Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

NEUTRAL TEMPLATE LIBRARIES FOR EFFICIENT DISTRIBUTED SIMULATION WITHIN A MANUFACTURING SYSTEM ENGINEERING PLATFORM

Kai Mertins
Markus Rabe
Frank-Walter Jaekel

Corporate Engineering Division
Fraunhofer-IPK
Pascalstr. 8-9, 10587 Berlin, GERMANY

ABSTRACT

The MISSION project develops an environment for integrated applications of simulation tools which can be offered by different vendors. The template library supports the generation of models from the view of the application instead of simulation tool features. The selection of simulation tools applied is performed with the mostly completed, but still neutral model. The template library is a reservoir of neutral re-usable elements incorporating their major attributes, and referencing to implementations of these models in different simulation tools. Within the manufacturing system engineering (MSE) process, the template library is mainly used as a flexible knowledge base. For this purpose, attributes can be defined depending on the design agents applied. Some attributes are predefined according to the requirements of the MISSION modelling platform (MMP) or according to available user requirements. The user has the chance to add templates and attributes of templates. Furthermore, the user can use objects of these templates within the MSE process. Concerning the simulation process, the template library contains for each application template a reference to simulation models. The simulation model implements the content of the template. The paper presents the template library approach and a short introduction to the MISSION platform.

1 INTRODUCTION

Global Enterprises have to face new ways of distributed work (Mertins et al. 1998a). Within this huge field, MISSION focuses on the Manufacturing Engineering process and, furthermore, on simulation. The global approach is enhanced by the integration of three regions from Japan, Europe and USA.

The E.U. and U.S. partners have defined a common architecture, called the MISSION General Architecture (McLean et al. 2000). The European demonstrator architecture (Figure 1) is one instance of the MISSION general architecture.

Figure 1 shows the outline of the European demonstrator architecture for the MMP, based on the perceived requirements for integrating the activities within the Manufacturing System Engineering Integration Infrastructure. The general goal of MISSION is the support of the Manufacturing System Engineering (MSE) process by integrating the tools, which can be employed in various aspects of the process. However, in order to enhance and accelerate the process, they must refer particularly to both, the usage of simulation, and the integration of simulation tools with other software tools. The suggested architecture therefore incorporates the MSE integration as well as the distributed simulation of the manufacturing processes.

The following are prerequisites for a comfortable use of the Mission Modelling Platform (MMP).

- Manufacturing Process - Run Time Interface (MS-RTI)
- Manufacturing System Engineering Integration Infrastructure (MSE Integration Infrastructure)
- Information Manager
- Project Agent
- MSE Moderator
- Template Library
- Simulation Manager

The European MISSION demonstrator includes a specification and first prototypes of MMP components. However, the specification will be made public. This enables software producers to develop and market their own tools for the MMP which can replace the prototypes developed within the project.

The demonstrator will show how the MMP can be used to bridge the gap between different simulation model

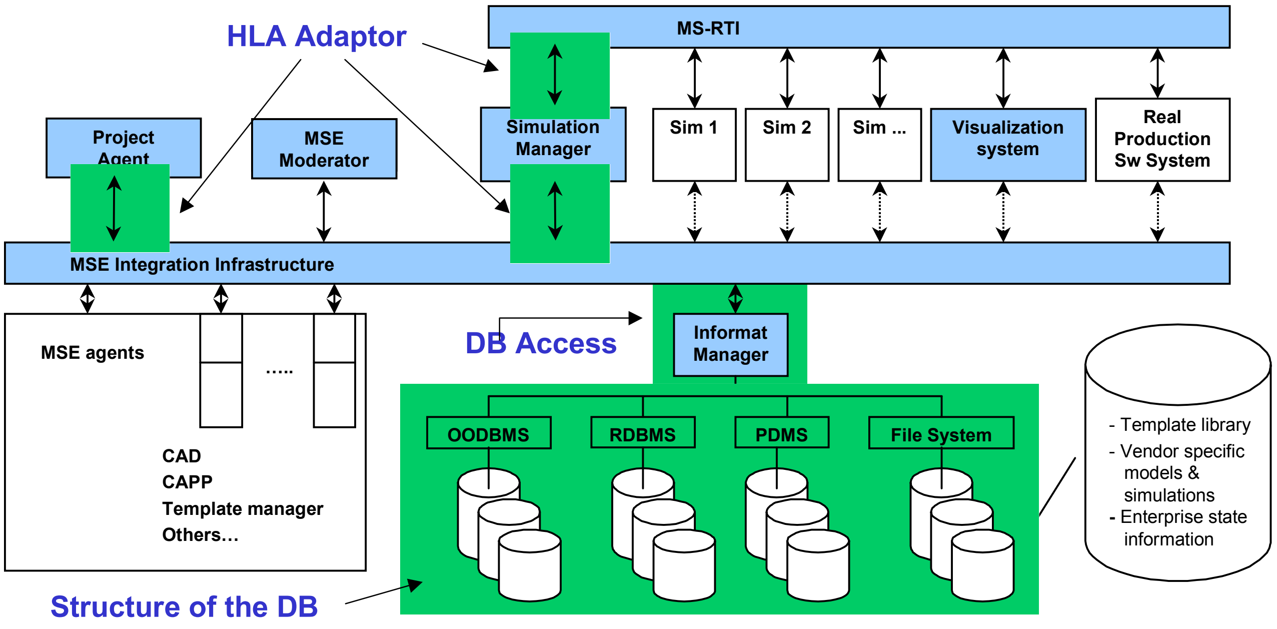


Figure 1: MISSION Modelling Platform Architecture

islands. In addition, the bridge between the simulation and the necessary information available within different software tools will be shown. The demonstrator is mainly based on on existing tools and methodologies. The MS-RTI and the MSE integration infrastructure are anchors for the different components of the MMP (Figure 1).

The MSE integration infrastructure provides the mechanism for interfacing software agents that participate in the Manufacturing System Engineering Process. Its purpose is to distribute information generated by any component to all components that have an interest in that information. In this way the MSE integration infrastructure supports the information flow within the MSE process. Additionally the information manager supports an object oriented data retrieval to access the mission repository. This repository holds a current state of all data exchanged within the MSE process.

Before an MSE process can be started a project has to be defined. The project agent supports management of the Manufacturing System Engineering Integration Infrastructure by monitoring the life cycle of a particular project from inception, through all stages of design, implementation, and perhaps, evaluation of outcome. It will also ensure that all information pertaining to a particular project is accessible to all activities in the project and maintained separately from other project information. The term “project” indicates a set of data, for which one engineer (or a small group of engineers) is responsible. This is, typically, be a small subset of all the information available to the MMP, and some projects may share common information. The Project Agent can be useful in four main areas:

- Configuration of a new project (Project administration)
- Definition of standard project structures which can be adapted to a special project
- Use of standard project structures
- Monitoring of the project progress.

During the MSE process conflicts can arise between the different involved agents. This is the point when the MSE moderator starts his part of the game. The MSE moderator observes the activities of other components, and in particular the decisions applied to the design through the other components. Its purpose is to identify the points at which one agent generates information which might conflict with the interests of other MISSION tools; it then signals the conflict to the agents involved, and orchestrates an dialogue between the components until the conflict is resolved.

A central structure within the MMP approach is the template library. Together with the MISSION repository it allows a flexible definition of classes, attributes and objects. The template library is mainly supported by the simulation manager. Both will be described in more detail within the next chapters.

2 TEMPLATE LIBRARY

Within the MSE process, the template library is mainly used as a flexible knowledge base. For this purpose, attributes can be defined depending on the used design agents. Some attributes are pre-defined according to the requirements of the MMP or according to user requirements. The user has the chance to add templates and attributes of templates. Additionally, the user can use objects of these templates within the MSE process. For example, the developer may decide to use a special AGV. If an AGV template already exists as a subclass of the "transport" template the developer can use it. Otherwise the developer can create a new template class and add the necessary attributes, as far as they are not derived from the transport class. Now the developer can create an object of the AGV class and set the values. Furthermore, the AGV object can be used during the design process. If a simulation model is available for the AGV class and the relevant attributes for the simulation were set, then this AGV object can be used directly within a simulation scenario.

Concerning the simulation process, the template library contains a reference to simulation models for each application template. The simulation model implements the content of the application template. Each model has to be able to execute this partial simulation process by itself. Furthermore, the notation "application template" will be used for templates directly applicable within a simulation scenario.

Additional descriptions are necessary to execute different simulation models in a distributed simulation scenario for a template. It is important to describe the relations between the different templates. In order to enable an automatic generation of simulations based on a given simulation scenario, an object scheme required. This scheme describes the objects exchanged with other simulation models ("Common Exchange Object Model"). Furthermore, the exchanged objects can be divided in two types:

1. Objects including necessary information but not associated with other objects
2. Objects including necessary information and associated with objects of other simulation models (e.g. the port of a warehouse is associated with a net node of a transport system).

In order to avoid modelling all the exchanged objects for each single simulation scenario anew , the template library includes an additional structure of common exchanged objects (Figure 2). This structure can be expanded by the template library administrator following the template library rules. The rules include mainly:

- Clear and unique name space for objects
- Clear and unique name space for attributes within one inheritance tree
- Unique semantic structures of each object class
- Unique semantic structures of each attribute within one object class.

This approach will allow an automatic generation of a common exchange object model for a user defined simulation scenario. Furthermore, it becomes possible to configure a simulation scenario including an automatic generation of HLA-RTI-FED files and federate configuration files.

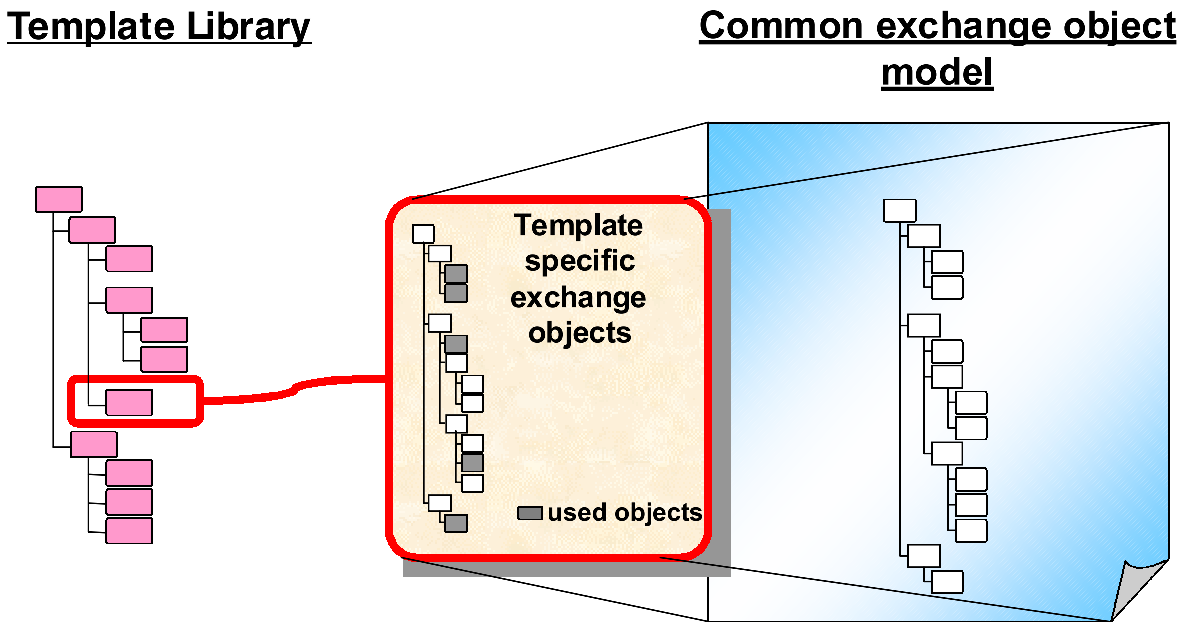

Template Library
Common exchange object model
Template specific exchange objects
used objects

Figure 2: Relation Between a Single Template of the Template Library and the Common Exchange Object Model

An additional part of an application template is the graphical representation of the template. 2D and 3D representations can be used for the graphical representation of the template. A possible format for the graphical representation is VRML. VRML allows a modular structure of objects within a 3D animation.

Summarizing, a template contains (Figure 3):

- References to simulation models
- Description of exchanged objects
- Visualization of the template
- Parameter descriptions of the template.

Now the templates can be used for the definition of simulation scenarios by deriving building blocks from the template library.

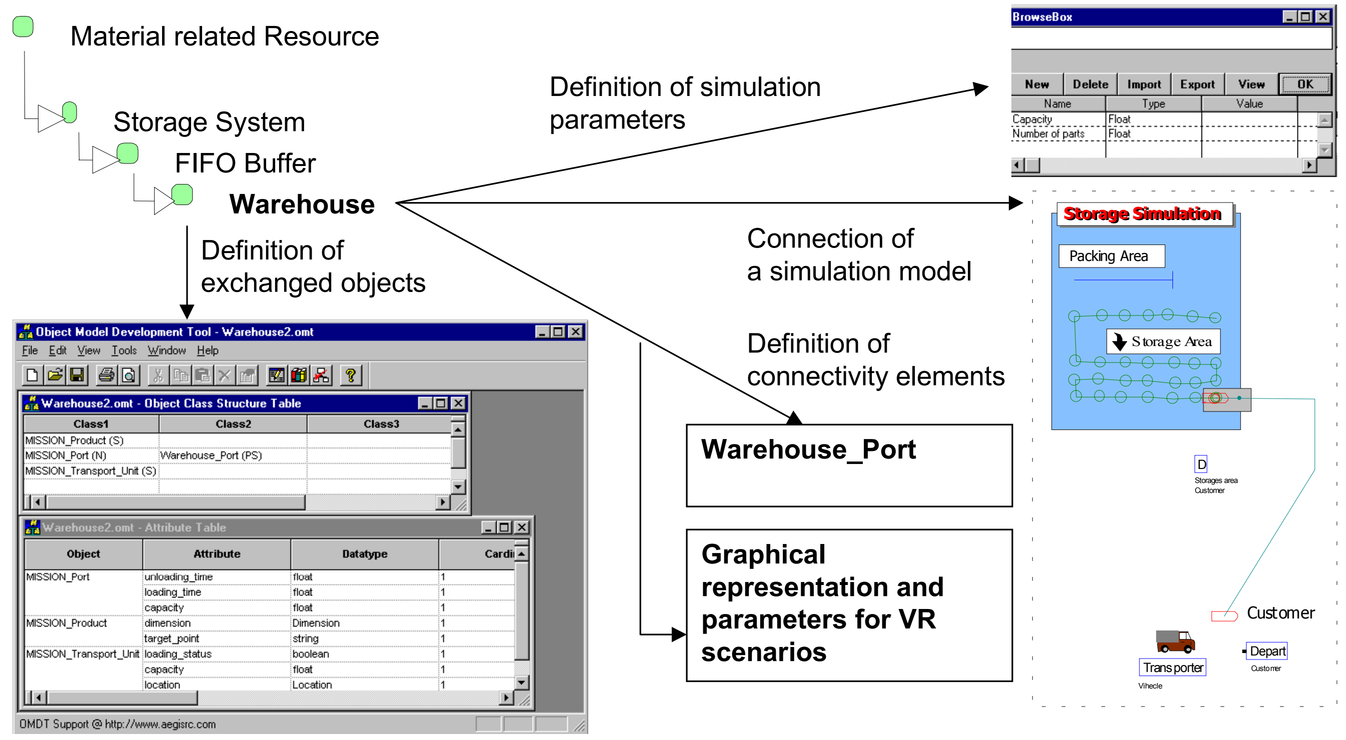
2.1 Building Blocks

A template can be seen as a building block of a huge simulation scenario, or as a predefined component of a manufacturing design like a transport system, a warehouse, etc.. Therefore, a building block in this context is defined as an object derived from a template of the template library.

A simulation study applying the building blocks consists of the following steps:

1. Selection of suitable templates
2. Insertion of a building block which is derived from the selected templates into the simulation scenario
3. Parameterisation of the building blocks
4. Selecting the relevant region of the scenario for the simulation
5. Consistency check and if necessary supplementation of the selected simulation scenario
6. Execution of the simulation scenario.

The template library supports the selection of suitable templates by a search structure and the special indication of application templates. Highlighting the templates already used within the current MSE project will furthermore speed up the selection process. The insertion of the building block into the simulation scenario can be done during the MSE process in order to identify the required processes and resources for the manufacturing system (MS). Then the model and especially the building blocks derived from the templates can be used as an information base for the whole design process. The information about the objects within the manufacturing process model becomes more and more supplemented during the MSE process.


Material related Resource
Storage System
FIFO Buffer
Warehouse
Definition of simulation parameters
Connection of a simulation model
Definition of connectivity elements
Definition of exchanged objects
Warehouse_Port
Graphical representation and parameters for VR scenarios
BrowseBox
New
Delete
Import
Export
View
OK
Storage Simulation
Packing Area
Storage Area
Storages area
Customer
Customer
Transporter
Vihecle
Depart
Customer
Object Model Development Tool - Warehouse2.omt
File
Edit
View
Tools
Window
Help
Warehouse2.omt - Object Class Structure Table
Warehouse2.omt - Attribute Table
OMDT Support @ http://www.aegisrc.com

Name	Type	Value
Capacity	Float	
Number of parts	Float	

Class1	Class2	Class3
MISSION_Product (S)		
MISSION_Port (N)	Warehouse_Port (PS)	
MISSION_Transport_Unit (S)		

Object	Attribute	Datatype	Cardi
MISSION_Port	unloading_time	float	1
	loading_time	float	1
	capacity	float	1
MISSION_Product	dimension	Dimension	1
	target_point	string	1
MISSION_Transport_Unit	loading_status	boolean	1
	capacity	float	1
	location	Location	1

Figure 3: Content of a Template Within the Template Library

In the next step this manufacturing process model is the base of the simulation scenario. A region of the model or the whole model can be selected as simulation scenario. Furthermore, a completeness and consistency check is started to identify missing parameter settings, missing simulation models and so on. For each building block, the user has to set the missing parameters, as far as they are required by the simulation model. For this task some supporting wizards are proposed (Figure 4). An important task is to associate the connectivity elements of the different building blocks. Each building block has a set of connectivity elements. It is necessary to define the input and output of the building blocks via connectivity elements defined for the building blocks. For example, the user has to associate the output of a processing line with a station of a transport system. When the simulation scenario is complete and consistent, the user can execute it. Three steps will be done during the initialisation of the execution:

1. The generation of a common information model and the generation of the configuration files
2. The generation of the runtime environment of the simulation scenario
3. The start of the runtime scenario.

The common information model of exchanged objects can be generated easily on the base of the exchange object model being part of each template, if the template library rules are adhered to. The different exchange object classes can be integrated into one common class structure based on the common exchange object model of the template (Figure 5). This is done by the following steps:

1. Modelling of a simulation scenario by application of building blocks derived from the templates
2. Supplementation or setting of the necessary parameters of the building blocks for simulation
3. Merging of the exchange object classes and generation of a common structure for the simulation scenario
4. On this base, automatic generation of a FED file to configure the RTI
5. Generation of the federate configuration files from the model of the simulation scenario
6. Conduction of experiments by simply changing the parameter settings.

For an experiment, the common information model will not be changed. Only the federate configuration files have to be adapted to the experiment.

The generation of the runtime environment of the simulation scenario is based on the previously generated federate configuration files. Each simulation model involved in the simulation scenario gets its own federate configuration file including all information necessary to configure the simulator interface for this simulation model.

The next stepis the consistent combination of simulator, simulation model and federate configuration file.

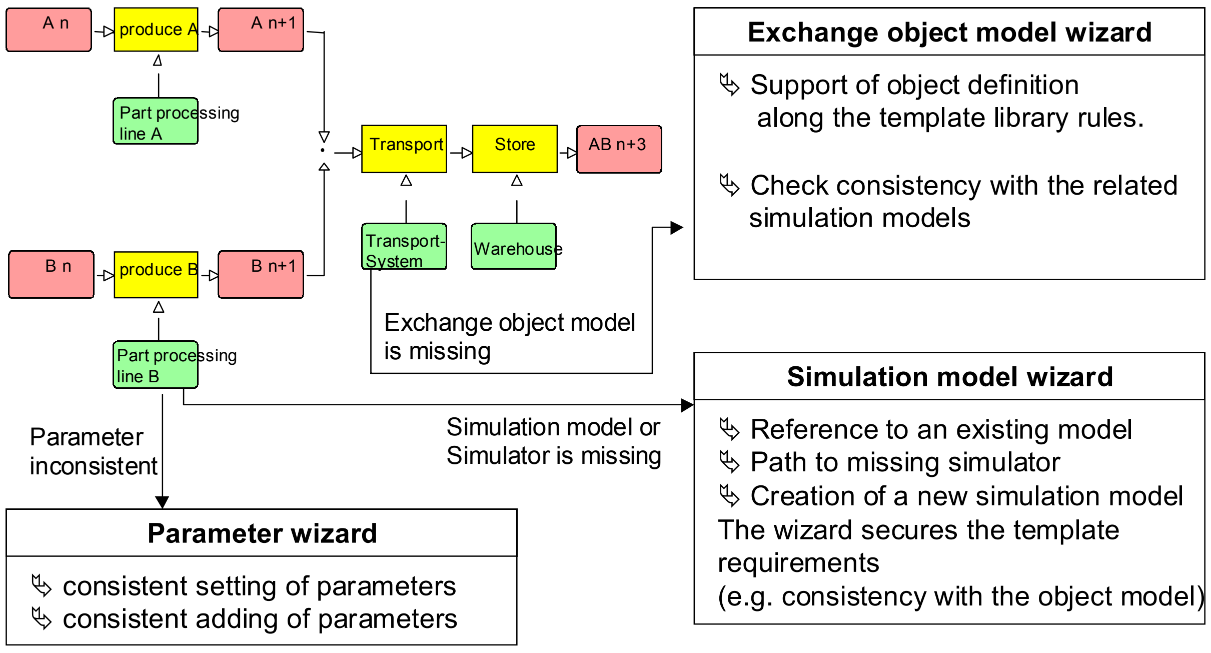

A n
produce A
A n+1
Part processing line A
Transport
Store
AB n+3
Transport-System
Warehouse
B n
produce B
B n+1
Part processing line B
Exchange object model is missing
Simulation model or Simulator is missing
Parameter inconsistent
Exchange object model wizard
Support of object definition along the template library rules.
Check consistency with the related simulation models
Simulation model wizard
Reference to an existing model
Path to missing simulator
Creation of a new simulation model
The wizard secures the template requirements (e.g. consistency with the object model)
Parameter wizard
consistent setting of parameters
consistent adding of parameters

Figure 4: Simulation Scenario Check

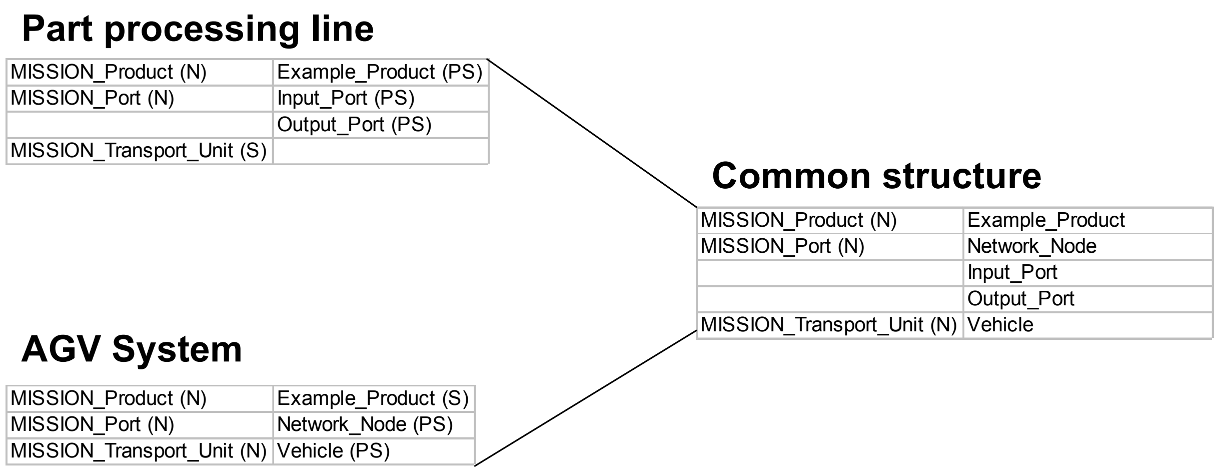


Figure 5: Merge of Exchange Object Classes of Different Templates

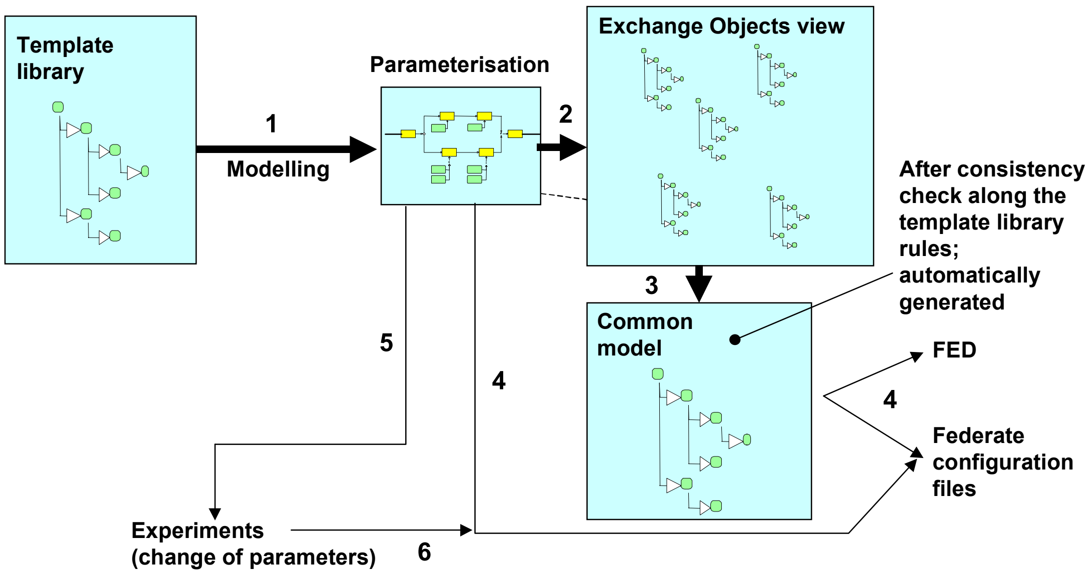

Template library
1
Modelling
Parameterisation
2
Exchange Objects view
After consistency check along the template library rules; automatically generated
3
Common model
5
4
FED
4
Federate configuration files
Experiments (change of parameters)
6

Figure 6: Generation of the Simulation Scenario Runtime Configuration Files

When this step is completed, the HLA-RTI can be started using the previously generated FED file. Now the different components of the simulation scenario runtime environment will be executed. Figure 7 shows an Adapter to the RTI that allows an easier connection of software tools. The adapter supports the retrieval of class and attribute information described within the federate configuration file. Furthermore it supports the association between class and attribute descriptions within the federate configuration file and the class and attribute names within the FED file. Therefore, it will be easy to retrieve information for an attribute name like value type, default values of attributes, etc.

2.2 Design of a Template

The approach described is based on the template definition. The consistent combination of template attributes, parameters of simulation models and the exchange-object-model is very important.

The simulation model can be built independently from the template, adhering the template description, only. Afterwards, the simulation model can be connected with the template. The parameters of the simulation model are connected with attributes of the template. These attributes have to be marked as attributes relevant for the simulation (e.g. for consistency checks). At the end, it is necessary

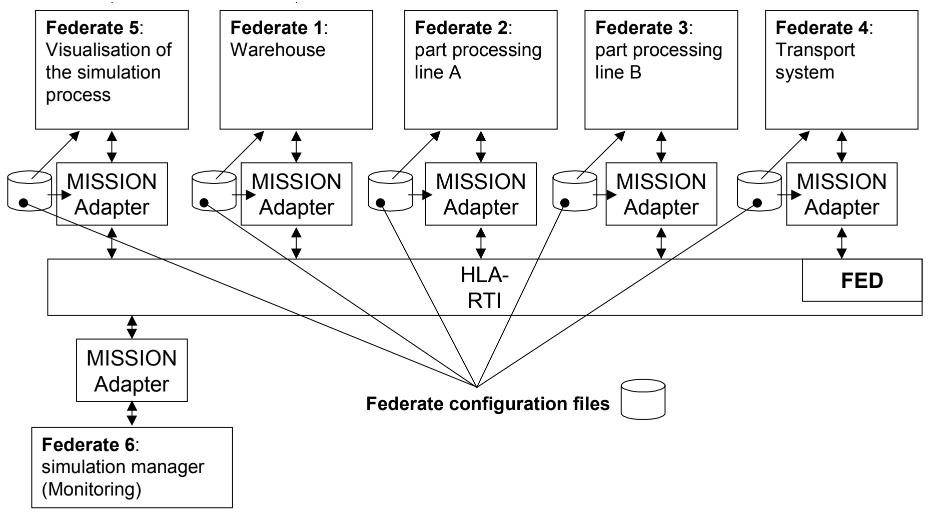

Federate 5: Visualisation of the simulation process
Federate 1: Warehouse
Federate 2: part processing line A
Federate 3: part processing line B
Federate 4: Transport system
MISSION Adapter
HLA-RTI
FED
MISSION Adapter
Federate configuration files
Federate 6: simulation manager (Monitoring)

Figure 7: Execution of a Simulation Scenario

that each parameter of the simulation model has a clear relation to a template attribute. The description of the exchanged objects and the integration into the common-exchange object model is more challenging.

For the simulation model, it has to be decided which objects are necessary for the communication with the distributed simulation environment. Furthermore, attention to the input and output segments as well as the input and output objects is necessary. An output segment of a processing line could be the output buffer. Within a simulation scenario this output segment will be associated with an input segment of an other simulation model, e.g. a transport system. In addition, the objects which pass those segments have to be described, also. In the remains of this paper, these segments are named as connectivity elements.

The connectivity element classes and attributes are a part of the exchange object model. Connectivity element objects and the connections between these objects are described within the federate configuration files.

Instead of a detailed description of a template, it is possible to compose a template of other templates (Figure 8). This allows a modular structure of the templates and of the components of the distributed simulation. For example, a transport system may be composed of a route planning system, a network simulation and a vehicle simulation (Figure 8). The hierarchical inheritance structure of the template library can support wizards to recognize the necessary elements of a more detailed template. This works because on a higher level the structure can be described in more generic terms. If, e.g., the description of a transport system includes a vehicle as an essential part, the wizard can ask for a vehicle as part of an AGV system.

The behaviour description of the template supports the retrieval of a suitable template and secures a similar content of the simulation models behind the template. The description could be done using a natural language. However, it is much better to use a formal language based on XML. The advantage of a formal language is the chance to load this description directly into a simulator. That does not result necessarily in an automatic generation of simulation models, but it provides a base to build such models faster.

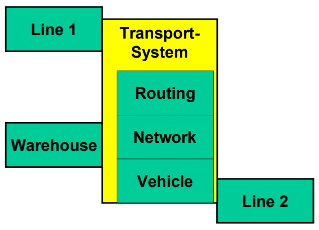


Figure 8: Bill of Material Structure for the Template “Transport System”

2.3 Federate Configuration File

The federate configuration (FC) file will close the gap between the HLA-RTI-FED-file and the interface definition of simulation models (federates). The description of the information within a federate configuration file (Table 1) will be defined via XML.

The definition of the object and attribute names within the FC files according to the FED file are necessary, because otherwise a federate does not recognize the names of the involved exchange objects. The type information is important for a federate to extract information from the objects handled by the RTI. Default values can be defined to substitute more detailed simulation models. For example, the speed attribute of a vehicle is defined as an attribute of an exchange object and set by an engine simulator. The default setting of the speed attribute will allow to leave out the engine simulator. On the same way,

Table 1: Federation Execution Definition File (FED) vs. Federate Configuration Files

Content of the Federation execution definition (FED) file used by the HLA-RTI	Content of the federate configuration files used for simulation interface configuration
• necessary class names of exchanged object classes • attribute names	• class names of exchanged objects • attribute descriptions of the classes including: + names + type information + default values • initial values of attributes or parameters • parameter settings (e.g. max vehicle speed) • Information about sequences (process flow) • Information about output and input relations

initial values allow to define a start status of a simulation model.

The setting of simulation parameters can be used to tune the simulation models. For example, two part processing lines A and B are used as building blocks. Then the performance of the processing lines can be defined for each building block by setting the performance parameters. Therefore, the same model can be used in different incarnations.

An additional requirement is the description of process sequences and input/output relations. This allows the configuration of simulators with relations between the different simulation models.

The content of the FC files will be proved during the implementation of the MMP. In addition, it will be adapted to the requirements of the interfaces to commercial simulation systems.

3 SIMULATION MANAGEMENT

The simulation manager mainly supports the template library. Most functions of the simulation manager are based on the template library approach. It includes views to manipulate template classes, objects, building blocks and manufacturing system (MS) process models. Furthermore, it supports the design and execution of simulation scenarios based on the template library mechanism. The MS process model is created during the MSE process and includes all objects needed during the MS process. Especially, machines, transport systems and storages with their attributes are specified within the process model. This MS process model can be used later to select a part of the MS process for the simulation.

The Simulation Manager uses the information from the template library and the information collected during the MSE process to construct a distributed simulation scenario. It further supports the design of simulation experiments, the creation of FEDEX (Federation Execution) and it executes the design. The results of experiments are predictions of the manufacturing process performance, which the manager makes available, through the Manufacturing System Engineering Integration Infrastructure to other components (Jaekel and Arroyo 2000).

The Simulation Manager is needed to start up, initialize and manage the execution of the different simulations in a distributed simulation execution. Summarizing the description given above the Simulation Manager consists of the following seven parts:

1. Template library editor
2. Modeling tool for the design of the whole simulation scenario using elements derived from the template library
3. Supplementation of elements in user and standard Template Libraries
4. Configuration of the runtime simulation scenarios
5. Execution and monitoring of the whole simulation scenario
6. Bridge between the MSE Integration infrastructure and MS-RTI
7. Experimenter (optimisation by variation of simulation parameters)

4 ADVANTAGES OF THE TEMPLATE LIBRARY APPROACH

Within different simulation systems mechanisms like templates, modules or classes are already available (e.g. Arena, eMPlant). These mechanisms have been further improved by reference models (Mertins et al. 1998).

The disadvantage today is that these methodologies are specific for one simulator and currently, these methodologies are not compatible for different simulators. The advantage of the template library is that this approach includes a mechanism to reuse simulation models from different simulators in different simulation scenarios. Furthermore, it allows distributed simulations between different enterprises without the necessity for these enterprises to use the same simulator.

HLA allows the connection of different simulation models via a runtime infrastructure (RTI) called federation. However, it is still a disadvantage, that each federation needs a hard programming of the interface between the federates and the RTI. This is one of the reasons, why the HLA is not used more frequently within the civil area.

The advantage of the MISSION template library approach is that the simulation manager, as a part of the template library implementation, will implement the generation of configuration files for federates participating in the simulation scenario to avoid intense programming.

ACKNOWLEDGMENTS

The European Module of the MISSION project is carried out with financial contribution of the European Commission under the specific RTD Programme, Esprit Project 29 656. Partners within this European Module are Bosch (D), CASA (E), Sisteplant (E), vr-architects (A), ProSTEP (D), Fraunhofer IPK (D) and Loughborough University (UK).

REFERENCES

McLean, C., F. Riddick, and S. Leong. 2000. Architecture for Modeling and Simulation of Global Distributed Enterprises. . In *The New Simulation in Production and Logistics*, ed., Mertins, K.; Rabe, M. 365-374. IPK, Berlin.

Mertins, K., M. Rabe, and P. Rieger. 1998. Taking Advantage of Process Oriented Reference Models for Setting up Federations for Distributed Simulation in HLA Environments. In *12th European Simulation Multiconference ESM'98, Manchester* ed., Zobel, R.; Moeller, D. 259-263.

Mertins, K., M. Rabe, and O. Krause. 1998a. Modeling for Planning and Operation of Global Distributed Enterprises. *IX. Internationales Produktionstechnisches Kolloquium PTK'98, Berlin*. 363-367.

Jäkel, F.-W., and J.S. Arroyo Pinedo. 2000. Development of a Demonstrator for Modelling and Simulation of Global Distributed Enterprises. In *The New Simulation in Production and Logistics*, ed., Mertins, K.; Rabe, M. 375-384. IPK, Berlin.

AUTHOR BIOGRAPHIES

MARKUS RABE, born 1961, is the head of the production planning department of IPK. He is responsible for business process planning, factory planning and simulation and head of the Berlin Demonstration Center for Simulation in Production and Logistics. Markus Rabe is active in the German Simulation Society ASIM, has been member in several conference programme committees and was chair of the conference "Simulation in Production and Logistics" in 1998 and 2000. More than 80 publications and editions report from his work. His email and web adresses are `<markus.rabe@ipk.fhg.de>` and `<http://www-plt.ipk.fhg.de/unternehmensplanung/index.htm>`.

FRANK-WALTER JÄKEL, born 1959 studied vehicle design in Hamburg and computer science at the Technical University of Berlin. Since 1989 he is working at the Corporate Management division of IPK-Berlin. Frank-Walter Jäkel is involved in various research and consulting projects. His main points of interest in consulting are business process modelling, analysis and optimisation, software development, adaptation and use of business process modelling methods in different domains. His email address is `<Frank-Walter.Jaekel@ipk.fhg.de>`.