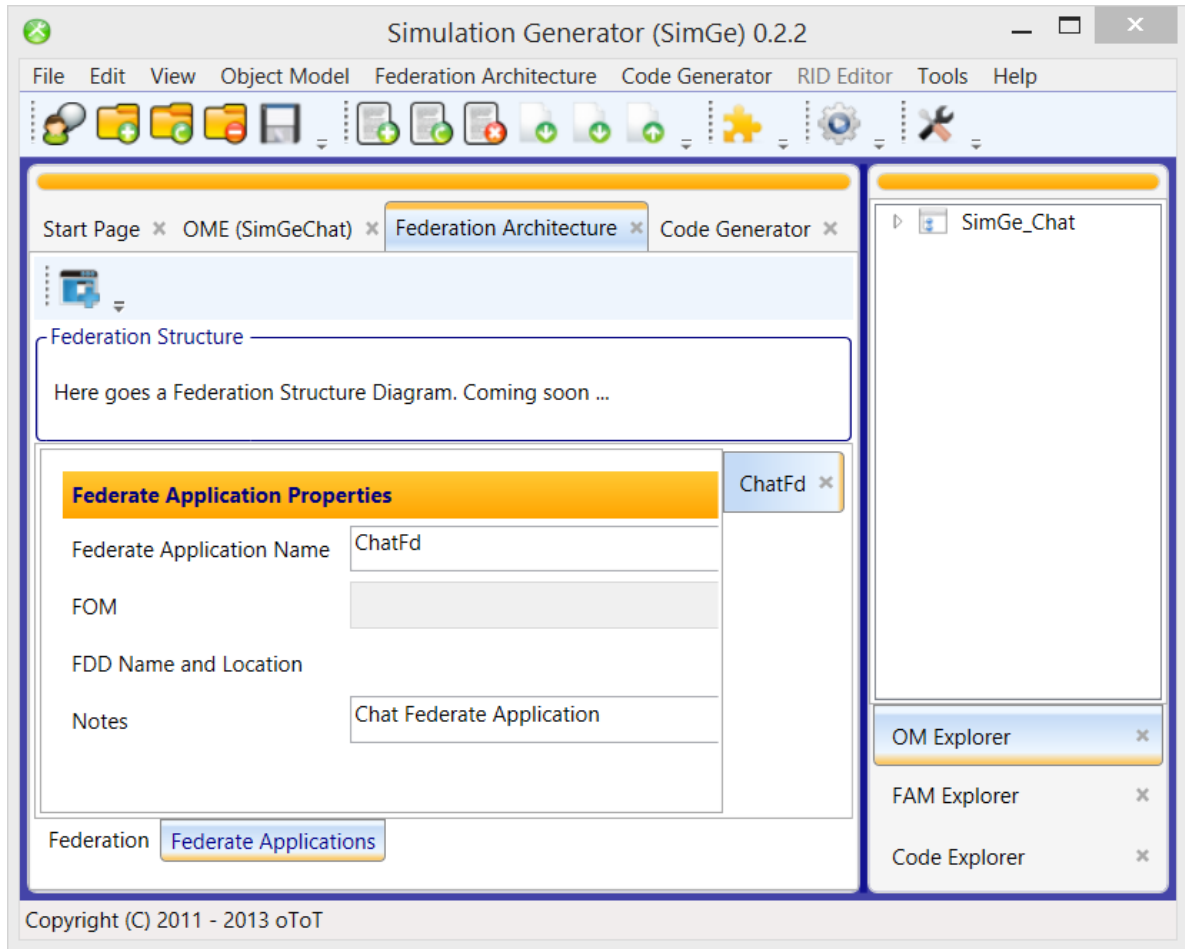


Figure 67. Federate Application Properties

The following table summarizes the Federate Application Properties.

Table 6. Federate Application Properties

PROPERTY	EXPLANATION
Federate Application Name	Federate application name.
SOM	SOM property links the federate with the object model. When code is generated, the FED file is automatically generated using this SOM.
FDD Name and Location	When code is generated, a FED file is generated and placed to the source code folder. When the user clicks the link, a folder explorer will be opened.
Notes	Notes about the federation execution.

5.2.3 Setting Project FOM

In the FOM Explorer, first select the FOM and then right click. In context menu, click “Set as Project FOM”. Now, you can work on the OME with the selected FOM.

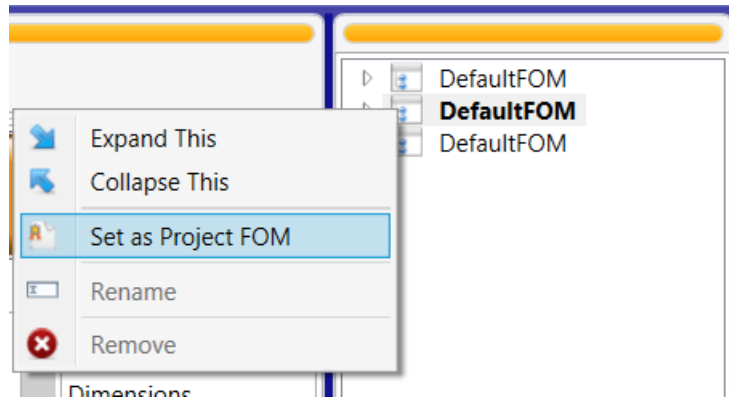


Figure 68. Setting Project FOM

To set the project FOM by using FAME, see Chapter 5.

5.3 FAM Explorer

Not implemented yet ☹

6. Code Generator

The target platform of SimGe code generation is the layered federation architectures that use the RACoN [3] as the communication layer. The generated code files are intended for a fast start-up to the layered federate development. In order to employ the generated code files, a RACoN project must be created and the generated code files must be added to the project in Visual Studio Integrated Development Environment (IDE). Hence, the simulation developer can implement the federate by manually coding on the generated code. See the RACoN programmer's guide [3] for development using RACoN.

Code generator uses the federation architecture model and object models. Before generating code, the user must construct an object model and a FAM that includes the federate applications. Code is generated for each federate application separately. So, it is important to specify the simulation object models for each federate.

Currently, SimGe has the following highlights:

- Code is separately generated for each federate application found in the FAM according to its SOM.
- The structure of the generated code conforms to the layered architecture.
- The following code is generated:
 - Application-specific federate class,
 - The skeleton code for federate ambassador callback event handlers for RACoN,
 - A class for each HLA class found in the object model and supports the inherited classes,
 - The federate SOM class,
 - The simulation Manager class.
- Code generation configuration dialog enables the user to select which management services that the callback handler code will be generated.
- FED file is automatically exported to the source code folder.

When the object model includes the management object model (MOM) classes (i.e. when MOM is not suppressed), the code generator generates MOM code too.

6.1 User Interface

In order to generate code, select Generate in Generator menu. The C# source code files for each HLA classes will be created. Each generation creates a time-stamped folder, where all the generated files are placed here in the `SourceCode` folder under the project home folder. Time-stamp format is year+month+day+hour+min+sec (e.g. 20131110021149).

6.1.1 Code Explorer

The result will be reported in the SimGe Code Explorer (see Figure 69). The user can click the hyperlink to open the folder in windows explorer.

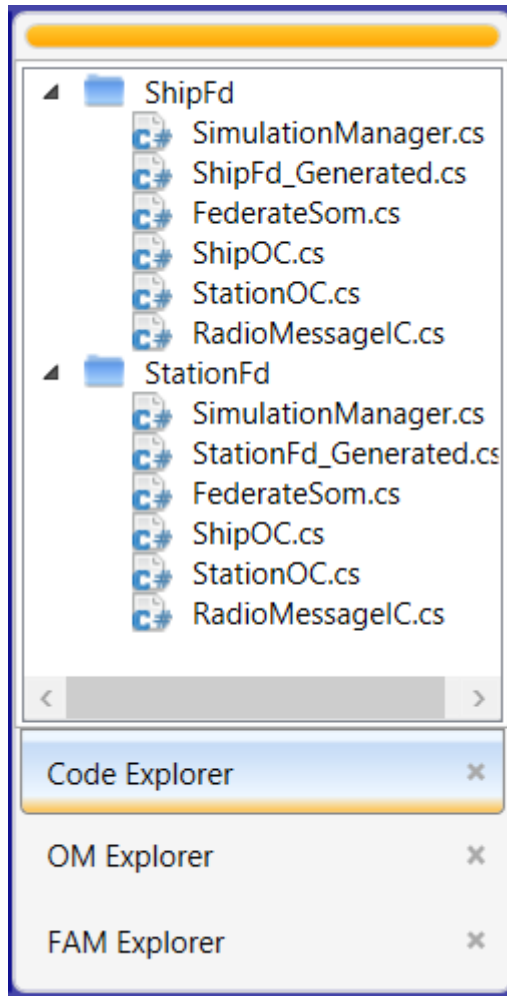


Figure 69. Code Explorer

6.1.2 Code Viewer

The generated code can be seen in the code viewer workspace for each federate application as presented in Figure 70. Please note that this is a read-only view. If required, one must copy or import all the code to the Visual Studio IDE code editor to edit it.


```

Start Page x OME (STMSObjectModel) x OME (ShipSom) x OME (StationSom) x Federation Architecture x Code Viewer (ShipFd) x Code Viewer (StationFd) x
//*****
//
//      CSimulationManager
//
//      generated
//      by          :      Simulation Generator (SimGe) v.0.2.4
//      at          :      Saturday, December 28, 2013 10:56:17 PM
//      compatible with      :      RACoN v.0.0.1.2
//
//      copyright      :      (C)
//      email         :
//*****
//<summary>
/// The Simulation Manager manages the (multiple) federation execution(s) and the (multiple instances of) joined federate(s).
//</summary>
using System;
using RACoN;
namespace stms
{
    public class CSimulationManager
    {
        #region Declarations
        // HLA Part (Communication Layer)
        // Federation Execution
        public RACoN.Federation.CFederationExecution federation;
        // Application-specific Federate
        public CStationFd federate;
        #endregion //Declarations

        #region Constructor
        public CSimulationManager()
        {
            // Initialize the federation execution
            federation = new RACoN.Federation.CFederationExecution();
            federation.FederationExecutionName = "Bhosphorus";
            federation.FDD = @"..\STMSObjectModel.fed";
            // Initialize the application-specific federate
            federate = new CStationFd();
        }
    }
}
SimulationManager.cs StationFd_Generated.cs FederateSom.cs ShipOC.cs StationOC.cs RadioMessageIC.cs

```

Figure 70. Code Viewer and Generated Code Sample

The style and naming convention of the generated code conforms to [12]. The following figure depicts a part of the generated code.

6.2 Architecture Style

SimGe Code generator (CodeGen) generates code that conforms to the Layered Simulation Architecture style [5] [4]. The layers of a federate application is given in Figure 71.

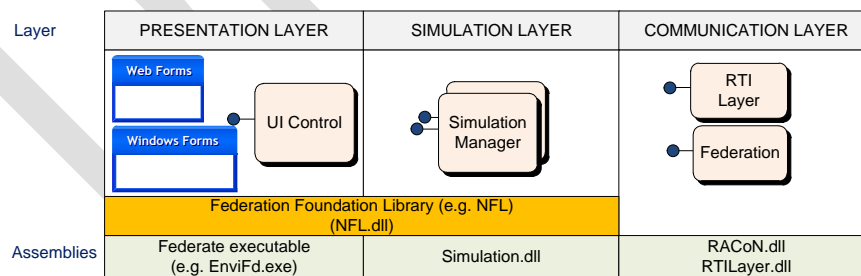


Figure 71. Federate Application Design [5]

The *presentation layer* (also called the *user interface layer*) includes the pure presentation (view), input, and interaction with the user (if involved). The main components are the graphical user interface (GUI), which can be developed using a GUI library (e.g. Microsoft Windows Forms¹, Java Swing class²) and a user interface controller. The UI controller binds the user interface widgets with simulation data,

¹ Microsoft Windows Forms, www.microsoft.com, last accessed November 13, 2011.

² Java Swing Classes, www.java.com, last accessed November 13, 2011.

manages and handles the user events, and iterates the simulation flow (i.e. the main simulation loop). The presentation layer is separately developed for each federate.

The *simulation layer* (also called the *processing layer*) includes the computation (the simulation) and the local data (the federate-specific objects). Its purpose is to generate the federate behavior. The simulation layer is federate application specific.

The *communication layer* deals with the HLA runtime infrastructure (RTI) level communication in order to access the federation-wide data (namely, the objects and interactions exchanged in the federation execution) using an RTI abstraction component for .NET (called RACoN). RTI is a middleware that manages the federation execution and object exchange through a federation execution.

Due to the presentation layer heavily depends on the project, CodeGen only generates code partially for the simulation layer and the communication layer. The class structure of the generated code is presented in Figure 72.

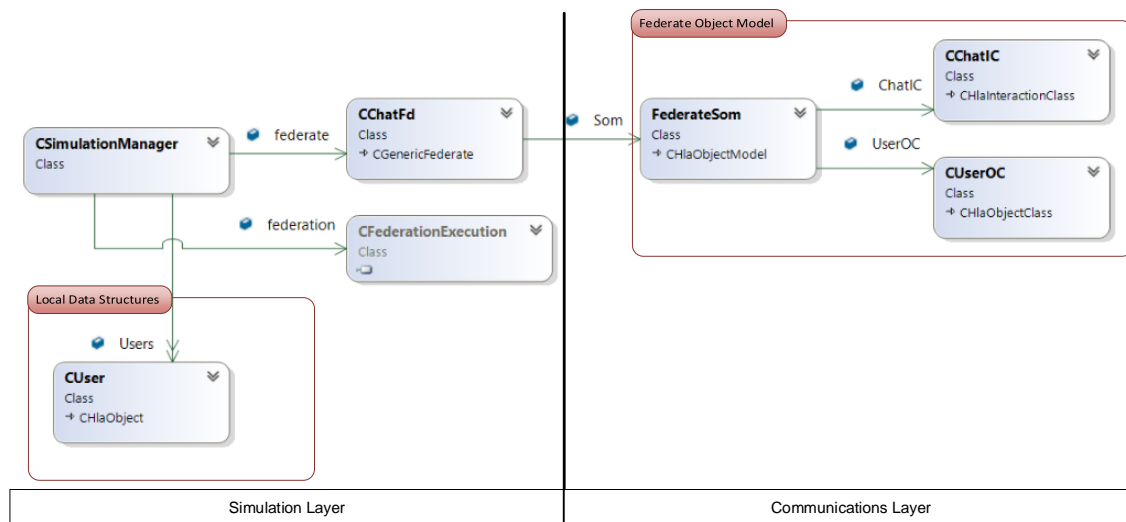


Figure 72. Class Diagram

Each generated class is placed in a separate C# file. Classes are grouped as follows:

- Simulation Layer
 - Simulation manager class (i.e. CSimulationManager)
 - Application-specific federate class (e.g. CChatFd)
 - Federation execution class (i.e. CFederationExecution)
- Federate Object Model
 - Federate SOM class (i.e. FederateSOM)
 - Interaction Classes for each interaction defined in FOM. For instance, CChatIC class.
 - Object classes for each object defined in FOM. For instance, CUserOC class.

6.2.1 Federate Object Model

All the necessary object model classes are generated using the project FOM. The federate SOM class is the main container class for the interaction and object classes.

6.2.1.1 Object/Interaction Classes

Each files for HLA classes are named as following the rule: HLA class name plus “IC” or “OC” added according to the HLA class type and then “.cs” extension is added.

6.2.2 Simulation Layer Code Generation

6.2.2.1 Application-specific Federate Class

The application-specific federate class code file is named as the project name with “.cs” extension and the federate SOM class code file named as “FederateSom.cs”.

The application-specific federate class includes the skeleton code for federate ambassador callback event handlers. The user generally will edit this class. Therefore, it is generated as a partial class, where it can be dispersed to multiple files. The user can easily create a manual copy of this class and edits this file instead of the generated file in order not to lose the changes when it is generated again as depicted in Figure 73.



Figure 73. Application-specific Federate Class

6.2.3 Management Object Model Code Generation

When MOM suppression is enabled (which is the default behavior), then MOM class code will not be generated, otherwise it will.

6.3 Code Generator Configuration

SimGe provides a Code Generation Configuration Dialog where the user can select which management services that the callback handler code will be generated. The code generation options are found in Tools/Options menu.

Code generator settings are divided into two groups: (1) general settings and (2) callback settings.

6.3.1 General Settings

General settings contain two groups: (1) the writer settings, (2) code file metadata, and (3) folders as seen in Figure 74.

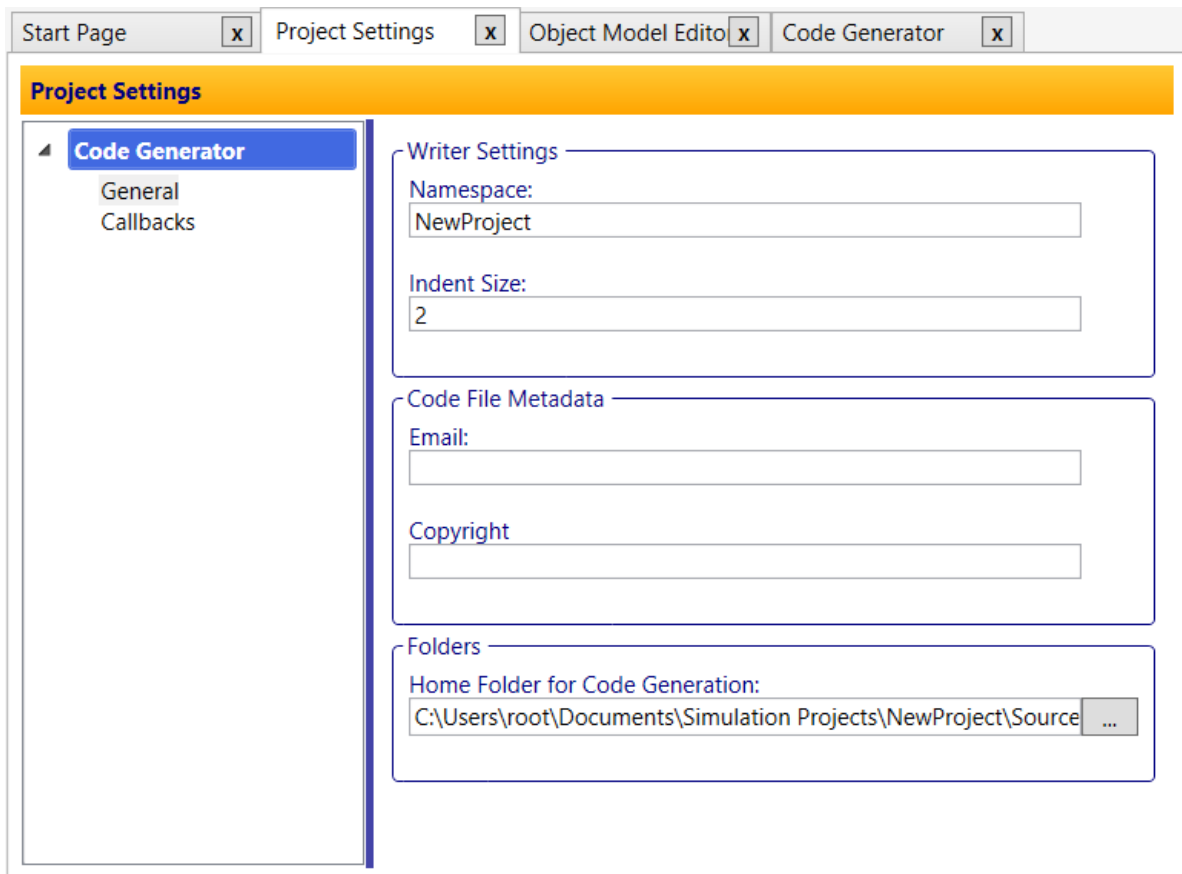


Figure 74. Code Generator Settings - General

The following table summarizes the parameters.

Table 7. Code Generator General Settings

PARAMETER	EXPLANATION
Namespace	Namespace is the namespace name of all the classes generated. It is the project name by default. The user may override it here. Note that naming must conform to MS .NET Framework naming rules. See Figure 75.
Indent Size	It is the indentation (space character) count for the nested codes. Indent size must be equal or greater than zero. See Figure 75.
Email	The email of the developer. See Figure 75.
Copyright	Copyright information. See Figure 75.
Home Folder	This is the home folder where the code generator will create a time stamped folder that contains all the code files after code generation.

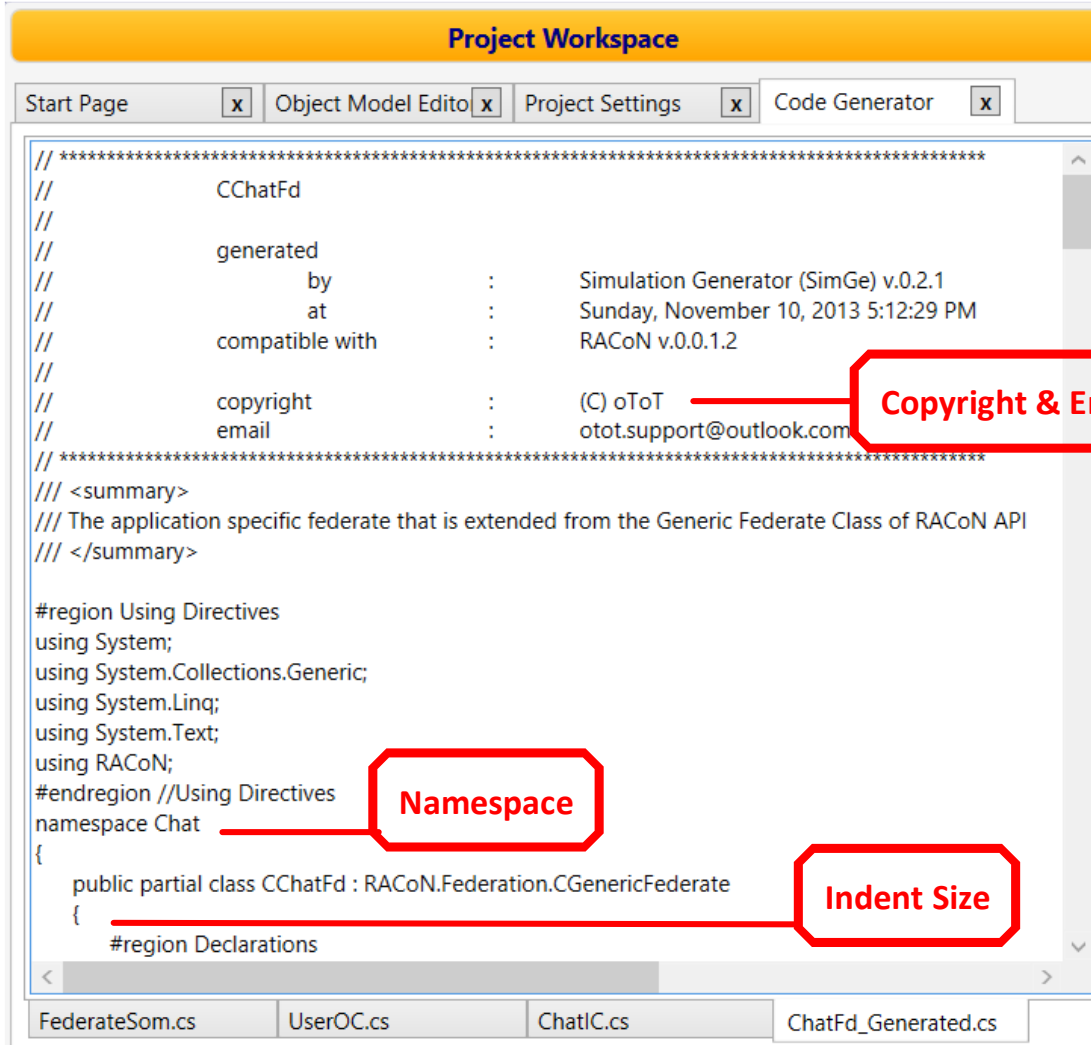


Figure 75. Generated Code Parameters

6.3.2 Callback Settings

In the callback settings pane, you can select the callbacks, which will be generated by default as seen in Figure 76.

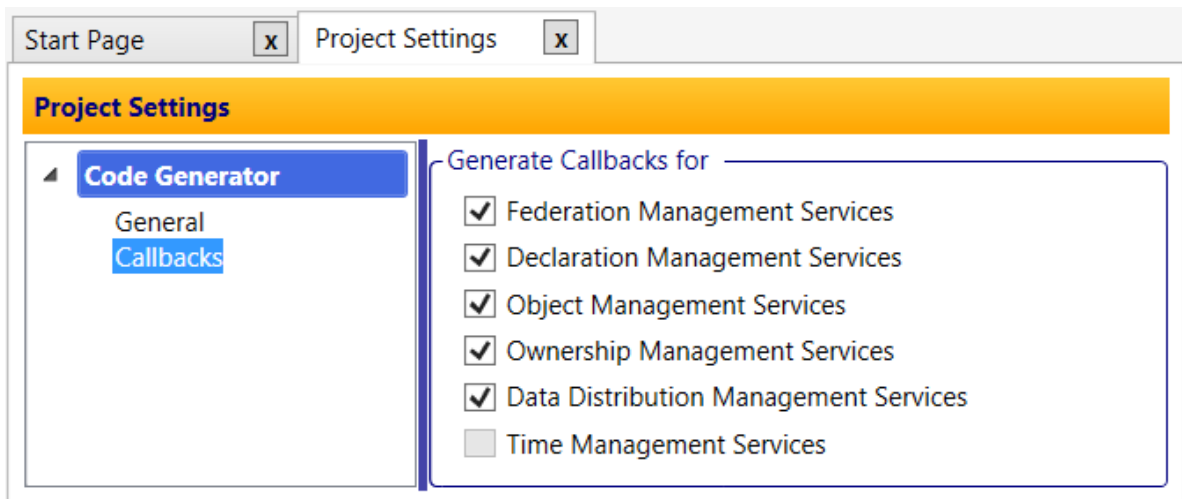


Figure 76. Code Generator Settings - Callbacks

7. Report Generator

SimGe Report Generator fully generates HLA OMT 1516-2010 specification [9] tables for documenting purposes. The report generator can be run using the OME Toolbar (see Figure 77), where each report is specific to an object model.

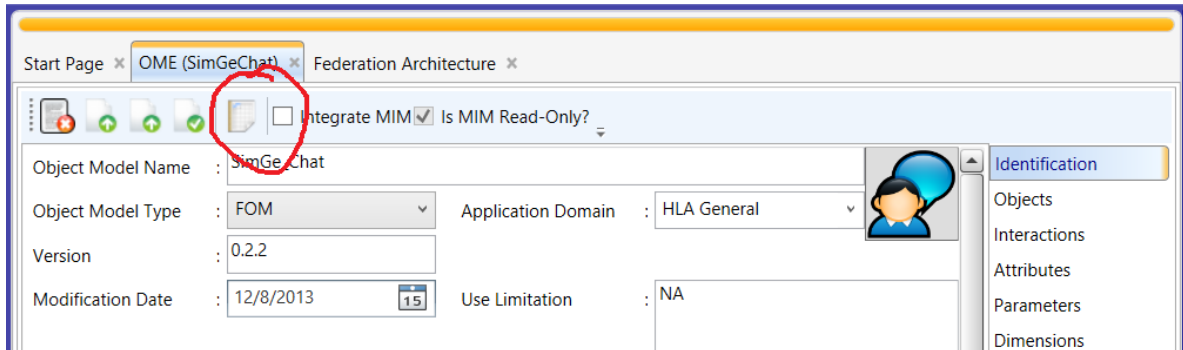


Figure 77. Running Report Generator

When report generator tab is displayed, the user can print or export (save) the report to PDF, Excel or Word format. The page layout is portrait and its size is A4 (29.7cm x 21cm).

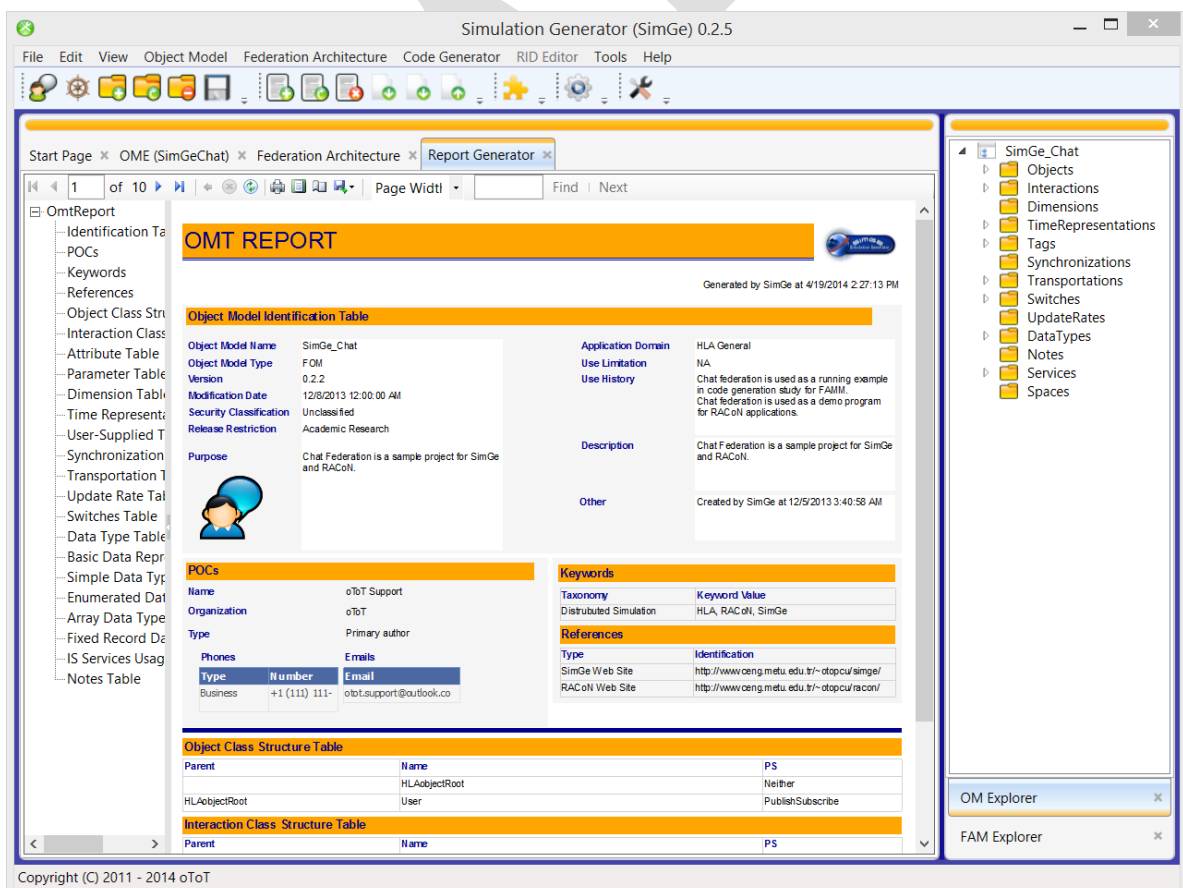


Figure 78. Report Generator Tab

A document map is also created automatically. When you export the report, then the document will help to navigate the document (e.g. for a pdf document, the document map will be converted to a list of bookmarks).

Currently, the generation of the first report sometimes takes some time. Do not worry, it will run.

8. Case Studies

There are two case studies coming with SimGe distribution. The first one is Chat federation where it is a single-federate application, which involves basic HLA functionality. The second one is more advanced and called as the Strait Traffic Monitor Simulation (STMS), which includes two federates with advanced data distribution management services.

8.1 Chat Federation

SimGe comes with a sample project called Chat Federation. Chat is an HLA based distributed interactive application that provides basic chatting functionalities such as selecting a nickname, entering a chat room, and so on. By using the chat client, a federate application, one can exchange messages with his/her friends in chat rooms. Before entering a chat room, he/she has to pick up a unique nickname. The “chat client” provides a graphical user interface for the user interaction and deals with the RTI communication. The conceptual view of the application is presented in Figure 79.

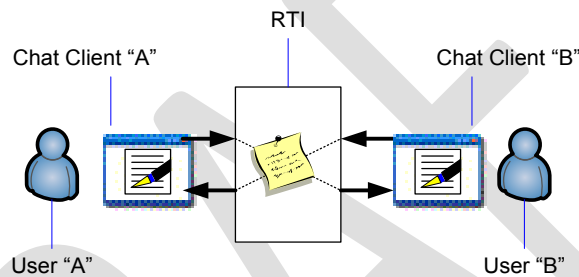


Figure 79. Chat Application Conceptual View

Please, note that this section is not complete yet. For the implementation of a chat federate application, see [3].

8.1.1 Loading Sample Project

In File menu click Sample Projects and then Chat Federation or more shortly click the sample project icon at toolbar as depicted in Figure 80. The default project files will be loaded as read only. SimGe allows modification to the default sample project but you cannot save it. In order to save it, use Save As command.



Figure 80. Loading Sample Project

The sample project is located in the default *Windows Common Application Data Folder* (i.e. generally C:\ProgramData\ for Windows 7) and named as SimGe/Sample.

8.2 Strait Traffic Monitoring Simulation (STMS)

This case study is first appeared in [10]. A traffic monitoring station tracks the ships passing through the strait. Any ship entering the strait announces her name and then periodically reports her position to the station and to the other ships in the strait using the radio channels. Channel-1 is used for ship-to-ship and channel-2 is used for ship-to-shore communication. The traffic monitoring station tracks ships and ships track each other through these communication channels. All radio messages are time-stamped to preserve the transmission order.

The traffic monitoring station and the ships are represented with two types of applications¹: a station application and a ship application, respectively. The ship application is an interactive federate allowing the player to pick up a unique ship name, a direction (eastward or westward), and a constant speed by means of a textual interface. Joining a federation corresponds to entering the strait, and resigning from the federation corresponds to leaving the strait. The station application is a monitoring federate, which merely displays the ships (in the strait) and their positions. The federation has a time management policy where each ship application is both time regulating and time constrained and station application is only time constrained.

The conceptual view of the application is presented in Figure 81.

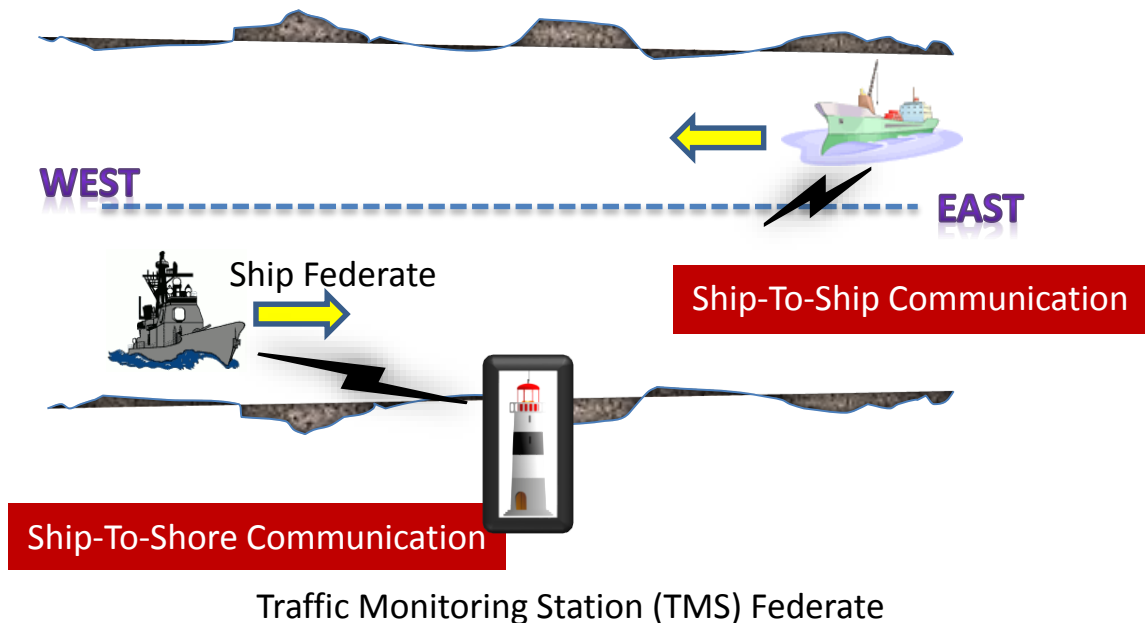


Figure 81. Strait Traffic Monitoring Simulation Conceptual View

While selecting this example, the following highlights were in mind:

- Clearly, the essence of this simple federation is an example of a set of objects tracking each other making it a common scenario/interaction for most distributed simulations.
- It is believed that this example has a simple conceptual model, which will make it easily understandable and capture the reader's attention immediately. Thus, it will force the user focus on the modeling part than the example itself.

¹ Application is used as a substitute for "federate application".

- Moreover, the sample federation naturally includes time management, ownership management, and data distribution management services in addition to the base services (e.g., federation management services).
- The sample federation involves two distinct federate applications and it has a potential to support multiple federations.
- It is an interactive simulation. Thus, it presents how to model the user interactions in FAME.

For the complete federation architecture, along with other supporting material, the reader is referred to [10].

8.2.1 Object Model

An appropriate FOM for the STMS federation is prepared conforming to the HOMM. The object class and interaction class hierarchies of the object model are presented in Figure 82. The FOM involves two object classes, namely “ship” and “station”, and one interaction class, namely “radio message”. The ship object has four attributes, namely “call sign”, “course”, “speed”, and “position”, and the station object has two attributes, namely “station name” and “location” as the radio message interaction class has two parameters, namely, the “call sign” and “message” parameters, indicating “the name of the entity that sent the message” and “the content of the message (i.e., the position data)”, respectively.

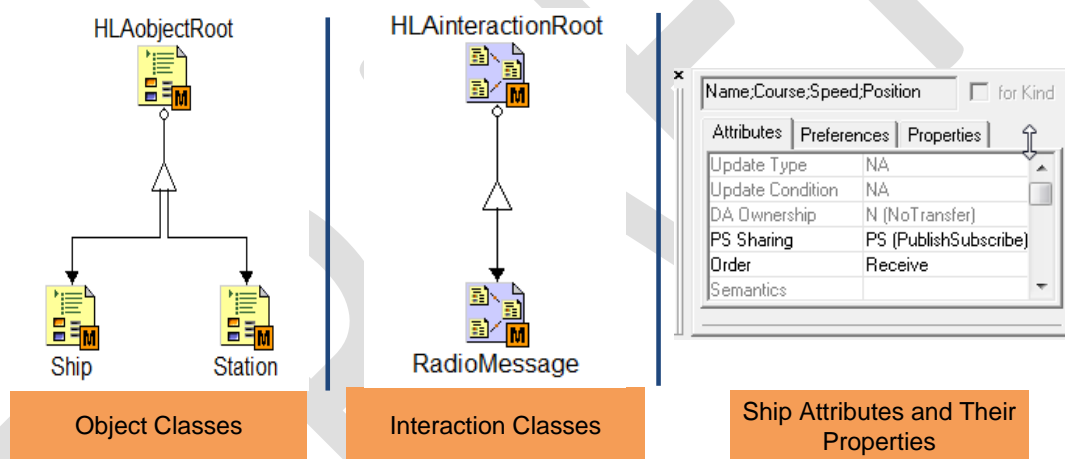


Figure 82. A Part of the STMS FOM

8.2.2 Federation Architecture

In the federation structure model of STMS, the connection is made for the federation and federate applications with the FOM and SOMs, respectively. The conceptual model is depicted in Figure 83. The federation is named “Bhosphorus Federation”. The multiplicity information is also supplied while connecting the applications to the federation. The ship application may join the federation multiple times while the station application is limited to one in this specific scenario. The lower pane of the screen shot shows this multiplicity.

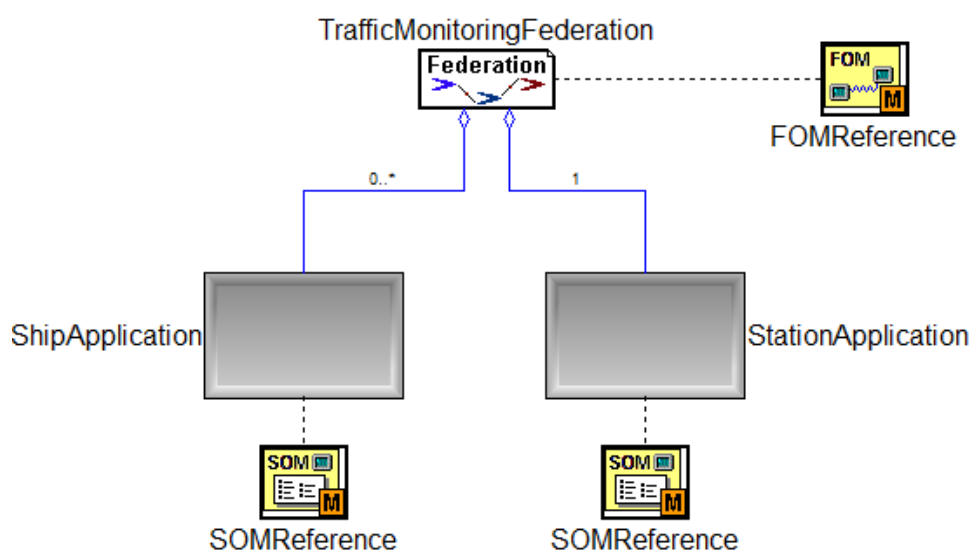


Figure 83. The Strait Traffic Monitoring Federation Structure Model

8.2.3 Data Distribution Management

As an example, a one dimensional **communication** space for radios is created to illustrate a distribution region. In our example, radio communication is carried out using the radio channels: channel-1 for ship-to-ship communication and channel-2 for ship-to-shore communication. Channels are regions over the channel dimension (there are two channels from zero to 3). Channels are defined by the dimension numbers as shown in Figure 84.

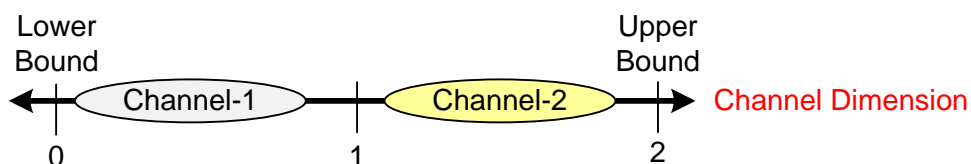


Figure 84. DDM Example

First, the dimensions, which constitute a region, must be defined in the federation object model. Each dimension has a type and a normalization function as described in [9]. Type is a reference to a pre-defined type in Data Types. In the normalization function element, one can specify the upper limit.

Regions are handled via the HLA DDM methods such as CreateRegion and CommitRegionModifications calls. All these methods have a reference to the regions defined in the variable lists. CreateRegion method call creates the regions while SetRangeBounds method call sets the boundaries for the regions.

References

- [1] IEEE Std 1516.1-2000, "Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification," IEEE, 2010.
 - [2] IEEE Std 1516.2-2000, Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Object Model Template Specification, IEEE, September 2000.
 - [3] O. Topçu, RACoN Programmer's Guide, 2011.
 - [4] O. Topçu and H. Oğuztüzün, "Mutli-layered Simulation Architecture: A Practical Approach," London, 2011.
 - [5] O. Topçu and H. Oğuztüzün, "Layered Simulation Architecture," 2012.
 - [6] M. S. V. DMSO, Federation Execution Details (FED) File Specification RTI 1.3 Version 3, 31 July 1998.
 - [7] IEEE Std 1516.1-2010, "Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules," IEEE, New York, 2010.
 - [8] DMSO, "High Level Architecture Object Model Template Specification Version 1.3 (Draft)," U.S. Department of Defense, 1998.
 - [9] IEEE Std 1516.2-2010, "Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) -Object Model Template," IEEE, New York, 2010.
 - [10] O. Topçu, M. Adak and H. Oğuztüzün, "A Metamodel for Federation Architectures," *Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 18, no. 3, pp. 10:1-10:29, July 2008.
 - [11] O. Topçu, *Metamodeling for the HLA Federation Architectures*, Ankara, Turkey: The Computer Engineering Department, The Graduate School of Natural and Applied Sciences, Middle East Technical University (METU), 2007.
 - [12] "C# Coding Standards and Naving Conventions," Data & Object Factory, LLC, [Online]. Available: <http://www.dofactory.com/reference/csharp-coding-standards.aspx>. [Accessed 23 04 2011].
 - [13] IEEE 1516.2, Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Object Model Template Specification, IEEE, September 2000.
 - [14] O. Topçu and H. Oğuztüzün, "Federation Architecture Metamodel," Middle East Technical University, Ankara, 2008.
 - [15] A. Ledezci, A. Bakay, M. Maroti, P. Volgvesi, G. Nordstorm, J. Sprinkle and G. Karsai, "Composing Domain-Specific Design Environments," *Computer*, vol. 34, no. 11, pp. 44-51, 2001.
 - [16] ISIS, A Generic Modeling Environment GME 7 User's Manual v7.0, Institute for Software Integrated Systems Vanderbilt University, 2007.
 - [17] O. Topçu and H. Oğuztüzün, "Developing an HLA Based Naval Maneuvering Simulation," *Naval Engineers Journal*, vol. 117, no. 1, pp. 23-40, Winter 2005.
 - [18] A. Molla, K. Sarioğlu, O. Topçu, M. Adak and H. Oğuztüzün, "Federation Architecture Modeling: A Case Stufy with NSTMSS," in *Fall Simulation Interoperability Workshop (SIW)*, 2007.
 - [19] A. Molla, "Modelling of Exercise Planner Federate with Federation Architecture Metamodel," METU, Ankara, 2007.
 - [20] K. Sarioğlu, "Modeling Federation Monitor Federate," METU, Ankara, 2007.
 - [21] D. Çetinkaya, "A Metamodel for the High Level Architecture Object Model," Middle East Technical University, Ankara, 2005.
 - [22] IEEE 1516.1, Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification, IEEE, September 2000.
-

- [23] O. Topçu, M. Adak and H. Oğuztüzün, "A Metamodel for Federation Architectures," *Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 18, no. 3, July 2008.
- [24] M. Adak, O. Topçu and H. Oğuztüzün, "Model-based Code Generation for HLA Federates," *Software: Practice and Experience*, 2009.
- [25] O. Topçu, M. Adak and H. Oğuztüzün, "Metamodeling Live Sequence Charts for Code Generation," *Software and Systems Modeling (SoSym)*, 2009.
- [26] Wikipedia, "Unmanned Surface Vehicle," 2011. [Online]. Available: http://en.wikipedia.org/wiki/Unmanned_surface_vehicle.
- [27] U. Navy, "The Navy Unmanned Surface Vehicle (USV) Master Plan," U.S. Department of the Navy, 2007.

List of symbols/abbreviations/acronyms/initialisms

CG	Code Generator
CRUD	Create, Read, Update, Delete operations
FED	Federation Execution Details
FAM	Federation Architecture Model
FAME	Federation Architecture Modeling Environment
FAP	Federation Architecture Project
FDD	FOM Document Data
FOM	Federation Object Model
HLA	High Level Architecture
IEEE	Institute of Electrical and Electronic Engineers
MIM	Management Initialization Model
MOM	Management Object Model
M&S	Modeling and Simulation
MRU	Most Recently Used
OME	Object Model Editor
OMT	Object Model Template
P/S	Publish/Subscribe
RACoN	RTI Abstraction Component for .NET
RG	Report Generator
RTI	Runtime Infrastructure
SimGe	Simulation Generator
SOM	Simulation Object Model
UI	User Interface

Index

C

Code Explorer · 57
Code Generation Configuration Dialog · 61
CRUD · 24

F

Federation Architecture · 53
Federation Architecture Model · 53
Federation Architecture Project · 16
Federation Execution Properties · 54, 55

L

Layered Simulation Architecture · 59

M

Management Object Model (MOM) · 19
MIM · 20

O

Options · 18

P

Project Explorer · 15
Project FOM folder · 19
Project Home Folder · 16
Project Settings · 17
Project Start Page · 17
Project Workspace · 15
Publish / Subscribe · 50

R

RACoN · 10
Report Generator · 65
Routing Spaces · 45

S

SimGe · 10
Simulation Project · 15