Building an HLA Federated Interoperable Framework from Legacy Information Systems

Zhiying Tu, Gregory Zacharewicz, David Chen

IMS-LAPS / GRAI, Université de Bordeaux, TALENCE cedex, France

*Zhiying Tu. Email: zhiying.tu@ims-bordeaux.fr, *Gregory Zacharewicz. Email: gregory.zacharewicz@ims-bordeaux.fr, *David Chen. Email: david.chen@ims-bordeaux.fr

(Received XX Month Year; final version received XX Month Year)

ABSTRACT
This paper aims at improving the re-implementation of existing information systems when they are called to be involved in a system of systems, i.e. a federation of enterprise information systems that interoperate. The idea is reusing the local experiences coming from the previous development of the existing information system with the process of Model Discovery. To avoid re-developing the entire system when the enterprise needs to cooperate with others, this approach proposes to create local interfaces to code and decode information. The interfaces are instantiated by using models discovered. The interfaces are developed in accordance with the HLA standard that proposes message interoperability and synchronisation mechanisms among distributed systems. First, the authors recall the strong points of MDA/ADM methodologies for model transformation from conceptual level to implementation. Besides that they also recall the HLA standard. From this, a MDA&HLA framework is proposed to implement distributed enterprise components from the conceptual level of the federated enterprise interoperability approach. In addition, the authors add a model reversal methodology to the framework to guide re-implementation of legacy information systems, when called to achieve interoperability with other systems. To extend the scope of the approach, implemented web services are combined with HLA in order to facilitate the use of HLA in large distributed execution. The paper ends with an implementation example for validating the approach.

Keywords: interoperability; HLA; model reversal; HLA evolved

1. Introduction

Nowadays, enterprise collaboration becomes more and more important because of the globalised economic context. The competitiveness of enterprises depends not only on its internal productivity and performance, but also on its ability to collaborate with others. This necessity leads to the development of a new concept called interoperability that allows improving collaborations between enterprises. Therefore, more and more networked enterprises are developed for which enterprise interoperability is considered as one of the most suitable solution compared to total enterprise integration. Nevertheless, the problem starts from the fact that each enterprise possesses its own, frequently specific, information system.

This paper proposes a solution to implement a federated framework for enterprise interoperability. This goal is achieved by contributing to a harmonized and reversible development framework for Interoperable Distributed Enterprise Information Systems and Applications. Previous works from (Zacharewicz et al., 2008) & (Tu et al., 2010) have proposed enterprise interoperability based on multi agent/HLA (High Level Architecture) systems and a unified reversible life cycle for interoperable enterprise distributed information systems development. These authors demonstrated that between enterprises which participate in cooperative projects, information exchange is huge but can be facilitated through a HLA based distributed execution platform. Indeed, HLA proposes to reuse existing systems without recoding them but rather by addition of interfaces layers that transform and synchronize exchanged data. This paper intends to harmonize both the MDA (Model Driven Architecture) and the HLA development process to form an accelerated development life-cycle by reusing both the information and the behaviour of the legacy systems. In detail, Model Reverse Engineering is used to discover quickly the enterprises' knowledge, to reverse the information system of enterprises backwards to models, and thereby solve the interoperability problems without rebuilding the information systems from scratch (Jouault et al., 2009). Furthermore, in order to improve flexibility and compatibility of the HLA based communication platform with web 2.0 functions, a "web-services" federate is created to achieve a web-enabled RTI (Run Time Infrastructure) according to the last release of HLA evolved IEEE 1516-2010 (IEEE 2010).

The paper is structured as follows. It starts out with a survey of related work in the area of MDA, MDI, Model Reverse Engineering, HLA and Web services in chapter 2. It is followed by a description of background, presents the problems known to be solved in chapter 3. Then, it describes the life-cycle development framework and relates scenario of use (chapter 4). After that, it explains the methodology separately in three parts, harmonization of MDA and HLA FEDEP, model reverse engineering and HLA Evolved Web services in chapter 5. At last, a simulation case study is illustrated in chapter 6.

2. Technical recall

2.1 Enterprise Interoperability Framework

The Enterprise Interoperability Framework was initially proposed within the INTEROP NoE and now in the process of standardization as CEN/ISO 11354-1 (Chen et al., 2007). The Framework defines three basic dimensions for Enterprise Interoperability as follows (see figure 1):

Figure 1 Enterprise Interoperability Framework

- "*Interoperability concerns*" which identifies the area of interoperation that may take place at various levels of the enterprise (data, service, process, business), i.e. the level at which the interoperation occurs.
- "*Interoperability barriers*" which identifies various obstacles to interoperability in three categories (conceptual, technological, and organizational), i.e. the type of obstacle to interoperability.
- "*Interoperability approaches*" which represents the different ways in which barriers can be removed (integrated, unified, and federated).

The first two dimensions, Interoperability concerns and Interoperability barriers, constitute the problem space of enterprise interoperability. All the three dimensions constitute the solution space of enterprise interoperability. The objective of this framework is to tackle interoperability problems through the identification of barriers that prevent interoperability to occur.

2.2 Model driven Architecture

The life cycle development methodology studied firstly is Model Driven Architecture (MDA). This methodology has been defined and adopted by the Object Management Group (OMG) in 2001, (updated in OMG 2003). It is designed to promote the use of models and their transformations to consider and implement different systems. It is based on an architecture defining four levels, which go from general considerations (conceptual level) to specific ones (implementation level).

- *CIM Level* (Computation Independent Model): focusing on the whole system and its environment. It is also named "domain model". It describes all work field models (functional, organisational, decisional, process, etc.) of the system with a vision independent from implementation.

- *PIM Level* (Platform Independent Model): modeling the sub-set of the system that will be implemented.
- *PSM Level* (Platform Specific Model): taking into account the specificities related to the development platform.
- Coding Level: last level, consisting in coding enterprises applications (ESA: Enterprise Software Application).

To complete this description, a Platform Description Model used for the transformation between PIM level and PSM level is added to these four kinds of models corresponding to the four abstraction levels.

2.3 MDI framework

The approach "Model Driven Interoperability" (MDI) considers interoperability problems at the enterprise model level instead of only at the coding level.

These concepts were realized in the Task Group 2 (TG2) of INTEROP-NoE project by defining an approach inspired by the OMG MDA concepts (Bourey 2007). The goal of MDI is to tackle the interoperability problems at each abstraction level defined in MDA and to use model transformation technique to link both vertically the different levels of the MDA abstraction and horizontally the corresponding models of the systems to interoperate. The main goal of MDI, based on model transformation, is to allow a complete follow-up from the expression of requirements to the coding of solutions and also to provide a greater flexibility thanks to the automation of these transformations.

In the context of TG2, experimentations have been realized and in particular the feasibility study to transform GRAI Methodology (Chen 1997) (Doumeingts 2001) Models to UML models between CIM and PIM levels (Bourey 2007). These works are complemented by additional works realized in the context of ATHENA to define UML profiles to take into account also the Service Oriented Architectures (SOA) at the PIM level (Gorka 2007). These results have been complemented by results presented by (Touzi 2007) who has proposed an interoperability transformations method from BPMN to UML in the context of SOA.

Nevertheless, the soundness of the methodology has been demonstrated, but no full industrial scale validation has been yet achieved. Some projects have especially carried demonstrate these concepts in an industrial real world significant application. The different methodological propositions are tested and refined by focusing on models and their interoperability. They consist in particular of ways to improve the flexibility of the MDI transformation process and in obtaining dynamic interoperability in the context of the federated approach.

2.4 MDA & Model Reverse Engineering

MDA is well-known for promoting the use of models and their transformations to design and implement different information systems. After MDA became an important change in software development, OMG launched another research activity leading to what was later called Architecture Driven Modernization (ADM) (OMG 2008).

The basic idea proposed in the MDA approach is to translate from an abstract platform-independent model (PIM) expressed in UML into a more concrete platform-specific model (PSM) from which the code still needs to be generated (OMG 2003). Reversing the MDA lifecycle, ADM is discovering models from the coding level of legacy information system, such as UML models, Knowledge Discovery Meta-model (KDM) and Abstract Syntax Tree Meta-model (ASTM). KDM and ASTM are aimed to satisfy someone interested in discover more specific models from a legacy system (OMG 2008).

KDM is a meta-model for representing existing software assets and their associations, as well as relationships among the function models in the system. It also describes the operation environments. It can insure the interoperability among the existing systems, make the data exchange among different vendor tools easier.

2.5 High Level Architecture

The High Level Architecture (HLA) is a software architecture specification that defines how to create a global software execution composed of distributed simulations and software applications. This standard was originally introduced by the Defence Modelling and Simulation Office (DMSO) of the US Department Of Defence (DOD). The original goal was reuse and interoperability of military applications, simulations and sensors.

In HLA, every participating application is called “federate”. A federate interacts with other federates within a HLA federation, which is in fact a group of federates. The HLA set of definitions brought about the creation of the standard 1.3 in 1996, which evolved to HLA 1516 in 2000 (IEEE 2000).

The interface specification of HLA describes how to communicate within the federation through the implementation of HLA specification: the Run Time Infrastructure (RTI). Federates interact using services proposed by the RTI. They can notably “Publish” to inform about an intention to send information to the federation and “Subscribe” to reflect some information created and updated by other federates. The information exchanged in HLA is represented in the form of classical object class oriented programming. The two kinds of object exchanged in HLA are Object Class and Interaction Class. Object class contains object-oriented data shared in the federation that persists during the run time; Interaction class data are just sent and received information between federates. These objects are implemented within XML format. More details on RTI services and information distributed in HLA are presented in (IEEE 2000). In order to benefit from the Web Services such as, the support for numerous newer and older languages and operating systems as well as the ease of deployment across wide area networks, HLA evolved IEEE 1516^{TM}-2010 was publish in August, 2010 (IEEE 2010).

The FEDEP (Federation Development and Execution Process) describes a high-level framework for the development and execution of HLA federation. FEDEP uses the seven-step process to guide the development of the simulation system through phases of (1) requirements, (2) conceptual modelling, (3) design, (4) software development, (5) integration, (6) execution and (7) evaluation (IEEE 2003). It has been recently integrated into the more general DSEEP (Distributed Simulation Engineering and Execution Process) framework (IEEE 2011).

2.6 Web Services

Web Services has achieved a great success in the business domain, which stems from the good characteristics of the technology itself, is widely recognized by enterprises and business organizations and provides effective support for the open source community (Richardson 2007).

The data exchange via Web services is based on open standards such as HTTP and XML and is not associated with any particular vendor, operating system and programming language, which makes Web services platforms with good vendor neutrality. Coarse-grained business functions can also be packaged for the Web service platform and can then be discovered by potential consumers.

At the same time, business-to-Web service is widely recognized. Microsoft, IBM and Sun and other leading manufacturers as well as Apache and other open source organizations support it. World Wide Web and other standardization organizations with active participation in the Web services technology provide for great maturity and popularity of the organization advantage.

3. Problem Statement

The enterprise interoperability framework (figure 1) allows identifying the barriers preventing enterprise interoperability. It classifies the barriers in three categories: conceptual, organizational and technological. The three crosses in figure 2 represent three concrete typical barriers in enterprise interoperability. For example, as a conceptual barrier, different IT systems can use different terms to name a "car". As a technological barrier, different IT systems can use different data format for message transmission. Finally, as an organizational barrier, diverse organization structure can exist in different enterprises making difficult to identify the responsible counterpart. The solution showed in this paper more focus on the overcoming those barriers in data and service concerns.

Figure 2. Problems to solve

Another problem can occur when one enterprise intents to participate in an existing cooperative project and has to connect to other heterogeneous Information Systems. How can this connection be established efficiently, correctly, and with low cost? The enterprise interoperability frameworks aim to answer those questions. As Figure 1 illustrates, the enterprise interoperability framework recaps three approaches to achieve enterprise interoperability. The difference between those three approaches lies in what can be called the "coupling coefficient", which means what is shared as a common format among all the models or system components. Historically, the pioneer interoperability approach (Integrated) has used a common format for all the models. Then, the Unified approach proposed to keep the common format only at meta-level at the intersection between the two systems for mapping exchanged data. Thereby enterprises can keep their local private formats. While, more recently, the federated approach proposed to work with no predefine common format. It uses dynamical adjustment and accommodation. This paper proposes a federated approach to achieve the development of a more flexible and dynamic framework.

4. Overview of the HLA Federated Interoperable framework

Nowadays, many applications have been developed to support the implementation of interoperability solution. (Zacharewicz 2011) has reported several applications to establish interoperability between enterprises Information System in various industrial domains. Most of those platforms were designed to exchange data inside the enterprise using distributed simulation for routing and synchronizing the information management using HLA. However, those applications emphasize more on system integration. The structure of data exchanged is static. The flexibility and compatibility are not very satisfactory regarding web 2.0 technologies capabilities and requirements. The framework presented in this paper is focused on reusability of components, compatibility with web services platforms to be interfaced with the web, facilities for community joining and resigning, and a rapid development life cycle.

4.1 Framework description

Harmonized and Reversible development framework for HLA based interoperable application is the general description of this framework. In this title, there are four keywords, harmonized, reversible, HLA, and interoperable.

Figure 3 Harmonized and reversible development framework

Harmonized means this framework is a synthetic framework, which combines several techniques. Shown in figure 3 is a five step development life cycle that aligns MDA and HLA FEDEP. MDA is widely used for IS development, approachable, tightly bounding Unified Modelling Language (UML), Meta-Object Facility (MOF), and Common Warehouse Meta-model (CWM). HLA FEDEP is the standard of development and execution of HLA federation, and it is similar with the waterfall development but with a look-back test phase. MDA & FEDEP have similar steps. The motivation of their alignment is to combine the simulation world (with synchronisation, behaviour, events, etc) to the information system world (with web services, flexibility, data bases) skills in one system of systems. Distributed simulation is the engine that can orchestrate the ISs messages exchange. This framework uses WS to improve the performance of HLA, which is explained in section 5.3.

Reversible means that this framework uses model reverse engineering techniques to discover the model of data and behaviour from the legacy system. The model reverse engineering technique involvement allows avoiding completely rebuilding the legacy system for different cooperation. The objective is to accelerate the development and reduce the cost. As figure 3 illustrates, there are two arrows, which have opposite direction in the five step development life cycle. These two arrows represent two different scenarios of model reverse processes that discover models from the existing information system. Those two scenarios will be detailed in section 5.2.

HLA means that this framework is dedicated to the development of HLA compliant applications that benefit from HLA routing and synchronisation of information. This project uses an open source RTI, poRTIco (poRTIco, 2009). The reason of this choice is the motivation to initiate an open framework for a new released standard: HLA Evolved detailed in 6.1, to receive contributions from participant who are interested in this idea.

Interoperable means that this framework provides solutions for achieving enterprise IS more economic ways to interconnect. As mentioned in section 3, this framework aims to overcome the enterprise interoperability barriers and help to realize the short-lived ontology approach, which mentioned by (Zacharewicz 2009).

4.2 Scenario

A schema of the related scenario is shown in figure 4. It must be assumed that before enterprises start to launch a cooperative project all have their own system. Thus, the goal is to achieve the interoperability among those existing systems in this project context. The steps of this approach are presented in the following:

Figure 4. Scenario description

Step 1 (arrows numbered with “1”): model reverse engineering is used to discover the models from the legacy system. The model discovery is guided by the enterprises new requirements and interest. Then, these “rewinded” MDA models go down again along the alignment of MDA and HLA FEDEP. It means models are generated from code to PSM then PIM and CIM level. At each level of the MDA models the interoperability problem is tracked according to the principle of the MDI framework.

Step 2 (arrows numbered with “2”): a test of the final models obtained by model reverse engineering is carried out. After that, the correct models are transformed from CIM to code, and generate a Federate Interface, which can plug into the HLA platform and transmit the information with other companies’ information systems via RTI.

Step 3 (arrows numbered with “3”): if other enterprises want to join this cooperative project, they also need to follow the step 1 and step 2, to rewind their legacy system into MDA models, and remove the incompatible parts, then generate the Federate Interface, finally, synchronize with other systems.

5. Specification of the framework

5.1 Harmonization of MDA and FEDEP

HLA is a distributed simulation standard that specifies a Federation Development and Execution Process (FEDEP), a synchronization standard: Runtime Infrastructure Specification, one Data

Standards. However, up to now in order to take advantages of this, a profit tool should be used since HLA mainly benefits from the commercial developments. In addition HLA is mostly restricted to the distributed simulation domain. On one side, MDA is popular, widely recognized and it can easily be aligned with HLA FEDEP. This alignment can facilitate the construction of simulators and provide the standardized meta-models to this integration (Andreas 2002) (Shawn 2003) (Trbovich 2005). On the other side, from an interoperability point of view, most of the enterprises build their information system by using MDA, so it would be the first choice for overcoming the interoperability barriers (Ullberg 2007). Finally, the still confidential Model-Driven Interoperability (MDI) Framework provides a foundation, consisting of a set of reference models, for how to apply Model Driven Development (MDD) in software engineering disciplines in order to support the business interoperability needs of an enterprise (Elvesæter 2007).

5.1.1 Specification of the harmonization

In this section, a new five steps development lifecycle based on HLA FEDEP and MDA (as shown in figure 5) is proposed to reconstruct Information Systems. This new methodology aims to adopt the strong points from both HLA FEDEP and MDA (as shown in figure 3) while overcoming some weak points, to achieve more proper component reuse and rapid development.

Phase 1: *Domain requirement definition*, its main task is to collect sufficient and clear requirements from the stakeholders in order to define the objective of the system, to describe the environment of the system, the scenario of the system and the business process. All these definitions and descriptions have to be reasonable and understandable for all of the stakeholders. The CIM level of MDA has a task that is similar to both the Define Federation Objectives and the Develop Federation scenario together in HLA FEDEP. As a result, their alignment in this phase, is to convert the user requirements that are textual based, into a more visual model, such as the UML use case to derive the federation requirement.

Figure 5. Harmonization of MDA and HLA FEDEP

Phase 2: *Domain scenario systematization*, its main task is to refine the domain scenario and the business process captured in the first phase. It identifies and describes the entities involved in the scenario and business process. Then, it defines the relationships among entities and their behaviours, events for each entity, etc. This phase integrates the PIM level in MDA, which describes the operation of the system but doesn't address the detail platform information yet. It also integrates steps of the Perform Conceptual Analysis, Develop Federation Requirements and Select Federates in HLA FEDEP. That also defines and selects general participators of the federation, describes their relationship, behaviours and event in general.

Phase 3: *System model specialization*. In this phase, according to the technique chosen and the platform selected, the system needs to be refined, for instance, to refine federation and federate structure, to allocate functions and attributes, etc. Detailed design is carried out at this time. This phase integrates the following parts in MDA and FEDEP.

- The PSM level in MDA that is in the form of software and hardware manuals or even in an architect's head, is based on detailed platform models, for example, models expressed in UML and OCL, or UML, and stored in a MOF compliant repository.
- The Prepare federation design, Prepare plan, Develop FOM, and Establish federation agreement in FEDEP, which produce federate responsibilities, federation architecture, supporting tools, integration plan, VV&A plan, FOM, FED/FDD and time management, date management, distribution agreements, etc.

Phase 4: *System Implementation*, its task is to transfer the specific system model into code, to create the executable federation and executable federate. At this level, MDA has various transformation techniques from model to code. In the FEDEP, Implement Federate designs provide modified and/or new federates and their supporting databases. Implement Federation Infrastructure provides implemented federation infrastructure and modified RTI initialization data. Plan Execution and Integrate Federation provide execution environment description and integrated federation.

Phase 5: *Test*. Throughout the previous steps of the MDA and HLA FEDEP alignment process, testing is essential to ensure fidelity of the models. Testing phase includes the Test Federation, Execute Federation and Prepare Outputs, and Analyze Data and Evaluate Results in HLA FEDEP. Meanwhile, it also refers to the outputs from the previous steps, such as the original user requirement in the first step, and federation test criteria from second phase.

5.1.2 Harmonized federate structure

A schema of the expected result can be seen in figure 6. After this harmonization, a federate can be considered as a *converter*. A federate has two parts, one is "*Adapter*" and another is "*Plug-in*". *Adapter* includes *Enterprise Business Behaviour Interface*, which connects to the enterprise business process related to specific strategies and algorithms of different enterprises. *Plug-in* includes *Integration code*, which manages the interactions between the enterprise business behaviour interface and the RTI, providing an RTI independent API to the enterprise business behaviour interface, and a simulation independent API to the RTI services.

Figure 6. Harmonized federate structure

The objective of these abstractions is to ensure that the enterprise business behaviour remains decoupled from RTI services. After the harmonization, all federates have the same Integration code but different Enterprise Business Behaviour Interface. Any simulation related services required by the enterprise business behaviour interface are accessed via the integration code, rather than through direct interaction with the RTI. As the result, the integration code is a common component for all federates of the existing coordinators and also a reusable component for future coordinators. The enterprise business behaviour interface adapts the different legacy systems of different enterprises to HLA function calls. The implementation is based on specific strategies and algorithms of the different enterprises; it accomplishes connection between public data and private data treatment not shared within the federation.

5.2 Model Reverse Engineering

5.2.1 Specification of model reverse engineering

This section describes a brand-new process of model reverse engineering with different scenario constraints (see figure 7). The reverse process re-characterizes the legacy system in order to capitalize on the information and functions of the existing system, and make it easy for reusing those in a new HLA compliant system. This methodology assists in the HLA FEDEP / MDA alignment mentioned in previous sections, to achieve the rapid development of federation and/or federate based interoperation of the legacy IT systems.

Figure 7. Model Reversal Framework

There are two different arrows, which represent two different scenarios of the model reverse process, illustrated in figure 7.

(1) First scenario, when an enterprise intents to start exchanging information in a new cooperative project with other enterprises. In that case, the HLA federation has not been created yet, so it is proposed to reverse the code of the legacy information systems to the first definition phase (domain requirement definition). Then from top to bottom, the models are

generated for each phase of the harmonized framework, finally a federation and federate rapid development template is produced.

(2) Second scenario, if an enterprise intents to participate in an existing cooperative project and exchange data with other heterogeneous Information System. Thus, it is assumed an HLA federation has already been created. Here, according to the HLA FEDEP, federate starts to be considered from the second step (Perform Conceptual Analysis). Therefore it is not necessary to follow the reverse process to the first phase, and it can stop at the second phase (Domain scenario systematization). It only reuses the model of the existing federation to create the model for the federate of new participator related to the legacy system. Finally, the model of the existing federation and the new federate model are used to generate the code template for the new federate for rapid development.

5.2.2 Model reversal structure

A schema of the expected result can be seen in figure 8. This illustration is based on the approach of MoDisco (for Model Discovery).

In MoDisco principle (Jouault 2009), a model (Mi) in the modeling world is a representation of a system in the real world and the nature of the model (Mi) is defined by its meta-model (MMi). It means that model Mi conforms to its meta-model MMi, and every step is guided by a meta-model. The very first step of a model discovery process is always to define the meta-model corresponding to the models you want to discover. Then, the second step is about creating one or many discoverers, which is illustrated in the middle of figure 7. These discoverers extract necessary information from the system in order to build a model conforming to the previously defined meta-model. The way to create these discoverers is often manual but can also be semi-automatic.

Figure 8. Model reversal

In addition, in order to adapt the MoDisco principle to this approach, *constraints* are added before the *discoverers* are defined. As known, the legacy system consists of lots of sub-systems, which are always based on various kind of platforms and techniques, thus it is big and complex. If the objective is to reverse a complete legacy system, the project can be extremely huge and complicated. Even more, it departs from our motivation to focus only on the enterprise public data and on the behaviour of actions on these data (periodicity to update, priority, sequence of treatment, etc.). As a result, constraints aim to specify the target source, which means that the bound of model reversal must be defined before start reversing. The boundary must be defined based on the shared interest and each enterprise's confidential information. This boundary specification is recorded as a configuration file which can be read by *discoverers*. For these *constraints,* many modelling standards could be helpful, for example CIMOSA standard CEN/ISO 19440 (ISO19440 2008), and GRAI (Chen 1997).

According to the ongoing research, none of the software tools can fully reverse the legacy system from code to model. Most of the tools can rewind the code to a static model but without the complete dynamic model and some of them can only discover the data model from database. With the goal of developing HLA components that interface with legacy IS, the behaviour of the actions on the data needs to be discovered for implementing the mechanism for data access, the periodicity of update and the sequences of modifications.

In the approach, the choice fell on MoDisco, which is an Eclipse GMT (Generative Modeling Technologies) component for model-driven reverse engineering. The common infrastructure of MoDisco is inspired by the KDM (Knowledge Discovery Meta-model), which provides a comprehensive high-level view of application behaviour, structure, and data and corresponds to a coarse-grained representation of a given legacy portfolio. MoDisco is a powerful tool even though it needs further improvements. It provides different kinds of Domain-Specific Languages (associated to a meta-model) to describe the facet you want to discover from a given legacy. Furthermore MoDisco provides a toolbox that facilitates the building of model discoverers from the definition of the target meta-models. MoDisco can generate a static model from the code of program, and can also generate a dynamic model from the execution of program. These models are used to declare HLA objects and interactions and to describe the behaviour to modify and exchange the data of both the *federate adaptor* and the *plug-in* (introduced in figure 6). Main important information from legacy system is the proper sequence of action on the data and periodicity of access, the access rights and the format of the input/output data. However, the output of MoDisco is not a priori fully adaptive to the requirements of HLA component descriptions. The output is a XML based file in the form of a treelike structure in eclipse that is not obvious to be interpreted without an IT background. One future objective is, to provide a visualization interface based on MoDisco output to add constraints in order to select the output interesting for public data and behaviour to design HLA components within the platform.

5.3 HLA Evolved Web Services

In the last decade, HLA has succeeded in different aspects, especially in the areas of interoperability and reuse of simulation components. It has also passed from military to the civilian domain (SISO 2011). However, in the pace of the rapid technical change and the future of the standard, HLA has to face to many new challenges. In order to benefit from Web Services, HLA Evolved standard promotes and supports the concept of simulation as readily available service provided within and between enterprises.

5.3.1 Conceptual Specification of HLA Evolved Web Services

HLA has been originally developed within the defence simulation community, and was more focused on the simulation in Local Area Network or in Virtual Private Network. Thus it exposes some lack of flexibility, lack of integration with global business.

Meanwhile, Web Services have been developed within the commercial enterprise community, and have achieved a great success in the business domain, which attests for the fine characteristics of the technology. Also, they are widely recognized by enterprises and business organizations and receive effective support from the open source community.

As the result, Web Services are the perfect option to help HLA to overcome those disadvantages. This general idea of HLA/Web Services coupling has been proposed in the latest version of the HLA named “Evolved”. The principle is illustrated in figure 9 using a concrete case. Assuming that a cooperative project has been launched between enterprises, their information systems run within a HLA federation. During this project, another enterprise likes to join this project with different expectation, such as different cooperation time periods, different cooperation domains, different expected results from federation, etc. However, with the number of differences, a rebuilding of the federation is impossible. The solution is to add one more federate (Tu, 2009) proposed to create a *WebservicesFederate,* as shown in figure 9. This federate provides various services, different access permissions, and a common API from the existing federation. The candidate can use this common API and select the service he requires to generate his remote federate. He connects to the existing federation trough this federate with different authorization via the wide area network.

Figure 9. HLA Evolved Web Services

For example, in figure 9, two enterprises X and Y want to participate in an existing project. Enterprise X is a supplier and enterprise Y is a client who is interested in the final product of this project. Thus, enterprise X has to know the workflow that relates to its business, and synchronize its information with the other stakeholders, while, enterprise Y only asks for information from the federation. So, this enterprise does not have to synchronize with other systems. As a result, enterprise

X asks *WebservicesFederate* for services with an authority of synchronization with other federates, but enterprise Y only requires services with the lowest authority that can manage to listen to the federation. Finally, the components connected via the WAN have transparent access to the services (functions). They require to be connected to the existing federation via web services as if they are inside the LAN.

The figure 10 presents the technical transcription of the problem presented in Figure 9. It details the "bridge federate principle" that links the federation inside the LAN with federates in WAN. This bridge federate is both in the federation in the LAN and also in the WAN. It can be compared to gateway computers at the edge between two TCP/IP networks.

Figure 10. Architecture of HLA Evolved Web Services

6. HLA evolved WS general framework

6.1 HLA Evolved WS Proposed Architecture

6.1.1 Elected RTI

The technical architecture of HLA Evolved Web Services that is retained in this approach is illustrated in figure 11. An open source RTI, poRTIco (poRTIco 2009) has been chosen for implementation, even if it does not provide Web-RTI functionality. Actually, only one mature commercial RTI, pRTI, supports some Web-RTI functionality (Möller 2007). Even in this one not all IEEE 1516-2010 features are already developed. As mentioned, the current status of commercial developments and the aspiration to develop an open framework has guided the choice to poRTIco. But, to reach some HLA evolved requirements, new features have been added to poRTIco. A *WebservicesFederate* component has been implement as a *bridge*, who takes charge of providing web services, connecting and synchronizing federates outside the federation with federates inside the federation.

As mentioned in section 5.1, after the harmonization of MDA and HLA FEDEP, an integration code is provided with a RTI independent API for HLA Federates. This API can be reused and published as common API. So, the candidates can reuse this API and follow the second scenario of model reversal, mentioned in section 5.2, to generate their own Enterprise Business Behaviour Interface adapted to the common API. After that, a new federate outside the federation can send the information to the bridge via the Web services interface and be synchronized to the HLA federation.

6.1.2 WebservicesFederate design

A schema of *WebservicesFederate* design proposed in this paper is illustrated in figure 11. In this design, *WebservicesFederate* is a special federate, which is inside the Local area network (LAN) but not fully included in the federation. According to this specialty, *WebservicesFederate* is divided into two parts: one is *WebservicesBridge*, which is inside the federation; another is *WebServicesServer*, which is outside the federation but still inside the LAN. These two parts are connected by a socket. This design is customized for poRTIco RTI. As mentioned, this simulation is based on poRTIco RTI that doesn't support natively Web RTI functionality. In order to implement Web RTI functionality, the approach defines *WebservicesBridge* and *WebservicesServer* for *WebservicesFederate*.

Figure 11. Web services federate design

WebservicesServer: it is used to publish web services interface to potential customers outside the federation. It takes charge of monitoring and replying to the federate via web service. When this server receives the message from the federate outside federation, it generates a User Datagram Protocol (UDP) data package and sends it to *WebServicesBridge* by the socket connection.

WebservicesBridge: it uses to synchronize the message from the federate outside the federation with other federates inside the federation. This bridge transmits messages to federate inside the federation by RTI, but exchanges messages with the *WebservicesServer* by the socket connection. When web services federation establishes, this bridge launches a thread to monitor the events happening in the web service server.

Socket data package: in order to ensure the security of the federation, common federation attributes are encapsulated into the web service interface, which is published by the web services server. So, *WebservicesBridge* encodes the attributes into the socket data package, and then this package is decoded by *WebServicesServer*. Afterwards, *WebServicesServer* generates the result that is requested by the federate outside the federation. In the opposite way, federates outside the federation can send request based on the web services it customized. While, the *WebServicesServer* receives the request, it translates the request based on the FOM, and then generates a data package which is decoded by the *WebservicesBridge*.

6.1.3 General solution for failure tolerance

As Web Services and UDP are involved in this simulation, the failure tolerance needs to be considered. In this paper, only two failures are being considered: data exchange delay and data package lost.

Figure 12. General solution for failure tolerance

First, let's describe this simulation and define its major elements. Because this simulation is a scale real-time simulation, the scale (simulation time unit) needs to be defined first. Thus, as shown in Figure 12, the simulation time unit (Δt) of the federation is assumed to be 3 seconds, which means every 3 seconds a new event is issued. The approach uses the conservative algorithm described in (Fujimoto 2002) and (Zacharewicz et al. 2005). For example, in Figure 12, Federate A sends one event with a Time stamp (T_{stamp}) plus L_A (Lookahead of A) equals 3 to the event queue, so when simulation time passes one Δt, this event is triggered. Every federate can announce its events with T_{stamp} plus Lookahead. Lookahead is a special non-negative value, which establishes the lowest value of time stamps that can be sent in its Time Stamp Order (TSO) messages. In the simulation, the lookaheads of WebServicesfederate and the federates outside of federation are assumed to be 0. Meanwhile, the lookaheads of the federates inside the federation are bigger than 0 and depend on their own process. When simulation time moves forward, RTI sends Event$_j$ of federate$_j$, whose T_{stampj} + L_j > LBTS$_i$ (Low Bound on Time Stamps), is triggered and sent to the related Federate$_i$. Due to the performance of Web Services and UDP and also this simulation context, the approach proposes that each federate can store three states, S_C, S_{P1}, and S_{P2}. S_C is the current state. S_{P1} is the previous state (back one Δt). S_{P2} is the state before the S_{P1} (back two Δts). The reason for saving three states is to backup necessary information in order to answer overdue customer requests from *WebServiceFederate*. The reason of only saving three states is to limit the times of re-ACK(ACKnowledgment) between the *WebServiceFederate* and federates outside the LAN, which can ensure the message channel between the *WebServicesbridge* and the *WebSerivesserver* fluent and strictly control the increase of each federate's memory load as well as the amount of redundancy in the federate. In addition, in this simulation context, the time scale allows federates inside the LAN to keep their current state for a quite long period, so three backup states are enough for querying (it does not roll-back the state for overdue customer request. It only provides the state query service. This roll-back querying does not affect the message synchronization inside the federation). Normally, the approach also proposes by no reply after the third PING (Packet Internet Grope), the web connection is broken.

The solution for failure tolerance is the following: In this project, some of the federates are federates outside the federation. They send events to the federation and synchronize with other federates via Web services, so the time delay within web transmission and the possibility of package lost should be considered.

Data exchange delay: for example, in figure 12, the federate C sends one message with a time stamp plus L_C equals 9 to the WebServicesFederate. Normally, when the WebServicesFederate receives this message, the current simulation time ($T_{current}$) should be less than the T_{stamp} plus L_C, but, if this message transmission has several seconds time delay, this message arrives $T_{stamp} + L_C < T_{current}$, which means this event has already expired. As the result, there is no reply for the federate C. The solution for data exchange delay is if $T_{current}$ is bigger than $T_{stamp} + L_i$ of the message$_i$, then the WebServicesFederate asks for the past state of requesting federate. There is another situation if the authority of message$_i$ (MA_i) is low, the federation ignores this message.

Package lost: for example, in figure 12, the federate D sends one message with T_{stamp} plus L_D equals 12 to the WebServicesFederate. However, if the package lost during the web transmission, then this message cannot join the simulation of the federation before its own time stamp. As the result, there is no reply for the federate D. The solution for package lost is to set the attribute in the federate D called waiting time (T_{wait}). If T_{wait} is bigger than Δt, then federate D resends the message. The maximum resend time (F_{resend}) is two times Δt. If the WebServicesFederate receives the resend message, it calculates the time difference ($T_{difference}$) and decides which state of the requesting federate is used for the simulation. Another situation is when the authority of the message is low, the federation ignores this message.

The general algorithm of the failure tolerance is the following:

- For federate outside the federation:

```
$F_{resend} = 0$;
while ($F_{resend} < 2$) {
   if ( $T_{wait} > \Delta t$ ){
      resend message;
      $F_{resend}$++;
   } else {
      $F_{resend} = 2$;
   }
}
```

- For WebServicesFederate:

```
if ($T_{current} > T_{stampi} + L_i$) {
   if ($MA_i$ != low) {
      $T_{difference} = (T_{current} - T_{stamp} - L_i)/\Delta t$;
         switch ( $T_{difference}$ ) {
           case 0 : state = $S_C$; break;
           case 1 : state = $S_{P1}$; break;
           case 2 : state = $S_{P2}$; break;
           case 3 : ignore message; break;
   }else{
      Ignore message;
   }
} else {
   if ($T_{stampi} + L_i > LBTS_j$){
      send event to Federate j;
      state = runSimulation();
   } else {
     state = $S_C$;
   }
}
```

- For federate inside the federation:

```
while ( simulation time passes $\Delta t$ ){
   $S_{P2} = S_{P1}$;
```

```
    $S_{P1} = S_C$;
    $S_C$ = runSimulation();
}
```

6.2 Simulation statement

Based on this framework, a demonstration of HLA evolved web services simulation has been carried out in IMS/LAPS of University Bordeaux 1. This simulation is based on poRTIco RTI and Java. It is implemented on Eclipse and can run in Windows NT or UNIX system with a JDK 1.6.0 (or higher) environment and poRTIco environment.

This case describes a working progress in a car manufacturing project. The actors of this case are a car sales agency, a car manufacture factory and suppliers of wheel and engine. The action of this case is simply defined as purchase, manufacture and delivery. The goal of this case is to use this methodology to achieve zero inventories for the car manufacture factory and a just in time order for the supplier by connecting the partners' Information Systems to LAN and WAN. The most important thing is that the car sales agency can track the order from the beginning to the end in real time.

The simulation deployment is illustrated in the figure 13. It establishes a "traditional" HLA federation in the LAN. It sets up some partners end-users as federates. In addition, in order to enable the web services federate, it deploys one server as the web service bridge inside the federation, and deploys another server as the web service server outside the federation, but still inside the LAN. These two special servers are connected by a socket connection. In order to ensure the security of this connection, the socket connection ports are designated and data encryption / decryption are enforced on both side of the *WebservicesServer* and the *WebservicesBridge*. After the *WebservicesServer* is deployed, JAX-RPC based web service interface of HLA federation is released. Then, the different federates outside the federation can link to the federation via diverse web connection.

Figure 13. Simulation deployment

6.3 "Car Selling" case study simulation specification

The use case diagram of this case simulation is shown in figure 14.

Once the Car sales agency (*CSA*) receives an order from a client, it sends an order to the Car manufacture factory (*CMF*), and then the *CMF* calculates the amount of raw materials based on the bill of material. After that *it* dispatches orders to the different suppliers, such as Wheel suppliers (*WS*) and Engine suppliers (*ES*), to get the parts. When suppliers finish the production, they deliver the products to the *CMF* who assembles them and then deliver to the *CSA*. In this simulation, the status of the order is the concerned issue. The *CSA* cares about when it can receive the cars it ordered, and is concerned about the status of this manufacturing process.

Figure 14. Car manufacturing use case

The class diagram of this demonstration is illustrated in figure 15. In this simulation, there are various kinds of federates, and for each federate, there is a correlated federate ambassador. Thus, the factory design pattern (Cooper 2000) is used to allow the creation of flexible federates. While, besides the inheritance from abstract "federate" class, the WebserviceFederate also needs to implement the HLA *WebServiceInterface*. Because this federate cannot synchronize with traditional federates by directly using RTI, it has to connect WebserviceBridge via Web Services, and thus it needs to implement Web Service Interface.

Figure 15. Class diagram of the simulation

Table 1. object class structure table

The object class structure table is shown in table 1. It defines four parameters for each federate. “dayToFinish” shows the remaining number of days of the manufacturing process. “currentState” shows the current state of the order. “count” shows the completed product. “price” represents the current cost.

Figure 16. Sequence diagram of simulation

It is assumed that the *CSA* is the federate outside the federation, while, the *WS*, *ES* and *CMF* are federates inside the federation. The *CSA* is synchronized with *WS*, *ES* and *CMF* by the *Webservicebridge*. Figure 16 illustrates the sequence diagram of this simulation. The *CSA* requires the status of the order through web services, and the *Webservicebridge* uses a socket to connect to the federation, and to listen to the federation, waiting for the answer.

6.4 Case study simulation implementation

The illustration of simulation user interface of the car sales agency and the car manufacture factory is shown in figure 17.

Figure 17. User interface of case simulation

The staff of the car sale agency can input the car number of one order. After the confirmation of this order, the client side shows the total number of manufacturing days. During the car manufacturing, the car sale agency can request the detailed information of the manufacturing process, and the federation immediately sends back the result, which has been generated as a bar chart to vividly illustrate the status of the manufacturing process.

In the car manufacture factory central control interface, the remaining number of days of car manufacture is being presented. In order to assist the car assembling process, the information of the suppliers is being presented. By clicking the “refresh” button, car manufacture factory can receive the latest information from each supplier.

The simulation runs properly with laboratory data, after bridging some gaps between this approach and the API provided by poRTIco. The solution is in the application layer, which means that the Web-RTI functionality is implemented without changing any mechanism or source code of poRTIco. As the result, the link, between the *WebServiceFederate* and the HLA federation, is still not a strong and a flexible link. It could also involve the security issue. At this time, the approach uses encapsulation and encoding method to ensure the security of the data package, but, this additional operation may cause another time delay issue. The simulated solution is currently carried out in a real case.

7. Conclusion

This approach has provided a new systematic methodology for achieving interoperation in heterogeneous information systems. It consists of a synthetic framework harmonized from HLA FEDEP & MDA, Model reverse engineering and HLA Evolved Web Services. This methodology provides a new five step process to generate models for simulation starting from conceptual enterprise models to be converted to MDA models and code to accelerate the rebuilding of legacy information systems for implementation of information system exchange facilities. This method bridges the gap from concepts to implementation in the field of enterprise Information Systems development. The long time experience in management, interoperability and synchronization of data of distributed simulation is reused by applying HLA standard for information exchange. Moreover, it uses the Web services to improve the flexibility of HLA in order to open it to web 2.0 fields of possibilities. This methodology seems promising regarding the growing requirements of real enterprise information systems in particular regarding the distribution, the federated interoperability and the agility to adapt to dynamic context.

This work is still an on-going research project. The methodology presented (each phase of HLA/MDA alignment and the model reversal process) still needs to be refined and detailed. The cohesion between the model reverse process and the HLA platform development needs to be strengthened.

Nevertheless, an implementation of HLA Evolved Web Services has been carried out with an open source RTI, poRTIco. This laboratory case study has already proved the feasibility and efficiency of HLA Evolved WS methodology. Furthermore, European FP7 Future Internet project will allow testing the proposed approach in a large scale ISs communication and improving its performance.

References

Bourey, J.P., Grangel Seguer, R., Doumeingts, G., and Berre, A.J., 2007. *Report on Model Driven Interoperability*, Deliverable DTG 2.3, INTEROP NoE, April, pp. 91. Available from: http://www.interop-vlab.eu/ [accessed 15 June 2009]

Bézivin J., 2005. On the Unification Power of Models. *Software and Systems Modeling*, volume 4, number 2, pp 171-188

Chen D., Vallespir B., Doumeingts G., 1997, GRAI integrated methodology and its mapping onto generic enterprise reference architecture and methodology, *Computers in Industry*, Volume 33 Issue 2-3.

Cooper J. W., 2000, *Java™ Design Patterns: A Tutorial*, addsion wesley longman, Inc. P.19 - 38

Doumeingts G., Ducq Y., 2001, Enterprise Modelling techniques to improve efficiency of enterprises, *International Journal of Production Planning and Control - Taylor & Francis*, Vol. 12, number 2, pp 146-163

Elvesæter B., Hahn A., Berre A., Neple T., 2007.Towards an Interoperability Framework for Model-Driven Development of Software Systems. *Interoperability of Enterprise Software and Applications*, 409-420.

Favre L., Martinez L., Pereira C., 2008. MDA-Based Reverse Engineering of Object Oriented Code. *SERA'08*, 153-160.

Fujimoto R. M., 2000, *Parallel and Distributed Simulation Systems*, Wiley Interscience.

Gorka B., Larrucea X., Elvesæter B., Neple T., Beardsmore A., Friess M., 2007, *A Platform Independent Model for Service Oriented Architectures*, Enterprise Interoperability, Ed, London, S., pp. 23-32. ISBN 978-1-84628-713-8

IEEE std 1516.2-2000, 2000. *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification*, New York: Institute of Electrical and Electronic Engineers.

IEEE std 1516.3-2003, 2003. *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federation Development and Execution Process (FEDEP)*, The Institute of Electrical and Electronic Engineer.

IEEE std 1516™-2010, 2010. *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) -- Framework and Rules*, New York: Institute of Electrical and Electronic Engineers

IEEE std 1730™-2010, 2011. *IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP)*, The Institute of Electrical and Electronic Engineer.

ISO 19440:2007, 2008. *Enterprise integration. Constructs for enterprise modelling*, Organisation internationale de normalisation

Jouault F., Jean B., Mikaël B., 2009. Towards an advanced model-driven engineering toolbox. *Innovations in Systems and Software Engineering*, volume 5, pp 5-12.

Möller B., Clarence D., Mikael K., 2007, Developing Web Centric Federates and Federations using the HLA Evolved Web Services API. *Proceedings of 2007 Spring Simulation Interoperability Workshop*, 07S-SIW-107.

Möller B, Löf S., 2005, Mixing Service Oriented and High Level Architectures in Support of the GIG. *Proceedings of the 2005 Spring Simulation Interoperability Workshop*, 05S-SIW-064.

OMG, 2003. *MDA Guide Version 1.0.1*. Object Management Group, Document number: OMG / 20030601. Available from: www.omg.org/docs/-omg/03-06-01.pdf [accessed 15 June 2009].

OMG, 2010. *Architecture Driven Modernization (ADM): Knowledge Discovery Meta-Model (KDM) v1.2*, OMG, Document number: formal/2010-06-03. Available from: http://www.omg.org/spec/KDM/-1.2 [accessed 15 July 2010].

poRTIco, 2009, *Developer Documentation*. Available from: http://porticoproject.org/index.php?title=-Developer_Documentation [accessed 15 July 2010]

Richardson L., Ruby S., 2007, *RESTful web services*, O'Reilly Media, Inc. p. 299

Shawn P., 2003. The Next Step Applying the Model Driven Architecture to HLA. *Proceedings of the 2003 Spring. Workshop*, 03S-SIW -123.

“Simulation Interoperability Standards Organization”, accessed May 10, 2011, http://www.sisostds.org/Home.aspx

Tolk A., 2002. Avoiding another Green Elephant —A Proposal for the Next Generation HLA based on the Model Driven Architecture. *Proceedings of the 2002 Fall Simulation Interoperability Workshop*, 02F-SIW-004.

Touzi J., 2007, *Aide à la conception de Système d’Information Collaboratif support de l’interopérabilité des entreprises*, PHD thesis, Institut National Polytechnique de Toulouse.

Trbovich S., Reading R., 2005. Simulation and Software Development for Capabilities Based Warfare: An Analysis of Harmonized Systems Engineering Processes. *Proceedings Spring Simulation Interoperability Workshop*, 05S-SIW-106.

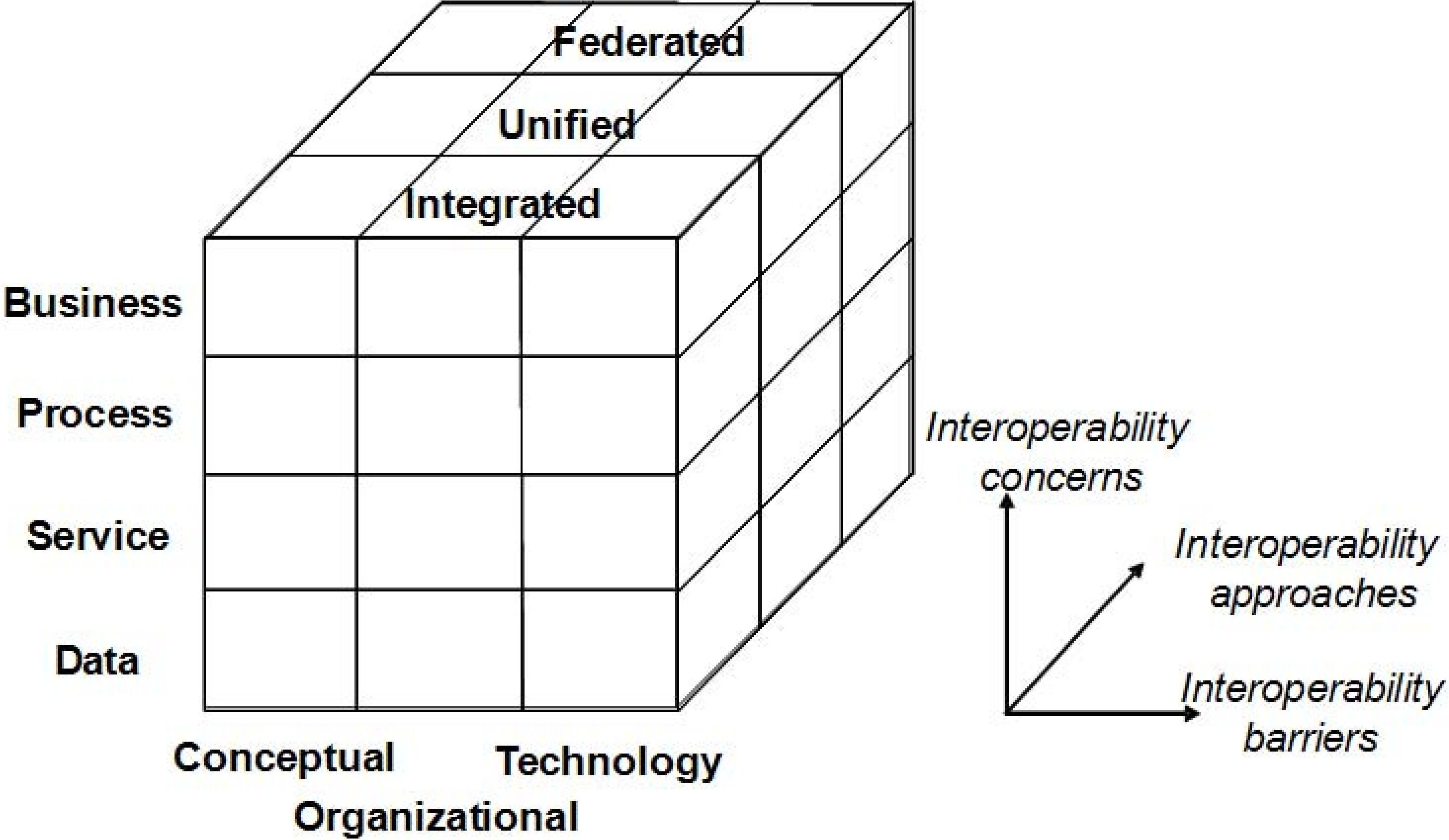
Tu Z., Zacharewicz G., Chen D., 2010. Unified Reversible Life Cycle for Future Interoperable Enterprise Distributed Information Systems. *IESA 2010-Interoperability for Enterprise Software & Applications 2010*, 57-66, April 14-15, Coventry (UK).

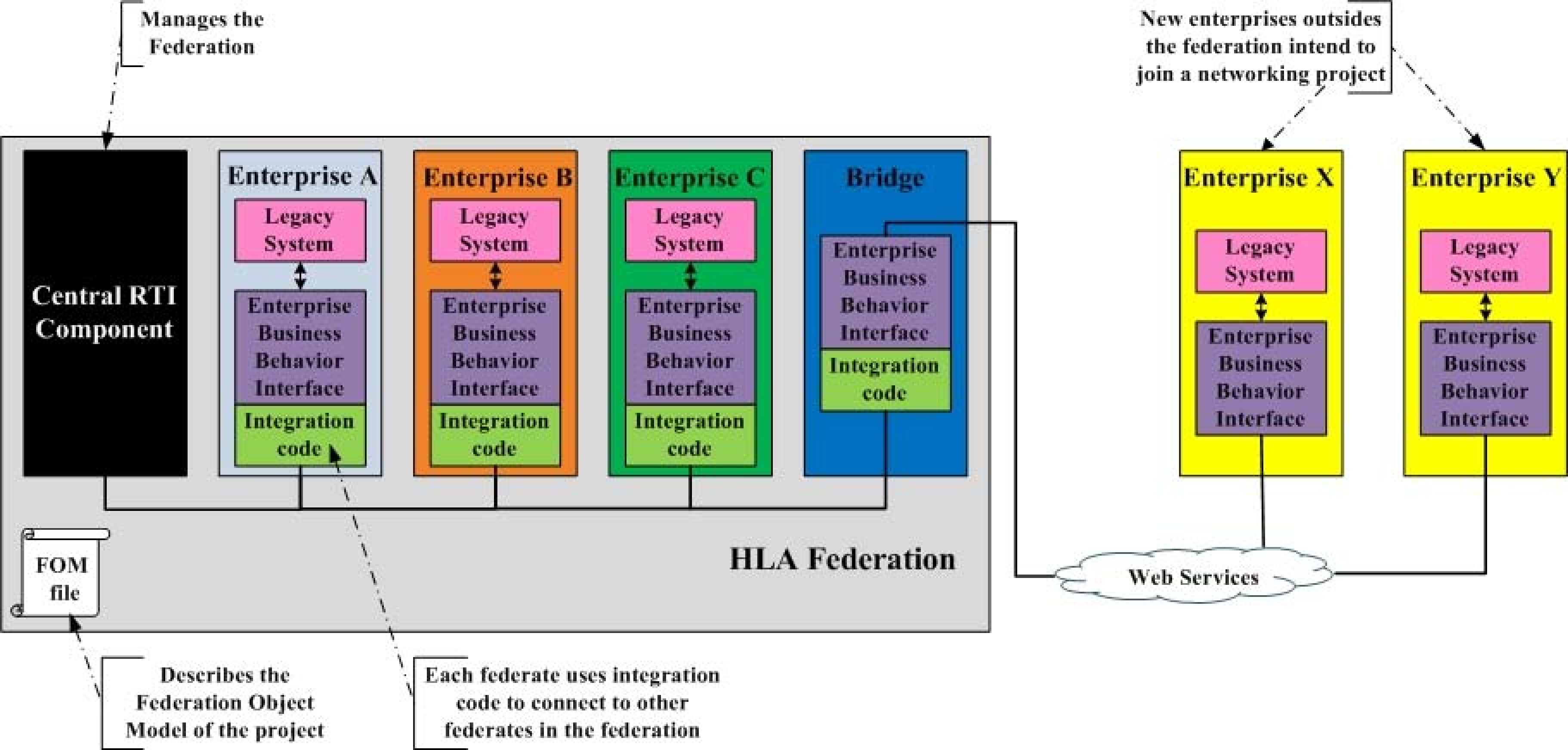
Ullberg J., Chen D., Johnson P., 2007. Barriers Driven Methodology For Enterprise Interoperability. *IFIP International Federation for Information Processing*, 453-460.

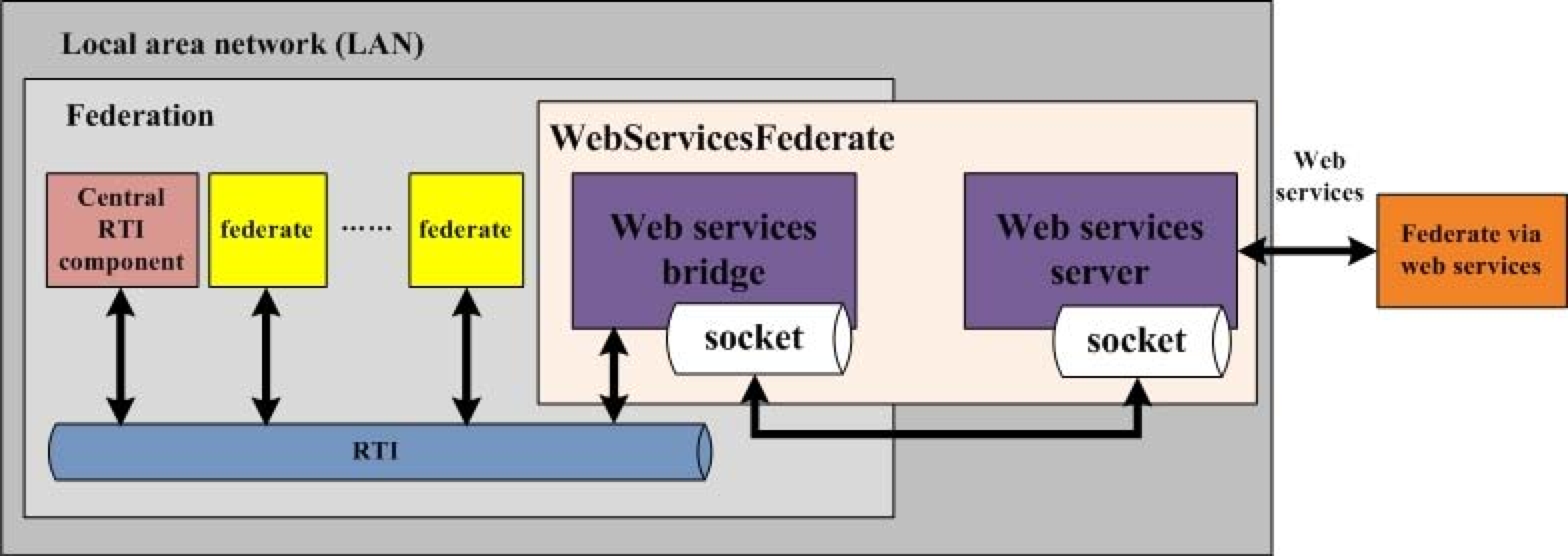
Zacharewicz G., Chen D., Vallespir B., 2008. HLA Supported Federation Oriented Enterprise Interoperability, Application to Aerospace Enterprises. *Proceeedings of 2008 International Simulation Multiconference EuroSISO*, 08E-SIW-074, June 19-19, Edinburgh (Scotland).

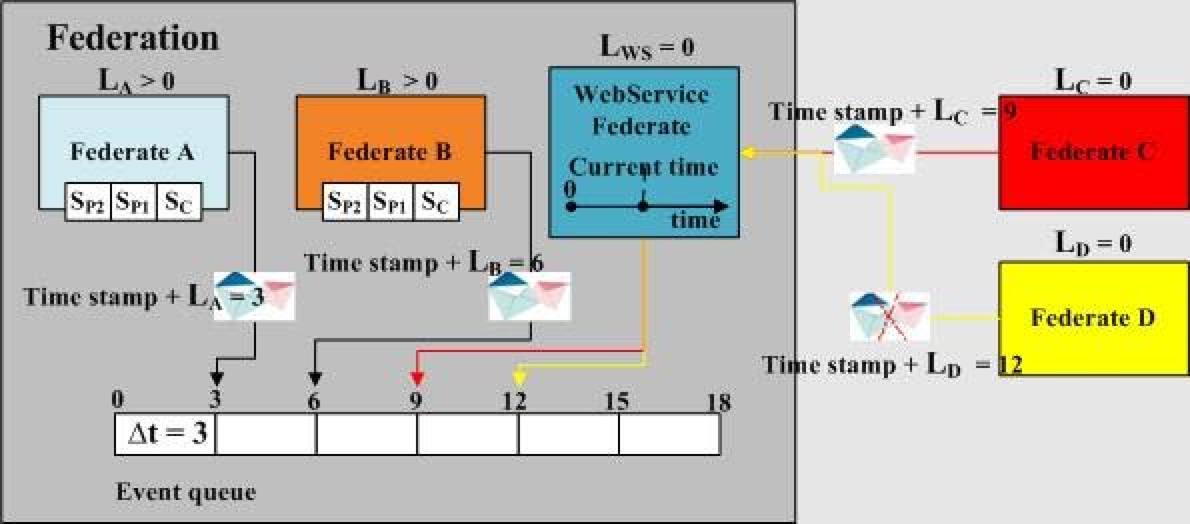
Zacharewicz, G., Chen, D., Vallespir, B., 2009. Short-Lived Ontology Approach for Agent/HLA Federated Enterprise Interoperability, *Proceedings IEEE of International Conference I-ESA China 2009 Interoperability for Enterprise Software and Applications*, pp. 329-335, April 22-23, Beijing (China)

Zacharewicz G., Labarthe O., Chen D., Vallespir B., 2011. A Multi Agent/HLA Platform for Enterprises Interoperability: Short-Lived Ontology Based, *Electronic Supply Network Coordination in Intelligent and Dynamic Environment: Modeling and Implementation*, 319-346


Federated
Unified
Integrated
Business
Process
Service
Data
Conceptual
Technology
Organizational
Interoperability concerns
Interoperability approaches
Interoperability barriers


Manages the Federation
New enterprises outsides the federation intend to join a networking project
Central RTI Component
Enterprise A
Legacy System
Enterprise Business Behavior Interface
Integration code
Enterprise B
Legacy System
Enterprise Business Behavior Interface
Integration code
Enterprise C
Legacy System
Enterprise Business Behavior Interface
Integration code
Bridge
Enterprise Business Behavior Interface
Integration code
Enterprise X
Legacy System
Enterprise Business Behavior Interface
Enterprise Y
Legacy System
Enterprise Business Behavior Interface
FOM file
HLA Federation
Web Services
Describes the Federation Object Model of the project
Each federate uses integration code to connect to other federates in the federation


Local area network (LAN)
Federation
WebServicesFederate
Central RTI component
federate
……
federate
Web services bridge
socket
Web services server
socket
Web services
Federate via web services
RTI



Federation

$L_A > 0$

Federate A

S_{P2} S_{P1} S_C

$L_B > 0$

Federate B

S_{P2} S_{P1} S_C

$L_{WS} = 0$

WebService Federate

Current time

0

time

$L_C = 0$

Time stamp + L_C = 9

Federate C

$L_D = 0$

Federate D

Time stamp + L_D = 12

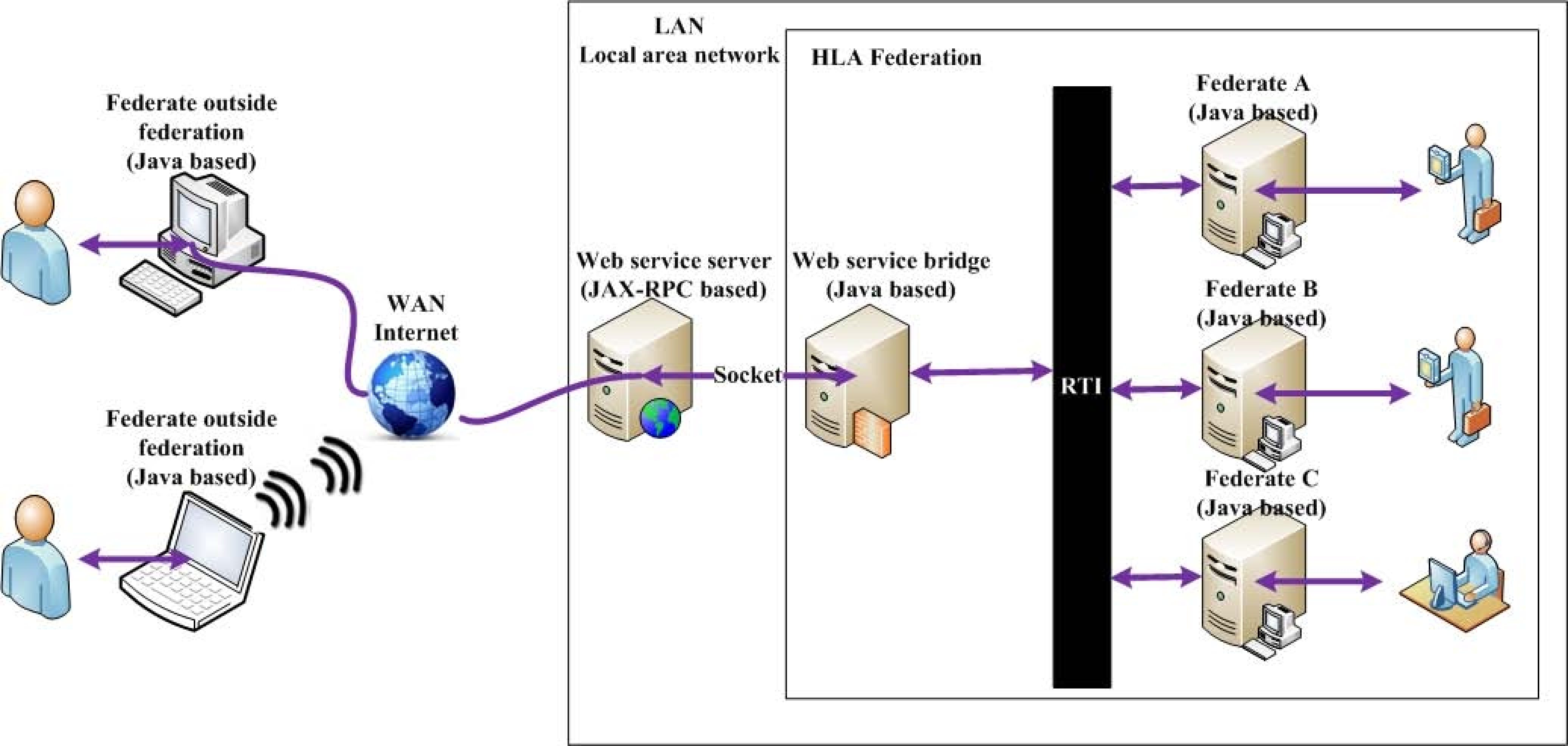
Time stamp + L_B = 6

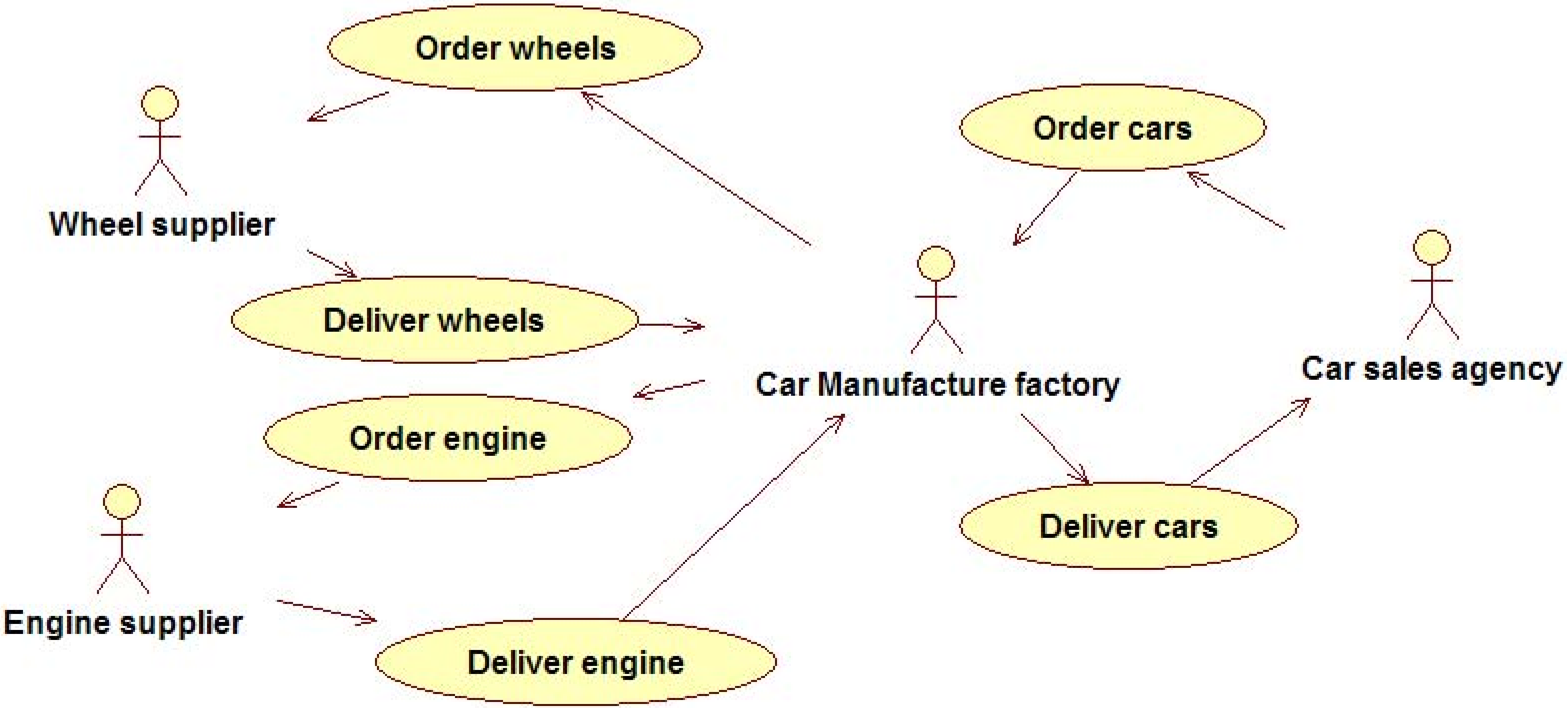
Time stamp + L_A = 3

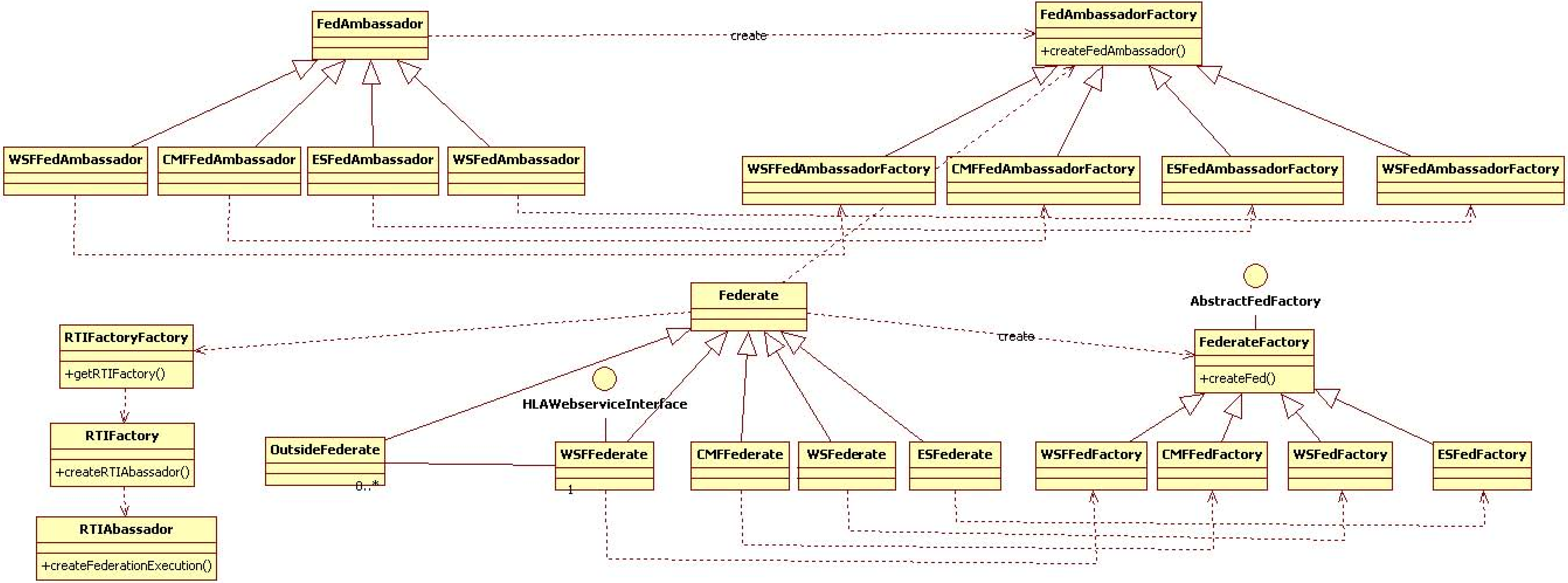
0 3 6 9 12 15 18

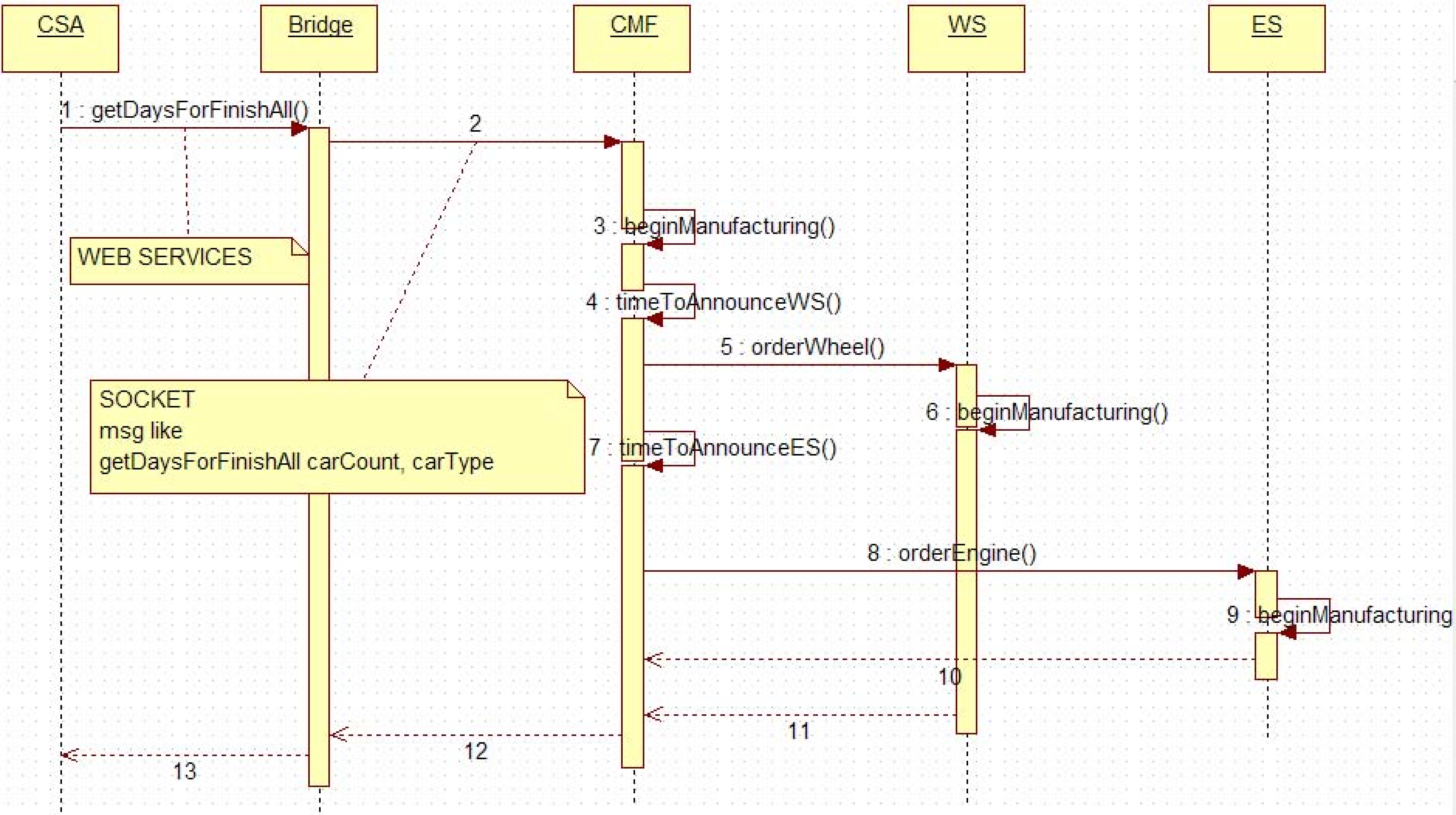
$\Delta t = 3$

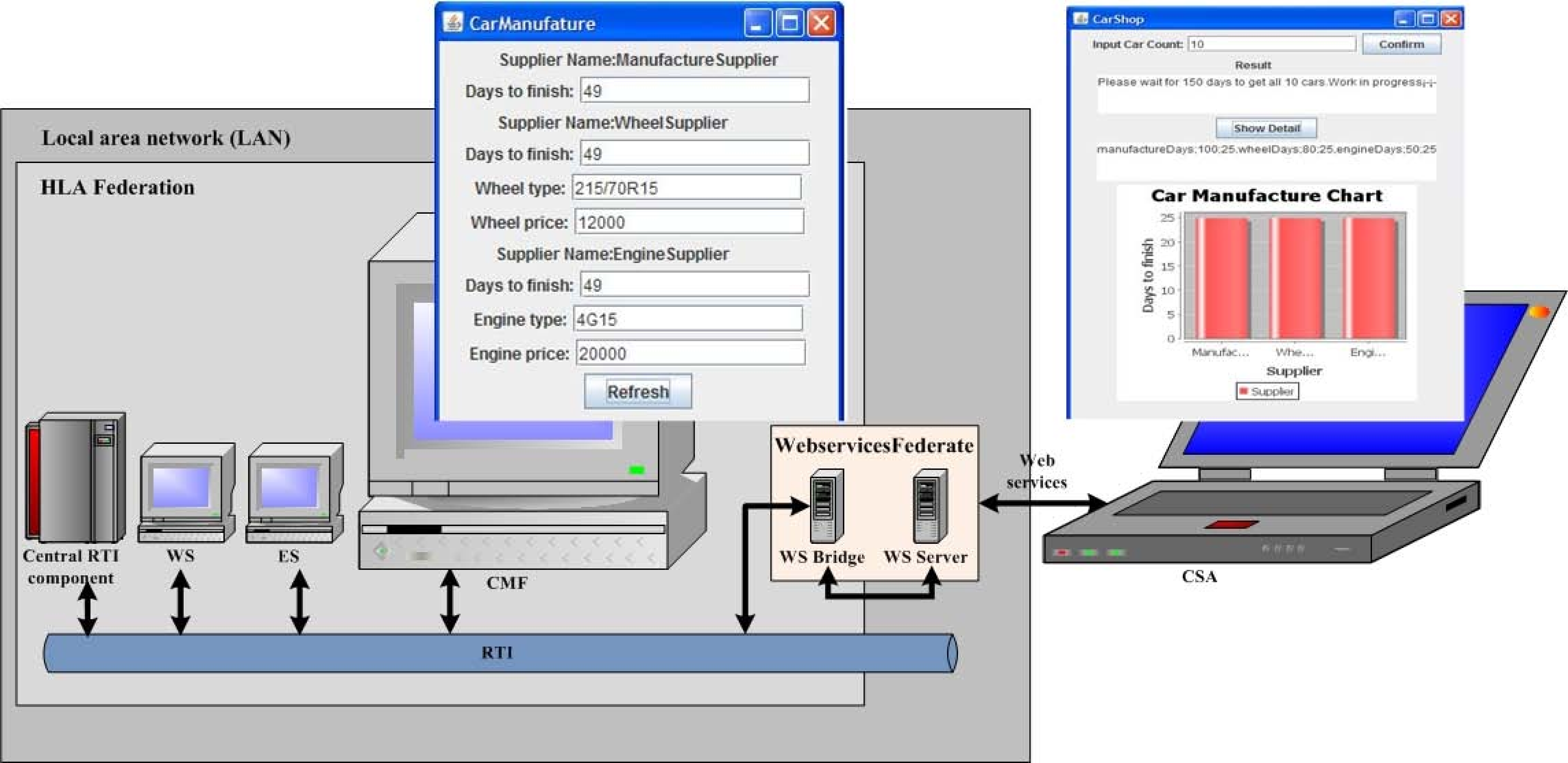
Event queue


LAN
Local area network
HLA Federation
Federate outside federation (Java based)
Federate A (Java based)
Federate B (Java based)
Federate C (Java based)
Web service server (JAX-RPC based)
Web service bridge (Java based)
WAN Internet
Socket
RTI
Federate outside federation (Java based)


Order wheels
Wheel supplier
Order cars
Deliver wheels
Car Manufacture factory
Car sales agency
Order engine
Deliver cars
Engine supplier
Deliver engine


FedAmbassador
create
FedAmbassadorFactory
+createFedAmbassador()
WSFFedAmbassador
CMFFedAmbassador
ESFedAmbassador
WSFedAmbassador
WSFFedAmbassadorFactory
CMFFedAmbassadorFactory
ESFedAmbassadorFactory
WSFedAmbassadorFactory
AbstractFedFactory
Federate
RTIFactoryFactory
+getRTIFactory()
create
FederateFactory
+createFed()
HLAWebserviceInterface
RTIFactory
+createRTIAbassador()
OutsideFederate
WSFFederate
CMFFederate
WSFederate
ESFederate
WSFFedFactory
CMFFedFactory
WSFedFactory
ESFedFactory
0..*
1
RTIAbassador
+createFederationExecution()


CSA
Bridge
CMF
WS
ES
1 : getDaysForFinishAll()
2
3 : beginManufacturing()
WEB SERVICES
4 : timeToAnnounceWS()
5 : orderWheel()
6 : beginManufacturing()
SOCKET
msg like
getDaysForFinishAll carCount, carType
7 : timeToAnnounceES()
8 : orderEngine()
9 : beginManufacturing
10
11
12
13



Supplier Name:ManufactureSupplier

Days to finish: 49

Supplier Name:WheelSupplier

Days to finish: 49

Wheel type: 215/70R15

Wheel price: 12000

Supplier Name:EngineSupplier

Days to finish: 49

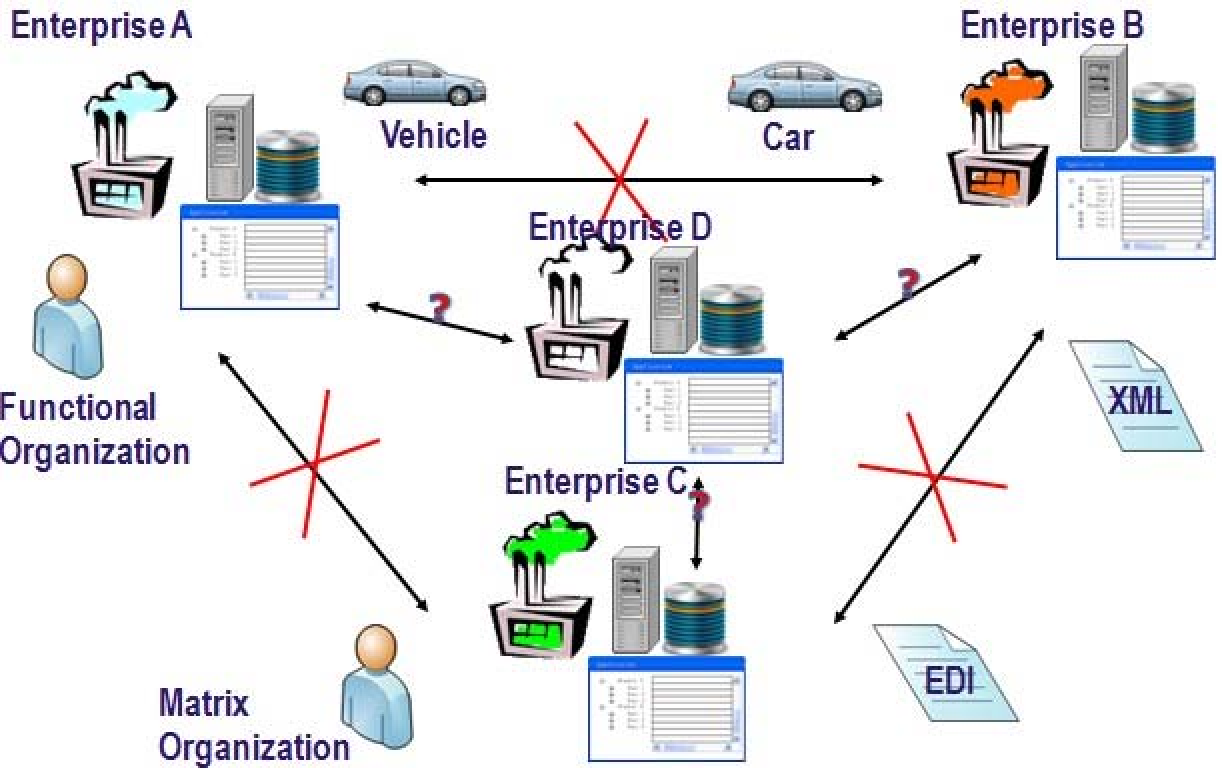
Engine type: 4G15

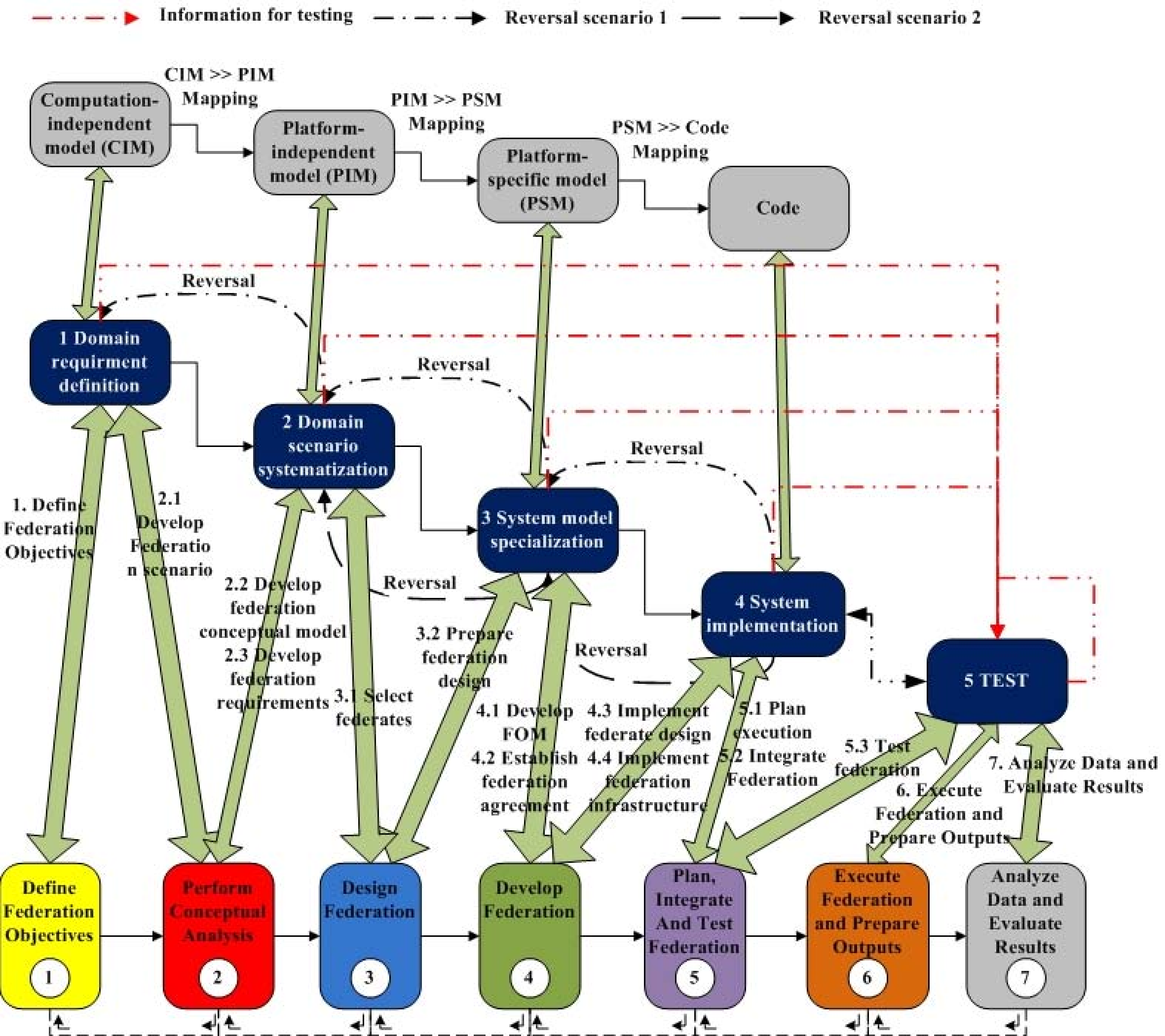
Engine price: 20000

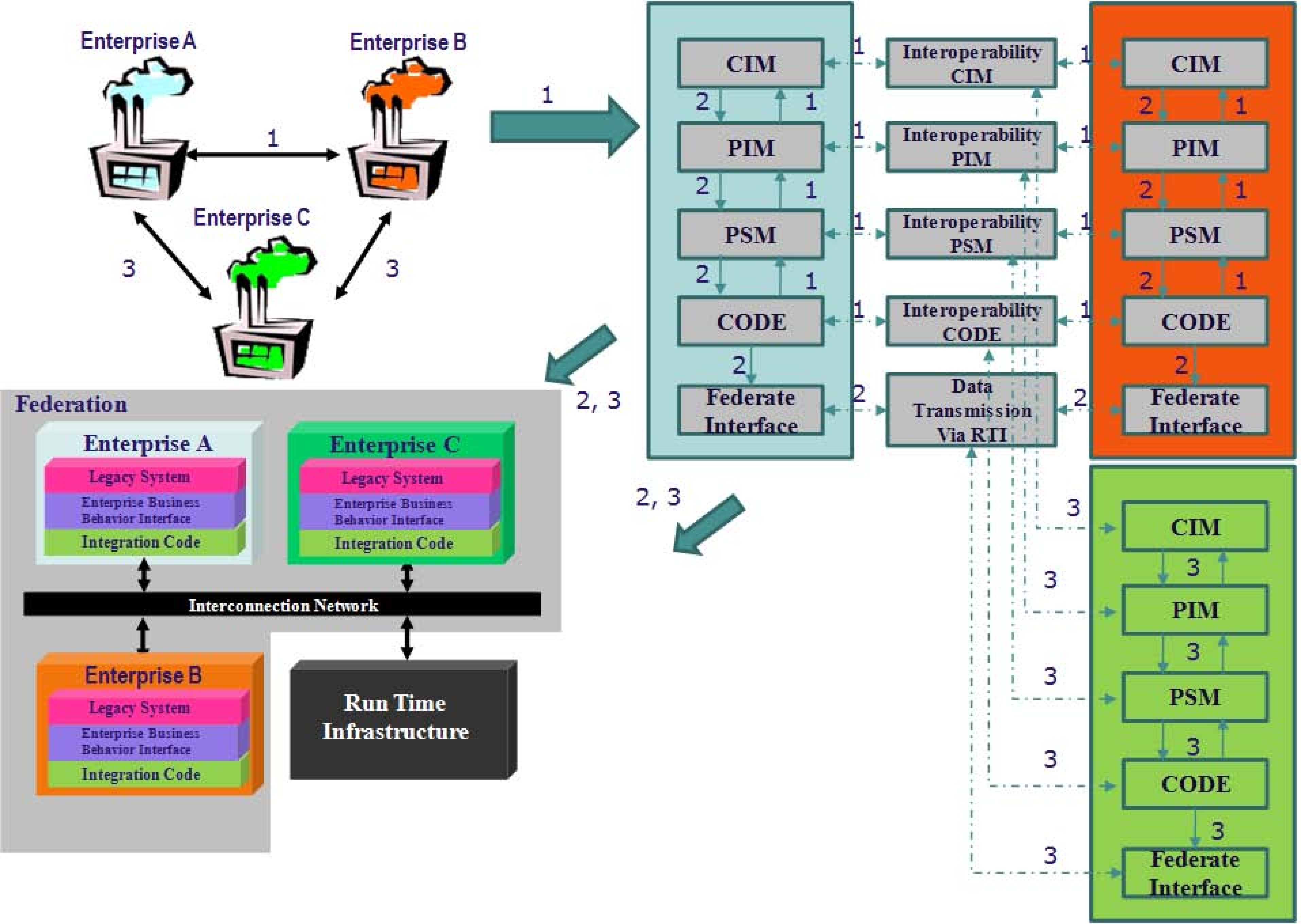
Input Car Count: 10

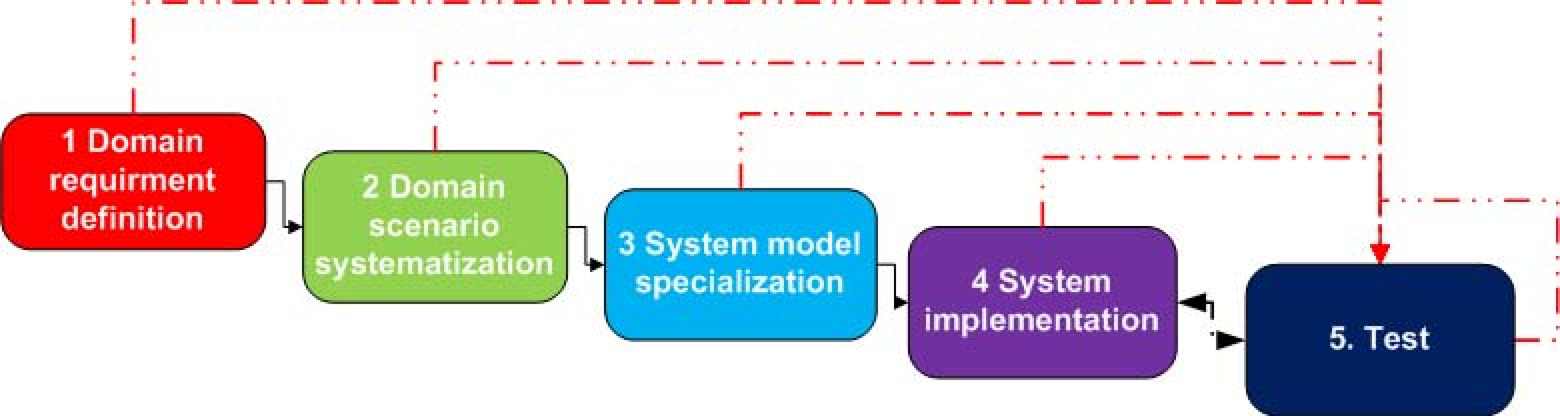
Please wait for 150 days to get all 10 cars.Work in progress;-|-

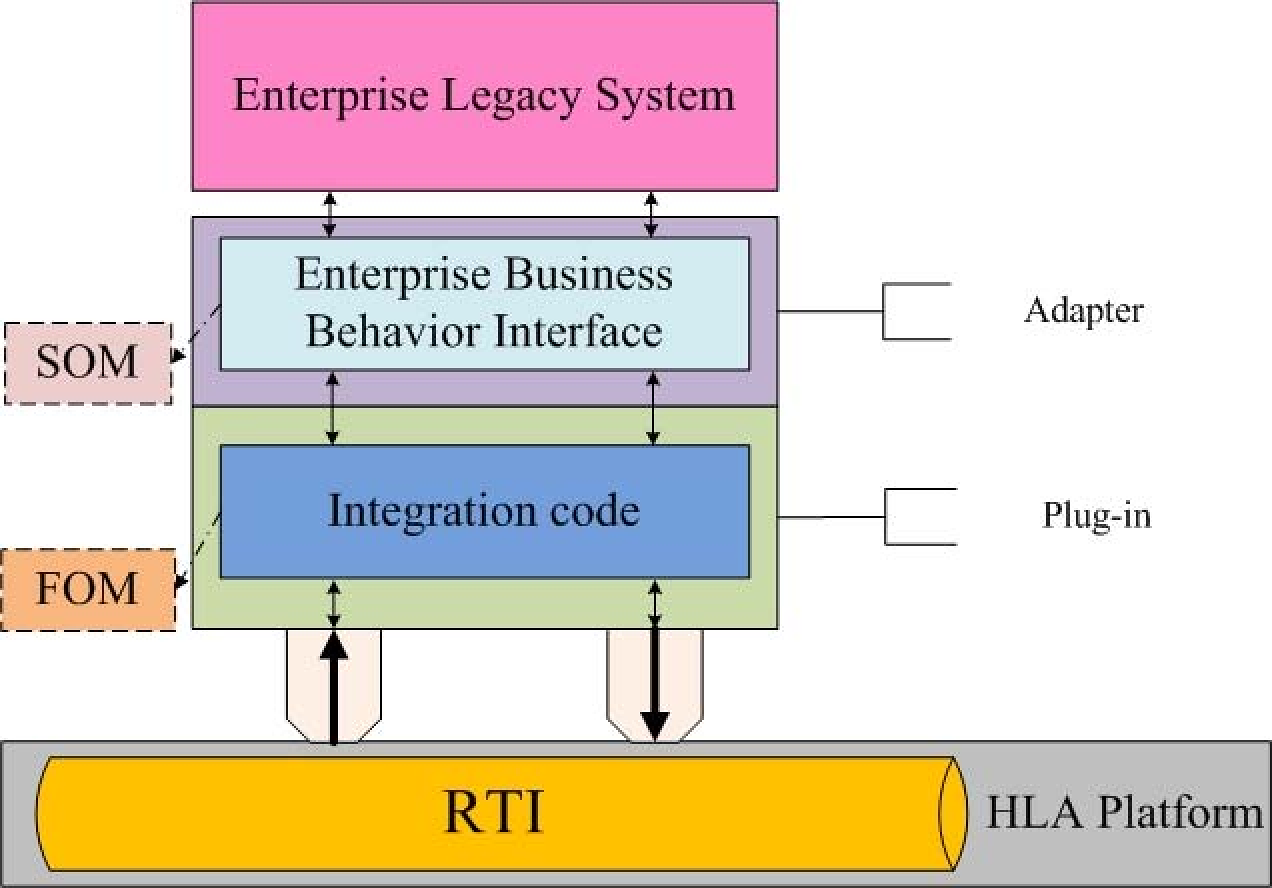
manufactureDays:100:25.wheelDays:80:25.engineDays:50:25

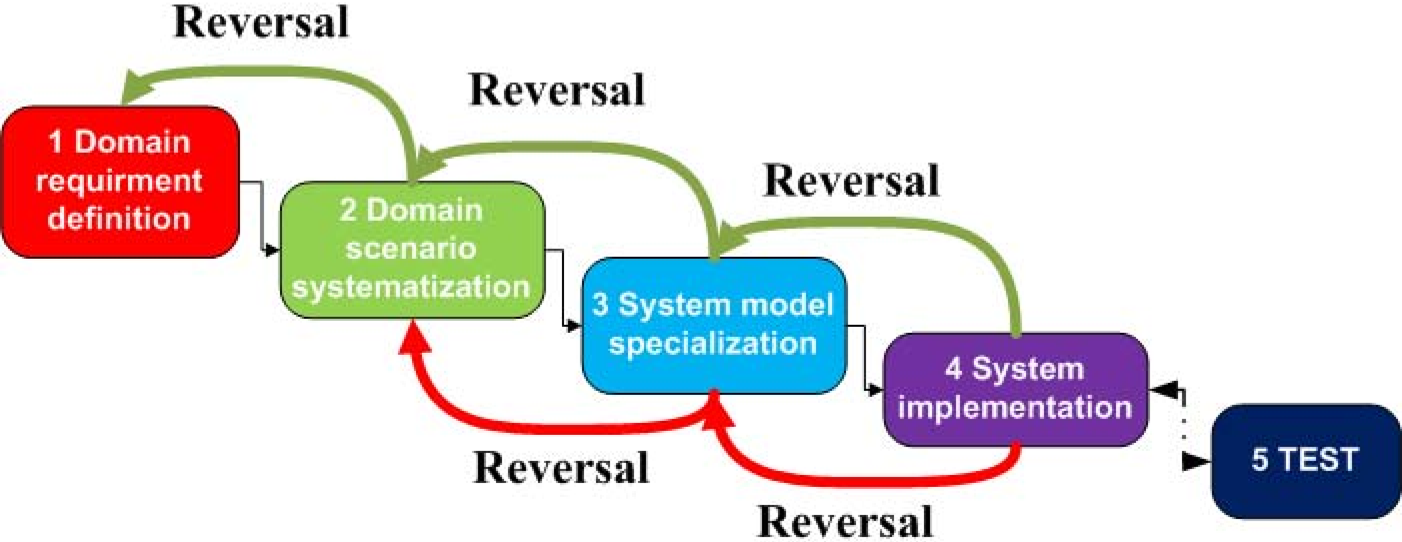

Enterprise A
Vehicle
Car
Enterprise B
Enterprise D
Functional Organization
XML
Enterprise C
Matrix Organization
EDI

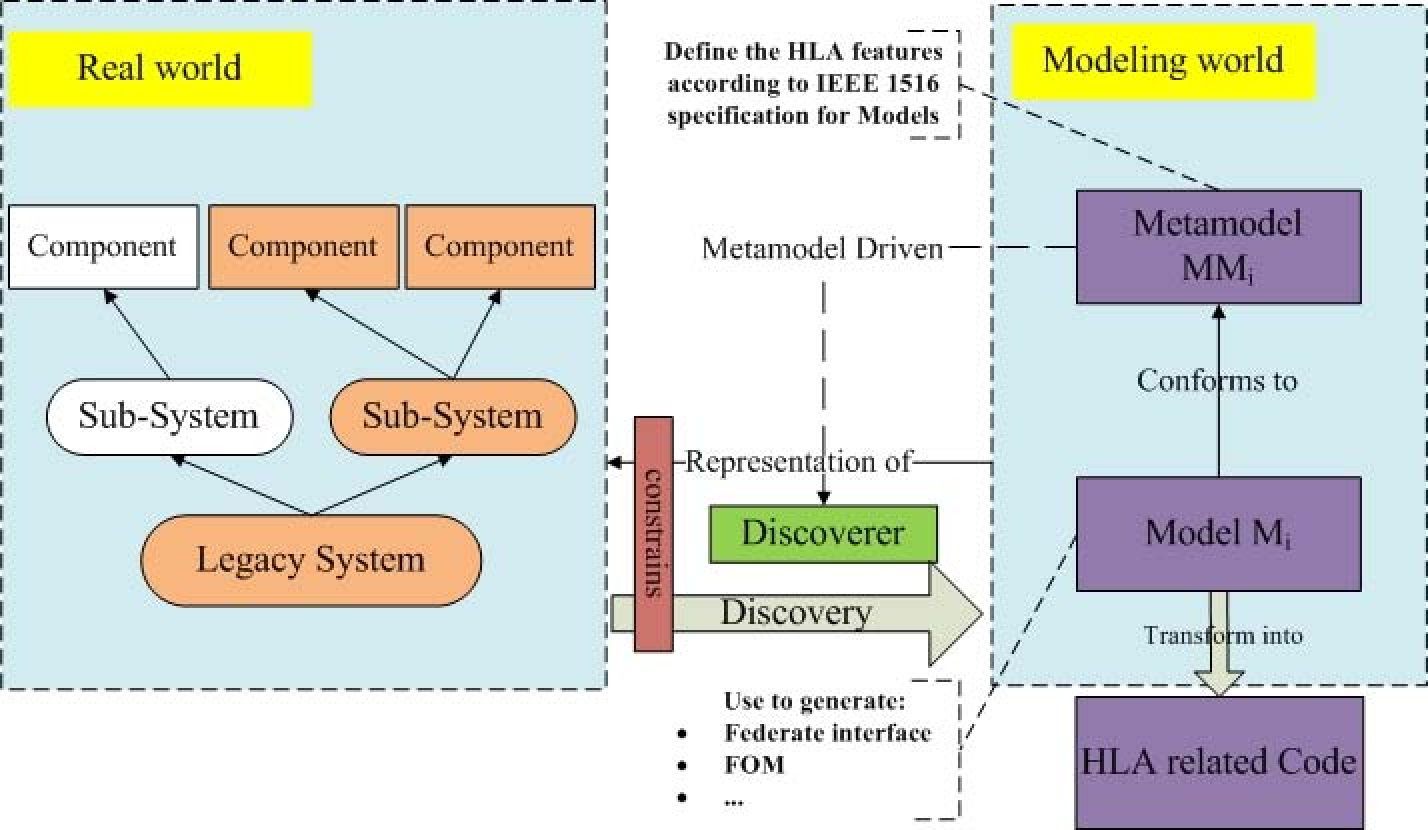

Information for testing
Reversal scenario 1
Reversal scenario 2
Computation-independent model (CIM)
CIM >> PIM Mapping
Platform-independent model (PIM)
PIM >> PSM Mapping
Platform-specific model (PSM)
PSM >> Code Mapping
Code
Reversal
1 Domain requirment definition
Reversal
2 Domain scenario systematization
Reversal
3 System model specialization
4 System implementation
5 TEST
1. Define Federation Objectives
2.1 Develop Federation scenario
2.2 Develop federation conceptual model
2.3 Develop federation requirements
3.1 Select federates
Reversal
3.2 Prepare federation design
Reversal
4.1 Develop FOM
4.2 Establish federation agreement
4.3 Implement federate design
4.4 Implement federation infrastructure
5.1 Plan execution
5.2 Integrate Federation
5.3 Test federation
6. Execute Federation and Prepare Outputs
7. Analyze Data and Evaluate Results
Define Federation Objectives
1
Perform Conceptual Analysis
2
Design Federation
3
Develop Federation
4
Plan, Integrate And Test Federation
5
Execute Federation and Prepare Outputs
6
Analyze Data and Evaluate Results
7


Enterprise A
Enterprise B
Enterprise C
1
3
3
1
2, 3
2, 3
CIM
PIM
PSM
CODE
Federate Interface
Interoperability CIM
Interoperability PIM
Interoperability PSM
Interoperability CODE
Data Transmission Via RTI
CIM
PIM
PSM
CODE
Federate Interface
CIM
PIM
PSM
CODE
Federate Interface
1
2
3
Federation
Enterprise A
Legacy System
Enterprise Business Behavior Interface
Integration Code
Enterprise C
Legacy System
Enterprise Business Behavior Interface
Integration Code
Interconnection Network
Enterprise B
Legacy System
Enterprise Business Behavior Interface
Integration Code
Run Time Infrastructure


1 Domain requirment definition
2 Domain scenario systematization
3 System model specialization
4 System implementation
5. Test


Enterprise Legacy System
Enterprise Business Behavior Interface
SOM
Adapter
Integration code
Plug-in
FOM
RTI
HLA Platform


Reversal
Reversal
Reversal
1 Domain requirment definition
2 Domain scenario systematization
3 System model specialization
4 System implementation
5 TEST
Reversal
Reversal


Real world
Component
Component
Component
Sub-System
Sub-System
Legacy System
constrains
Define the HLA features according to IEEE 1516 specification for Models
Metamodel Driven
Representation of
Discoverer
Discovery
Use to generate:
- Federate interface
- FOM
- ...
Modeling world
Metamodel MM$_i$
Conforms to
Model M$_i$
Transform into
HLA related Code

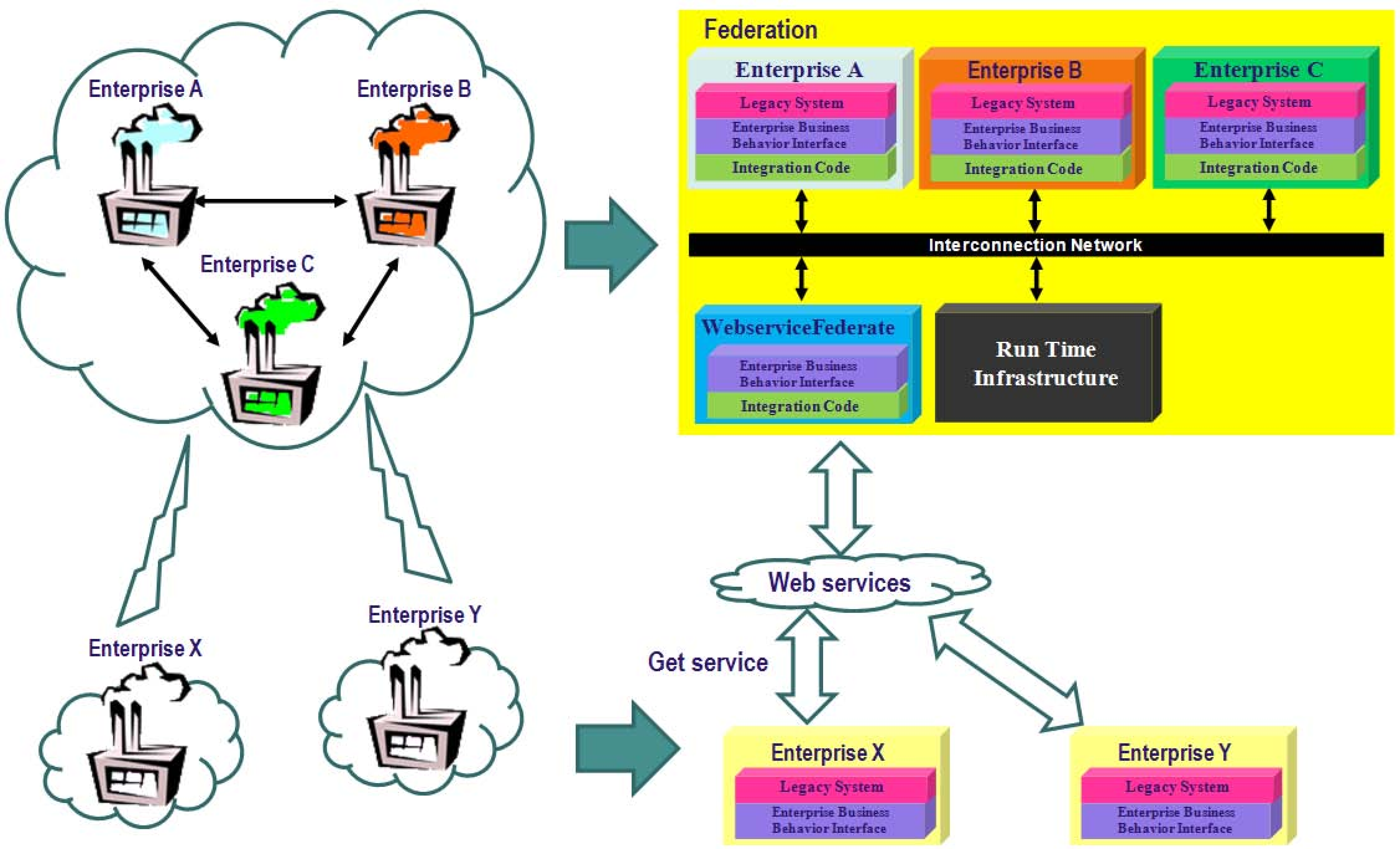

Enterprise A
Enterprise B
Enterprise C
Enterprise X
Enterprise Y
Federation
Enterprise A
Legacy System
Enterprise Business Behavior Interface
Integration Code
Enterprise B
Legacy System
Enterprise Business Behavior Interface
Integration Code
Enterprise C
Legacy System
Enterprise Business Behavior Interface
Integration Code
Interconnection Network
WebserviceFederate
Enterprise Business Behavior Interface
Integration Code
Run Time Infrastructure
Web services
Get service
Enterprise X
Legacy System
Enterprise Business Behavior Interface
Enterprise Y
Legacy System
Enterprise Business Behavior Interface

Table 1. object class structure table

Object root	WheelSupplier	dayToFinish
		currentState
		count
		price
	CarManufacturer	dayToFinish
		currentState
		count
		price
	EngineSupplier	dayToFinish
		currentState
		count
		price