Developing an enterprise operating system for the monitoring and control of enterprise operations

Joseph Youssef

université de BORDEAUX

THÈSE PRÉSENTÉE

POUR OBTENIR LE GRADE DE

DOCTEUR DE

L'UNIVERSITÉ DE BORDEAUX

ECOLE DOCTORALE DES SCIENCES PHYSIQUES ET DE L'INGÉNIEUR

SPECIALITE : PRODUCTIQUE

Par Joseph YOUSSEF

DEVELOPING AN ENTERPRISE OPERATING SYSTEM FOR THE MONITORING AND CONTROL OF ENTERPRISE OPERATIONS

Sous la direction de : David CHEN
(Co-directeur : Gregory ZACHAREWICZ)

Soutenue le 21 Décembre 2017

Membres du jury :

M. DUCQ, Yves	Professeur, Université de Bordeaux	Président
M. KASSEL, Stephan	Professeur, Université des Sciences Appliquées de Zwickau	Rapporteur
M. ARCHIMEDE, Bernard	Professeur, Ecole Nationale D'Ingénieurs de Tarbes	Rapporteur
M. CHEN, David	Professeur, Université de Bordeaux	Directeur
M. ZACHAREWICZ, Gregory	MCF (H.D.R), Université de Bordeaux	Co-directeur
M. DACLIN, Nicolas	Maître-Assistant, Ecole des Mines d'Alès	Examinateur

Titre : Développement d'un Système D'exploitation des Entreprises pour le Suivi et le Contrôle des Opérations.

Résumé: Le système d'exploitation (OS) est un concept bien connu en informatique comme interface entre l'Homme et le matériel informatique (MacOS, Windows, IOS, Android, ...). Dans le but de développer la future génération de systèmes d'entreprise basés sur les principes de l'IoT et du Cyber-Physique, cette thèse propose de développer un système d'exploitation d'entreprise « System d'Exploitation des Entreprises » (EOS); Contrairement à ERP, qui est défini comme un programme qui permet à l'organisation au niveau opérationnel d'utiliser un système d'applications intégrées afin d'automatiser de nombreuses fonctions liées à la technologie et aux services, EOS servira d'interface entre les gestionnaires d'entreprise et les ressources d'entreprise pour le suivi en temps réel et le contrôle des opérations.

Nous présenterons d'abord le contexte, les priorités, les défis et les résultats escomptés. Ensuite, un ensemble d'exigences et de fonctionnalités d'EOS est décrit. Après, un état de l'art existant sur les travaux pertinents est donné et mis en correspondance avec les exigences spécifiées liées à EOS. Par la suite, et en fonction des exigences et des résultats, les architectures conceptuelle, technique et d'implantation sont décrites, y compris tous les composants internes et externes. La dernière partie présenteront deux exemples dans les secteurs bancaire et manufacturier pour illustrer l'utilisation de l'EOS.

Mots clés : Système d'exploitation, architecture, modèle, infrastructure, interopérabilité d'entreprise, simulation distribuée.

Title: Developing an Enterprise Operating System for the Monitoring and Control of Enterprise Operations.

Abstract: Operating System (OS) is a well-known concept in computer science as an interface between human and computer hardware (MacOS, Windows, IOS, Android,…). In the perspective of developing future generation of enterprise systems based on IoT and Cyber-Physical System principles, this doctorate research proposes to develop an Enterprise Operating System (EOS); Unlike ERP, which is defined as a platform that allows the organization at the operational level to use a system of integrated applications in order to automate many back office functions related to technology and services, EOS will act as an interface between enterprise business managers and enterprise resources for real time monitoring and control of enterprise operations.

The thesis presents at first the context, priorities, challenges and expected results. Then a set of requirements and functionalities of EOS is described. After that, a survey on existing relevant works is given and mapped to the specified requirements related to EOS. Afterwards, and based on the requirements and state-of-the-art results, the EOS conceptual, technical and implementation architectures are outlined including all internal and external components. The last part draws two examples in the banking and manufacturing sectors to illustrate the use of the EOS.

Keywords : Operating system, Architecture, Model, Infrastructure, Enterprise interoperability, Distributed simulation.

Laboratoire de l'Intégration du Matériau au Système

[IMS, CNRS UMR 5218, Bât. A31 - 351 Cours de la Libération - 33400 Talence]

Acknowledgment

This thesis took place within the group “Productique du Laboratoire de l'Intégration du Matériau au Système (IMS, CNRS UMR5218)” of the University of Bordeaux.

Firstly, I would like to express my sincere gratitude to my advisors: Mr. David CHEN - Professor at the University of Bordeaux, and Mr. Gregory ZACHAREWICZ - Associate Professor at the University of Bordeaux, for their continuous support, patience, motivation, and immense knowledge. Their guidance helped me in all the research and thesis period. I could not have imagined having better advisors and mentors for my Ph.D. study.

Besides my advisors, I would like to thank the Members of my PhD Committee: Mr. Stephan KASSEL - Professor at “Université des Sciences Appliquées de Zwickau”, Mr. Bernard ARCHIMEDE - Professor at “Ecole Nationale D’Ingénieurs de Tarbes”, Mr. Yves DUCQ – Professor at the University of Bordeaux, and Mr. Nicolas DACLIN – Assistant Professor at “Ecole des Mines d’Alès”; for serving as my Committee Members and for their helpful career advice and suggestions.

My sincere thanks also goes to Mr. Xavier MOREAU – Professor at the University of Bordeaux and Head of the “APSI” group at the “IMS” Laboratory, and Mr. Roy ABI ZEID DAOU – Assistant Professor at the Lebanese German University; who provided me the opportunity to join the “IMS” Laboratory team.

I would like to thank all the PhD students, researchers and staff of the “IMS” Laboratory for their kindness and for all the exchanges we have had.

Last but not the least, my eternal gratitude goes to my Parents, my Wife, my angel Boy, and all my Family for their unfailing support during all these years of study.

Outline

Figures

Tables

Chapter 1. General Introduction

Nowadays, daily enterprise operations are not effectively monitored and controlled. This is mainly due to the lack of a system-wide operating system, the variety of heterogeneous enterprise applications and the difficulties to obtain real-time data of enterprise operations. In the past thirty years, tremendous efforts have been spent to develop enterprise integration (such as CORBA, EAI…) (Linthicum 1999), enterprise resource planning and management solutions known as ERP (such as SAP, Oracle…). These various approaches are still not satisfactory because of their imposed architectures, high cost and long delay to implement in enterprises. With the development of the next generation enterprise manufacturing systems based on Internet of Things (IoT) and Cyber Physical System (CPS) principle, future enterprises will need tailored "plug & play" interoperable solutions to quickly configure their systems and to real-timely monitor and control their daily operations.

This thesis proposes to develop an Enterprise Operating System (EOS) that aims at joining existing approaches together for the sake of complementarity and synergy. The proposed EOS will behave in the same manner as a computer operating system (OS), but in the enterprise context to monitor and control enterprise operations. The envisioned EOS will execute enterprise models defined by business managers, trigger enterprise operations with dynamically allocated enterprise resources and monitor the use of enterprise resources (Human, Machining, Computing) through various sensing devices and front-ends.

This general introduction will present the research context, define research objectives, outline the proposed contribution and give the structure of this research report.

1.1 Context and problems

Industrial organizations operating in the market are becoming increasingly complex and dynamic due to the arrangement, coordination, and management of complex computer systems, middleware and services. These enterprises are constantly redesigning their structures and business processes to practice enterprise application integration in order to accommodate and implement their increasing services, data orchestration and distribution methods. On the other hand, rapid changing market requirements will oblige an enterprise to be small, smart and scalable to be able to

operate quick change.

In the last 30 years, there are many initiatives to develop enterprise integration platforms and infrastructures such as CORBA, EAI, etc. However, those initiatives fail to provide satisfactory solutions to integrate heterogeneous enterprise application software, especially for SMEs. Enterprise integration is still seen as a complex, costly and time-consuming task.

On the other hand, several packaged software that are provided by ERP vendors and other products (PeopleSoft, SAP and JD Edwards) deliver improved enterprise application integration by offering an integrated suite of applications to perform standard business functions.

Until recent years, packaged-software vendors concentrated on one application or a suite of applications that they automated without regard to other possible applications that a company might have in its systems portfolio. Consequently, large corporations that purchased and installed packaged applications (such as ERP packages) found themselves with islands of data and processing that must be bridged to other islands. As these bridges grew, the maintenance of a system and its interfaces also grew to take up more and more resources.

It has been estimated that 78% of enterprises have chosen ERP solutions or multiple systems in order to facilitate the data orchestration by connecting several software and hardware together at the operational level (Burnson 2015); however during the last couple of years, the image of ERP systems seems to have changed from a highly promising into a very demanding system (Burnson 2015). This solution may constraint the business due to the top down “enclosing” methodology. It is becoming enormously in size and complexity which make the implementation and the updates very time consuming, involving high costs and disrupt current processes and operations. Other products are being also developed and offered at lower costs while being characterized by higher financial and operational risks (J. Gray and Reuter 1993).

Facing the problems in enterprise integration and ERP packaged solution; a different approach consists in developing an Enterprise Operating System (EOS) by setting loosely coupled connections between enterprise’s software in the idea of federated interoperability with only one simplified central orchestrator component (Chen, Youssef, and Zacharewicz 2015). It is also seen as one of the key steps towards the future generation enterprise manufacturing systems to provide an alternative for enterprises with a tailored solution especially for Small and Medium-sized Enterprises.

It has been considered that traditional factory automation, empowered with Internet of Things (IoT), will operate a challenging industrial migration towards the Fourth Industrial Revolution (or Industry 4.0). Enterprises will need more and more dynamic engineering capability to allow quickly reconfiguring their systems, in order to set up collaboration relationships with their internal and external business partners (Liffler and Tschiesner 2013; Hermann, Pentek, and Otto 2016). In this context, the massive use of connected sensors and mobile devices connected to humans will make possible the collect of real time data throughout the enterprise. Those data will be used by EOS to dynamically finding and allocating available resources, triggering enterprise operations and monitoring their accomplishment.

1.2 Research objectives

The research proposed to develop an Enterprise Operating System is explorative. The work performed is still in its preliminary stage. The main objective is to design the envisioned EOS and to test its feasibility through case examples for the proof of concept.

The EOS we are proposing to develop is seen as an alternative approach that aims at combining existing enterprise integration and ERP approaches to monitor and control enterprise operations. It will manage and interpret the enterprise model and requests defined by business managers and activities owner, trigger various enterprise operations with dynamically allocated enterprise resources, control the processes execution, and monitor the status of enterprise resources (Human, Machining, Computing) through various sensing devices and front-ends (Burnson 2015; J. Gray and Reuter 1993).

More precisely, the EOS and existing ERP solutions have similar objectives but different approaches. ERP is a top-down integrated approach that includes in its framework the main enterprise internal applications and operating functions. While EOS is a bottom-up federated approach only providing enterprise operating system (functions) that allows divers heterogeneous and multi-vendors applications to connect to EOS and to operate together. From this point of view, EOS provides more options for enterprises to implement tailored solution.

It is important to note that implementing an EOS in a company does not mean that the EOS will monitor and control all the resources and all operations carried out in the

company. This depends on the business of each specific company and the degree of automation of that company. In other words, a company will have to decide what operations and resources that are important for its functioning to reach its targeted performance. Broadly speaking, only operations defined in an enterprise model and the resources required in the model will be controlled by the EOS. This decision is left to enterprise's users.

Main research challenges in developing the EOS can be stated as follows:

- To solve some interoperability problems if heterogeneous enterprise software applications need to be connected to EOS and worked together. Although many researches and solutions (ATHENA 2003) have been developed in recent years, some specific interoperability utility services (ESoCE-Net 2012) still need to be developed and/or adapted in the context of EOS.
- To identify and define main functions / capabilities that an EOS must provide. This issue is also closely related to a question: what are the requirements on EOS? To answer this question, at least the three basic aspects must be covered: Enterprise activities interoperation, monitoring and control.
- To find a suitable technology to implement EOS so that it technically works. There are several existing technologies that would implement the EOS. One of the criteria is that the chosen solution should facilitate the interoperability of market available enterprise applications.

With a lower cost comparing to ERP, a SME equipped with an EOS will only deploy and install its needed software applications potentially coming from different vendors. It is no doubt that the success of the EOS will depend on potential industry recognition. Once EOS adopted in industry, a foreseen ecosystem would be developed in consequence to provide varieties of enterprise applications compatible to EOS, just like what happened with Apple's ecosystem to IOS and Google's one to Android. To make this happen, the envisioned EOS needs to be recognized as a standard so that enterprise solutions vendors develop their applications according to the specifications of the EOS.

1.3 Contribution of the thesis

Developing an EOS is a long research endeavor. As a contribution to this end, this doctorate research is mainly focusing on the early phases of EOS engineering lifecycle:

- To identify a set of requirements and functionalities that the envisioned EOS

must meet and satisfy.

- To elaborate EOS architecture at various abstraction levels: the conceptual, technical and implementation architectures that support the various phases of EOS engineering lifecycle.
- To develop a simplified prototype implementing EOS architectures so that the EOS concepts can be tested and validated through case studies.

Through the development of the EOS, the expected benefits can be summarized as follows:

- Allowing enterprises (especially the future ones based on IoT and Cyber physical principle) to dynamically monitoring and controlling their operations based on real-time data
- Reconciliating two different but complementary initiatives for enterprise management (IT platforms/infrastructure and ERP based application packages)
- Reducing the implementation cost, efforts and delay of enterprise integration and enterprise processes orchestration
- Improving data protection, efficiencies and decision-making with better interoperability of heterogeneous software applications within the organization.
- Providing a bottom-up approach with more options for enterprises to implement tailored solution, especially for SMEs.

1.4 Organization of the thesis

This thesis is organized as follows:

- Chapter 1 gives a general introduction of the thesis. It presents the context and problems, the objective and the contribution of this research as well as the overall structure.
- Chapter 2 provides the basic concepts and requirements that are relevant to the development of an Enterprise Operating System. This includes definitions and architectures of a computer operating system and EOS, distributed simulation concepts, enterprise interoperability approaches as well as the functional requirements to develop an EOS
- Chapter 3 presents the state-of-the-art on models and architectures relating to enterprise integration that are relevant to define the EOS architecture. Comparison between those various approaches are also provided.

- Chapter 4 presents the EOS architecture for the development of an Enterprise Operating System. The EOS architecture is defined at three levels of abstraction: conceptual, technical and implementation.
- Chapter 5 is concerned with two case studies in the Banking and Manufacturing sectors. These cases aim at showing the feasibility of the EOS concepts proposed in chapter 2, and illustrate the functioning of the technical and implementation architectures elaborated in chapter 4.
- Chapter 6 finally gives the general conclusions, discusses the achievements and main advantages, and points out perspectives for possible future works.

Figure 1 presents the organization of the thesis. From top to bottom: the domains studied, the research problems and the contributions provided.

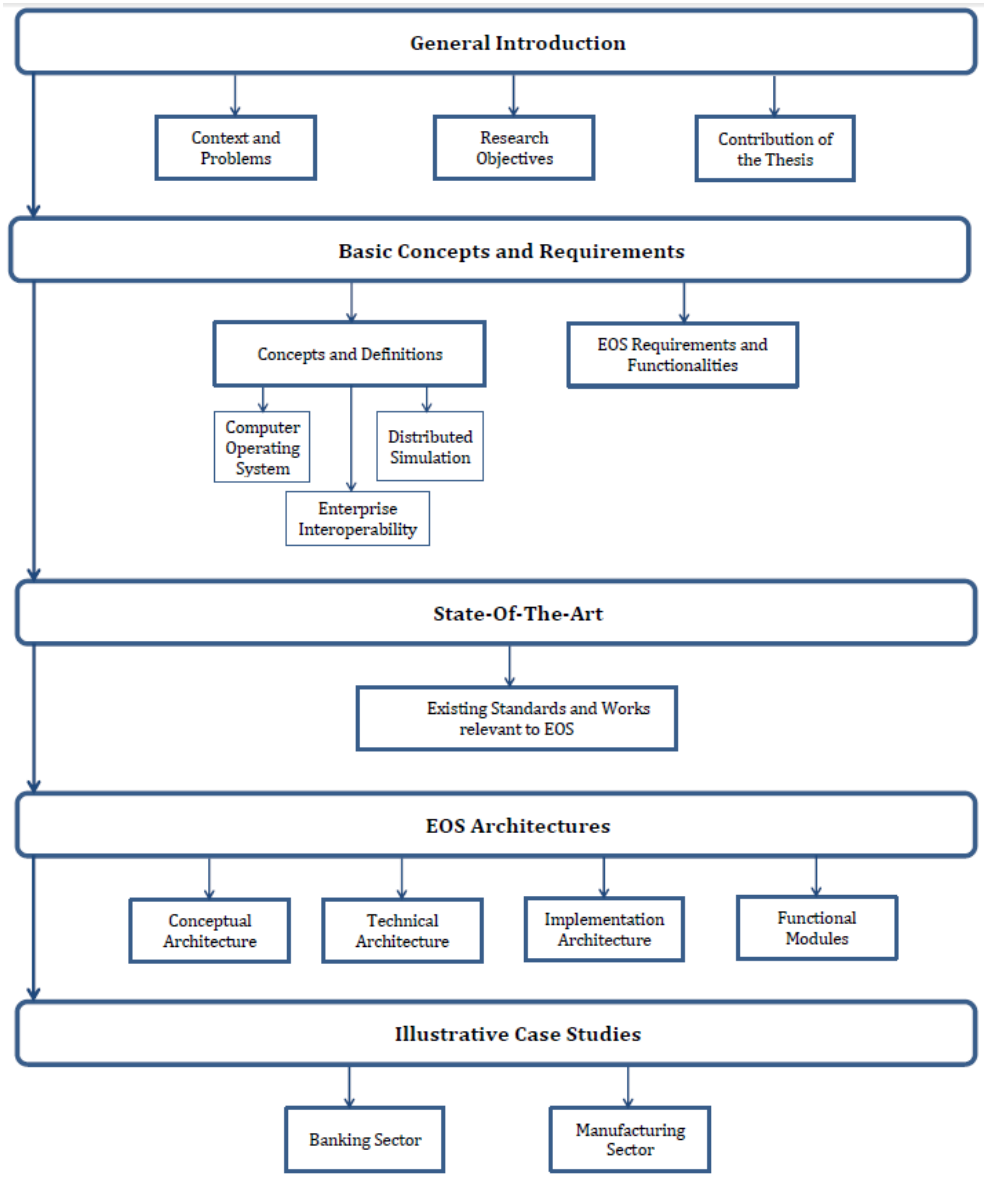

Figure 1. Overview of the thesis' structure

Chapter 2. Basic Concepts and Requirements

The concept of Enterprise Operating System is inspired from the OS in computer domain as an alternative to ERP. This chapter first review the definitions and architectures of computer operating system as well as the concepts of distributed simulation techniques useful to implement EOS. An analogy between computer OS and EOS is also given with the proposed definition of EOS. The concept of interoperability and models of enterprise interoperability relevant to EOS development are overviewed. The last part of this chapter presents the requirements and functionalities that the proposed EOS must meet and satisfy.

2.1 Computer Operating System

On the first computers, with no operating system, every program needed the full hardware specification to run correctly and perform standard tasks, and its own drivers for peripheral devices. The growing complexity of hardware and application programs eventually made operating systems a necessity for everyday use (G. Gray 2014).

The first operating system used for real work was GM-NAA I/O, produced in 1956 by General Motors' Research division for its IBM 704 (Fiedler 1983; Patrick 1987).

Early operating systems were very diverse as each vendor produce a specific operating system related to their particular mainframe computer with different models of commands, operating procedures, and debugging aids. Typically, each time the manufacturer brought out a new machine, there would be a new operating system, and most applications would have to be manually adjusted, recompiled, and retested.

Through the 1960s, several major concepts were developed, driving the development of operating systems such as the development of the IBM System/360 produced a family of mainframe computers available in widely differing capacities, for which a single operating system OS/360 was planned (Deitel, Deitel, and Choffnes 2003).

Today, command line interface operating systems can operate using only IT external components. The appropriate choice for personal computers has been largely limited to the Microsoft Windows family and the Unix-like family of operating systems (G. Gray 2014). The operating systems (Android, Microsoft Windows, Linux…) are found on many devices such as cellular phones, video game consoles and computers.

2.1.1 Basic definitions

Several definitions were found describing and defining the role of an Operating System “OS”.

(Bic, Lubomur F. and Shaw, Alan C. 2003) defined the “OS” as an intermediary between application programs and hardware functions (input / output devices, memory allocation), developed in order to manage computer hardware and software resources, to provide common services for computer programs, and to schedule tasks for efficient use of the system including mass storage, printing and other resources.

On the other hand, (Beal 2015) settled the previous definition by explaining the interaction of the “OS” with the computer controls such as recognizing input from the keyboard, sending output to the display screen, keeping track of files and directories on the disk, and controlling peripheral devices such as disk drives and printers. The OS manages the execution of different programs and processes running at the same time to prevent the interference with each other, and also ensure that unauthorized users cannot access the computer system.

The Operating System “OS” is defined by (Al-Shaikh 2005) as the computer’s master program which sets the standards for all application programs to run and operate. It includes a command line interface which executes the technical operations by programmers and administrators, a job management which controls the time and sequence that applications are run, a task management which allows the execution of multiple programs (multitasking), data management which keeps track of the data on disk and all storage devices by interacting through the programming interface “API” in order to retrieve and save data by opening / reading / closing files, a device management which controls peripheral devices by sending them commands in their proprietary command language.

(Bassi 2017) added a value to the operating system by explaining its role in case of multi-processors and multi-users, so the “OS” allows for multiple users to use the same computer at the same or different times and supporting and utilizing more than one computer processor.

(Franklin and Coustan 2000) highlighted on the role of the “OS” in case of multiple development languages where it can manage the computer's software and allows users to communicate with the computer without knowing how to speak the computer's language. The OS coordinates all kind of operations and tasks and organizes the

execution and time of each operation.

(Rouse 2014) mentioned the role of the graphical user interface in the "OS" which allows the use of icons or other visual indicators to interact with electronic devices rather than using only text via the command line.

(A. Silberschatz, Galvin, and Gagne 2009) highlighted on a component built in the "OS" that runs at all-time called Kernel, and it is implemented in order to execute user programs, make solving resource problem easier, make the computer system convenient to use and controls the execution of processes to prevent errors.

Based on the above definitions, the Enterprise Operating System "EOS" will be developed with the same concept of the "OS" to operate an enterprise instead of a computer. It will be presented as a boot program acting as an intermediary between the enterprise internal and external components, and by taking into consideration the security part by including an authentication system which is capable to distinguish between privileged of the resources and also between requests which should be allowed to be processed and others.

Moreover a Graphical User Interface "GUI" is needed which views the directory structure and requests the services from the "EOS" by allowing the use of icons or other visual indicators to interact with electronic devices rather than using only text via the command line. Moreover, a "EOS" password protection tool needs to be added to keep unauthorized users out of the system by maintaining activity logs and providing backup and recovery routines for starting over in the event of a system failure.

2.1.2 Architectures of Computer OS

Several proposals were presented for describing the "OS" architecture including its related internal and external components and the interaction process between them.

(Shaun 2009) defined the "OS" architecture (shown in figure 2) by defining the main components: Process Manager, Memory Manager, Device Management, File Management, Network Management and the Graphical User Interface, which provide an efficient intercommunication between systems and controls by creating / deleting processes, allocating CPU to processes, and allocating / deallocating memory space.

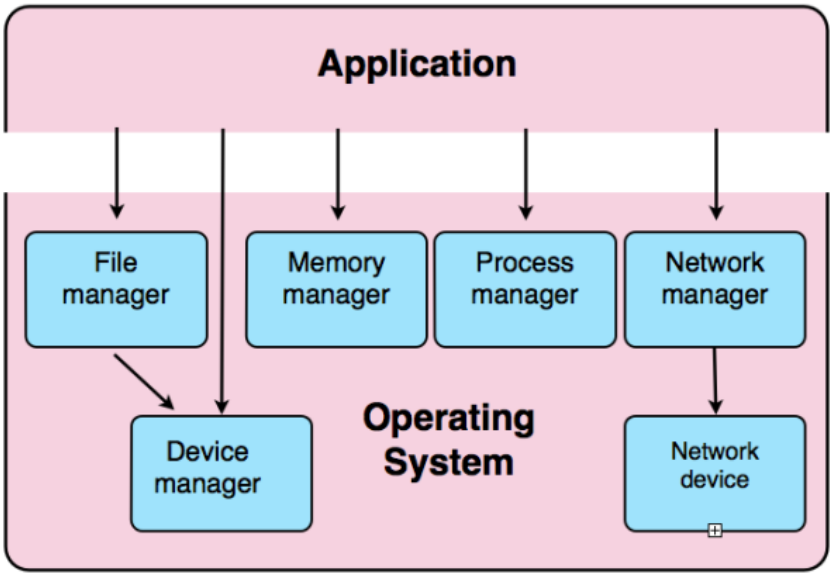

Figure 2. Architecture posted in Word Press journal (Shaun 2009)

(Abraham Silberschatz, Gagne, and Galvin 2006) mentioned in the "OS" diagram (figure 3) that the Computer Operating System provides environments in which programs run including: The User Interfaces (GUI, command line interface and batch command) that allow users to issue commands to the system. The Program Execution which permits to the OS to load programs into RAM, run the program and terminate it. The I/O Operations including keyboards, terminals, printers and storage devices. The File-System Manipulation that helps to maintain directory / subdirectory structures and mapping file names to specific blocks of data storage. The Communications performed between processes or on the same process. The Error Detection in order to detect and handle hardware and software errors with a minimum of harmful repercussions. The Resource Allocation including the CPU, main memory, storage space, and peripheral devices. The Accounting keeps track of system activity and resource usage. The Protection and Security that help to prevent harm to the system and to resources, either through wayward internal processes or malicious outsiders.

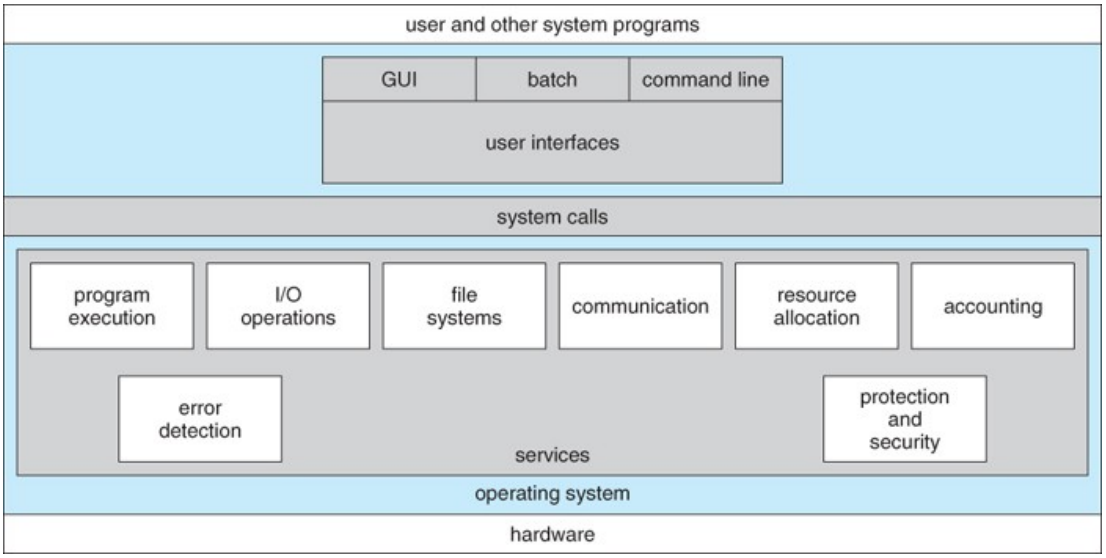

Figure 3: Architecture posted in "Operating System Concepts" paper - Ninth Edition (Abraham Silberschatz, Gagne, and Galvin 2006)

The "Windows" "OS" is defined and described by (Russinovich and Solomon 2007) as presented in figure 4, which provides several services in order to organize and ensure the communication between different system resources. It consists of: The Environment Subsystems that provide exposed and documented interface between applications and Windows APIs; it consists of the environment subsystem process and the kernel-mode / graphics device drivers. The Kernel Mode is the privileged mode of operation in which the code has direct access to all hardware and all memory including the address spaces of all user mode processes; it is comprised of the Windows Executive which includes the Executive Services, the Kernel and the Hardware Abstraction Layer.

The Windows Executive is the upper layer which provides core OS services and contains major components such as various objects, security, processes and virtual memory. The Kernel is the lower layer which provides the most basic operating system services such as thread scheduling, first-level interrupt handling and deferred procedure calls. The Hardware Abstraction Layer is a loadable kernel-mode module that enables the same operating system to run on different platforms with different processors.

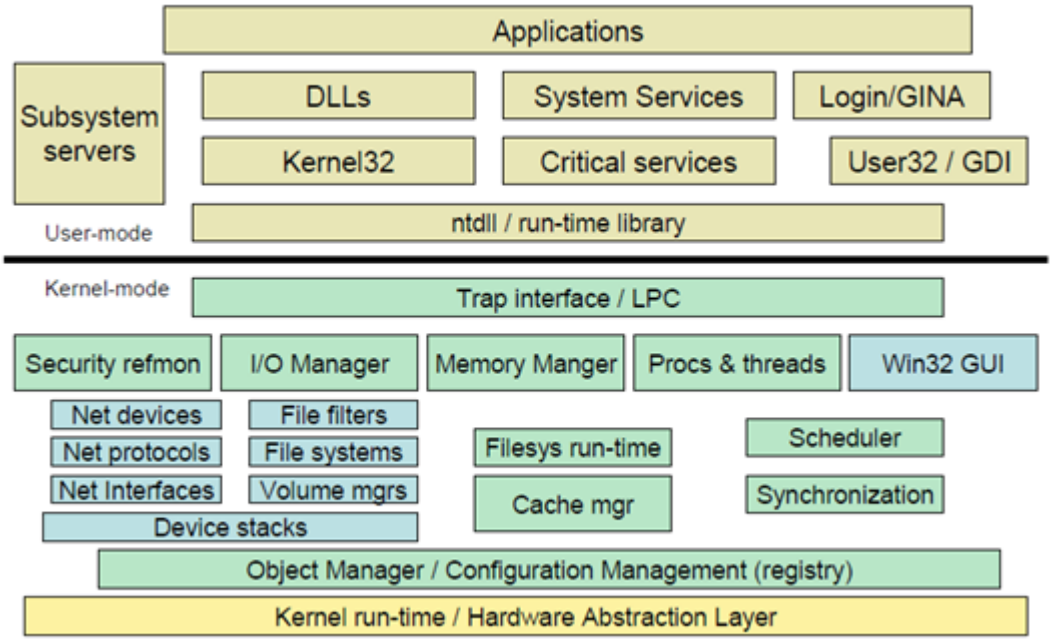

Figure 4: Architecture posted in "Windows Internals" paper - 4th edition (Russinovich and Solomon 2007)

(Anguish 2016) presented the "Mac OS X" diagram (figure 5) which is the latest operating system for Apple's Macintosh computers. It is designed as a development platform that supports multiple development technologies including UNIX, Java, the proprietary Cocoa, Carbon runtime environments and a host of open source. Mac OS X is developed with an updates Kernel and Driver Model.

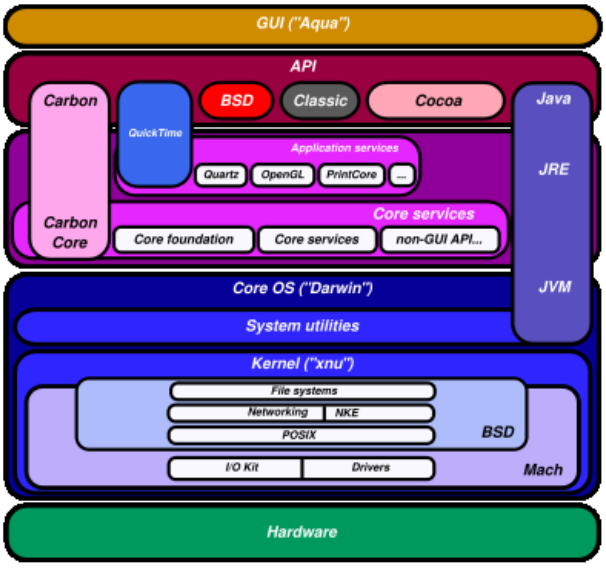

Figure 5: Architecture posted in a "WWDC 98 Summary" paper (Anguish 2016)

Based on the above architectures, the Enterprise Operating System will be developed using the similar concept of interaction of the "OS" modules after defining and

implementing the “EOS” components. However, because of the specificities of enterprise systems compared with computer systems, necessary adaptations are also needed based on the requirements and functionalities that an EOS must satisfy.

2.2 Concepts of Distributed Execution

2.2.1 Service Oriented Architecture Approach “SOA”

The Service Oriented Architecture “SOA” is an architectural style defined by the Open Group in order to support service orientation including service-based development and outcomes services. This technique involves the interaction between resources that function independently and enables networked applications to cooperate and exchange information. It permits for users to combine functionalities to form applications from existing software services and controlling the usage (delivery, acquisition and consumption) of related services (The Open Group 2012; AIIM 2000); however, this approach presents several relevance and insufficiencies in the concept of distributed simulation and specially in matching required capabilities to services, ensuring the synchronized simulations between entities, and maintaining the data interoperability of divers data sources.

2.2.2 Distributed Simulation

In order to reduce execution time, integrate simulators executed on different manufacturers, ensure synchronization and interoperability of simulations, and provide real-time execution for “EOS”, the distributed simulation “DS” concept should be implemented based on a specific algorithm and protocol that can provide the required facilities for an efficient interaction between federates.

The Distributed Simulation is presented as a collection of logical processes (LP) interconnected by a communication network (LAN or WAN), modeling different parts of the physical system, executing a single simulation program, and providing a sufficiently realistic environment into which users, possibly at geographically distant locations, can be embedded (Training, Entertainment, Social Interaction…) (Kuijpers et al. 2005).

The following algorithms and protocols are defined as accepted standards for interoperability among separately developed simulators.

2.2.2.1 Distributed Interactive Simulation (DIS)

DIS is a government/industry initiative to outline an infrastructure for linking simulations of various types at multiple locations to create realistic, complex, virtual worlds for the simulation of highly interactive activities (IEEE 1995, 2017). As appeared in figure 6, the DIS network can understand the correspondence among various frameworks built for particular purposes, with various innovations, and giving diverse products services, so that they can interoperate. A standard set of Protocol Data Unit (PDU) has been defined for describing the format of messages exchanged between participating simulation hosts. The individual simulation host has a dis_mgr, which is PDUs dispatcher between the DIS network and divers programs. The client-server protocol executed between the dis_mgr and application programs use TCP/IP (Transmission Control Protocol/Internet Protocol) to interchange data (Zhiying, Chen, and Zacharewicz 2012). The connection between the DIS network and dis_mgr is based on UDP/IP (User Datagram Protocol/ Internet Protocol). Once the simulation host changes its status, it will broadcast a message to all other members.

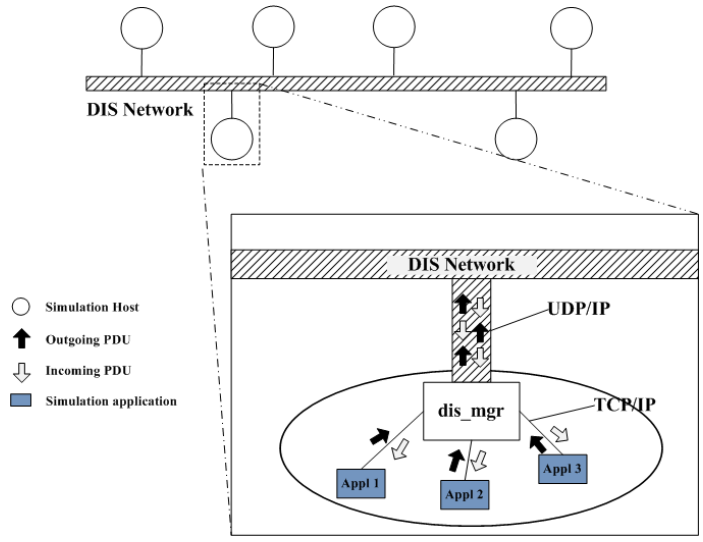

Figure 6: Distributed Interactive Simulation

2.2.2.2 Aggregate Level Simulation Protocol (ALSP)

The ALSP is under the auspice of the Advanced Distributed Simulation, which is the nomenclature emanating from the U.S. Department of Defense. It provides a mechanism for the integration of the existing simulation models to support training via theater-level simulation exercises (Weatherly, Wilson, and Griffin 1993).

Similar to DIS, ALSP describes a collection of infrastructure software and protocols for passing the messages between the various participants of a distributed simulation.

Different from DIS, ALSP has global time synchronization and use object-oriented approach to describe the shared object model of a distributed simulation.

Both DIS and ALSP provide a protocol of the distributed systems communication. Both of them have proven successful in supporting the interoperation of disparate systems/platforms/services, and the ALSP even starts to take into account the time issue. However, they still cannot support time management and data distribution management.

2.2.2.3 High Level Architecture (HLA)

The 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 Defense Modelling and Simulation Office (DMSO) of the US Department Of Defense (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 (Zacharewicz, Chen, and Vallespir 2008, 2009).
The Run Time Infrastructure (RTI) is the supportive middleware for the distributed simulation. It is the fundamental component of HLA. It provides a set of software services for the dynamic information management and inheritance, in which federates coordinate their operations and exchange data during a runtime execution.

HLA software architecture is a supportive framework for the distributed simulations and will be implemented for "EOS" in order to specify the object model used by the federation, provide facilities for allowing federates to interact with each other and managing the execution (Federation management, Time management, Data Distribution management,..), enable interoperability among different operating systems, and allow events to interact in an EOS.

2.3 Enterprise Integration (EI)

The Enterprise Integration (EI) is mainly defined as a concept in enterprise engineering to provide the right information at the right place and at the right time and thereby enable communication between the enterprise's computers and applications and

their efficient co-operation and co-ordination (F. B. Vernadat 1996).

During the 80's, Integration of software systems has entered among the top priorities of many organizations as a technical field of enterprise architecture, which focused on the study of topics such as system interconnection, electronic data interchange, and product data exchange and distributed computing environments (Petrie 1992).

In the service and manufacturing enterprises, enterprise integration may be regarded as an extension of computer integrated manufacturing that integrates financial or executive decision-support systems with manufacturing, tracking and inventory systems, product-data management, and other information systems (CIMOSA 2009).

2.4 Enterprise Resource Planning solution (ERP)

The Enterprise Resource Planning (ERP) is a platform that allows the organization at the operational level to use a system of integrated applications in order to automate many back office functions related to technology and services, by collecting, storing, managing and interpreting data from the business activities. This module can deliver improved enterprise application integration by offering an integrated suite of applications to perform standard business functions (Burnson 2015; Diercksen 2012).

At its most basic level, ERP software integrates these various functions into one complete system to streamline processes and information across the entire organization. The central feature of all ERP systems is a shared database that supports multiple functions used by different business units.

2.5 Enterprise Operating System (EOS) earlier concepts

Since many years, many projects started to search in order to define, requirements and functionalities for the development of an enterprise operating system that can cover the enterprise's operations.

(Jonah 1999) proposed the "EOS" as a multithreaded and multitasking framework that allow installation of new applications without a system shutdown and to remain running in the event of an application crash. It must be able to scale up to multiple processors in either a symmetric multiprocessing or massively parallel processing configuration. It includes an object architecture to provide communication between software components and should support remote distribution, installation and configuration of software, off-site backups and routine maintenance.

On the other hand, (Mstrohlein 2011) defined the Enterprise Operating System as a program providing the services that business functions and operations run on. It is comprised of people, organization and culture, leadership, technology, and processes (how an organization gets things done). Enterprise OS is lean, containing only the essential moving parts necessary to support the business.

(Shaughnessy 2011) believed that any developed "EOS" should be around anonymous and automated business relationships that waste no time. People congregate in organization not in employment pools in order to create value; they interact on platforms that scale global interaction and relationships. EOS should be based on five pillars to allow companies allocating resources and managing scaled relationships in much better ways: 1) The universal connector: through API, RSS and Open Graph people can collaborate, enter a standardized contractual relationship and grow a line of business in a very low friction and low cost way. 2) The new business ecosystem: includes the community of commentators as well as suppliers and reputation risk, producing goods and services of value to customers, who are themselves members of the ecosystem. 3) The business platform: facilitates massively scaled human relationships in relatively open environments. 4) The cloud infrastructure: patches together new business lines from multiple partner offers and shows cloud at the core of new business development. 5) Sapient leadership: orchestrates ecosystems as well as knows how to parley technology into new strategies.

However, those initiatives failed to bring the concepts to market available EOS system, most of them stopped at the concepts level.

2.6 Comparison between concepts

2.6.1 EI, ERP and the proposed EOS concepts

The Enterprise Integration (EI) has been developed and implemented in the 1980's to be able to rapidly respond to the changing environments, the progress in the information technology, and the growth and complexity of the enterprise operations (Linthicum 1999; Gable 2002). However, the integration of different systems required rewriting codes on source and target systems, consuming much time and money. Moreover, this concept failed to ensure the full integration and intercommunication and especially between enterprise's heterogeneous applications and their related operating systems (as illustrated in figure 7-A).

During our research we noticed that a good number of enterprises have chosen ERP solution in order to facilitate the data orchestration and especially in the 1990's and 2000's (Diercksen 2012); however, this platform presents several major insufficiencies in providing the federated interoperability, and because of becoming enormously in size and complexity which make the implementation and the updates very time consuming, involving high costs and disrupt current processes and operations.

As shown in figure 7-B, the ERP module is basically presented as a black-box application implicating built-in services and modules used in order to manage and control the enterprise's operations and activities; however, ERP does not ensure an efficient integration between its built-in applications. Moreover, this platform cannot ensure the compatibility and effectiveness of the direct intercommunication between enterprise resources.

On the other hand, the proposed EOS, which we expect being implemented in the market and industry during the 2020's (Grabot 2012; Aurousseau et al. 2013), aims at monitoring and controlling enterprise resources and the operations carried out by those resources. After performing the messages processing by the enterprise resources, the EOS acts as a central orchestrator that intercommunicate and interconnect with the enterprise's IT, Human and Machine dialogues, in order to monitor enterprise components, applications and operations, and allocate resources to required activities based on the interoperability technique (as shown in figure 7-C).

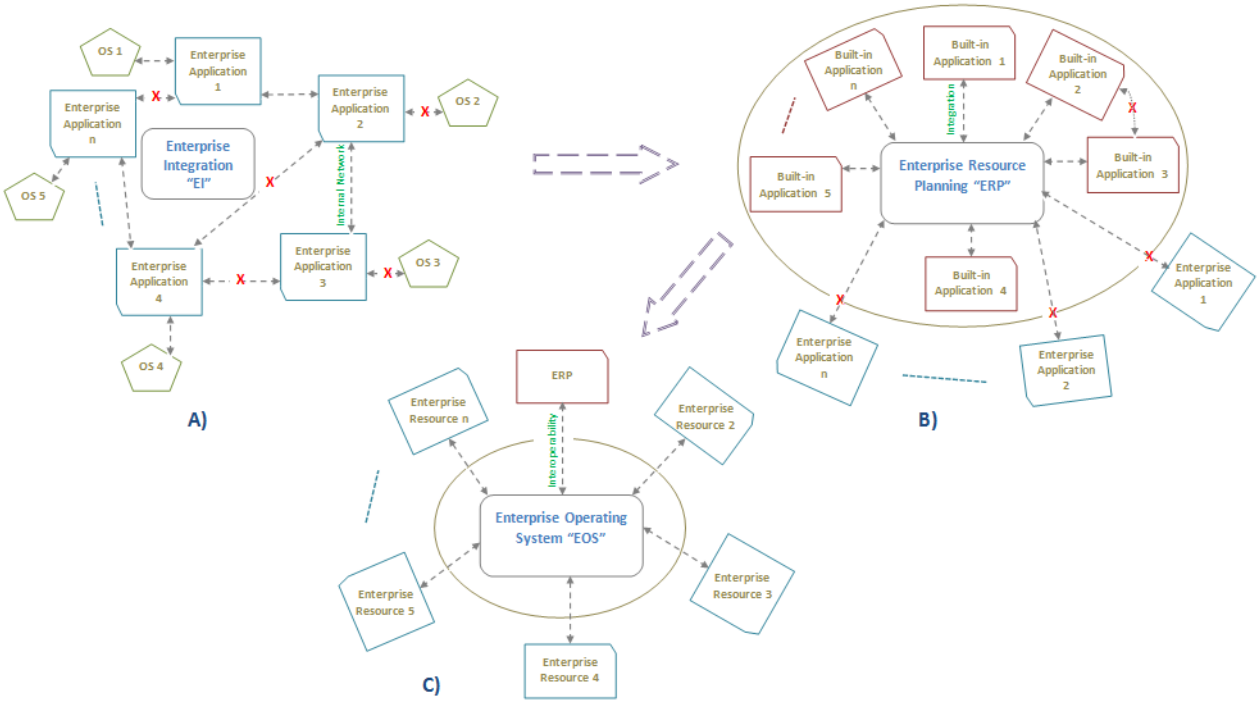

Figure 7: Comparison between EI, ERP and the proposed EOS

2.6.2 OS and the proposed EOS concepts

Before giving our definition on EOS, let's see table 1 that shows the analogy between a computer OS and an EOS to highlight the similarities and differences of these two concepts.

	Computer OS	EOS
Purpose	To control and monitor IT components / resources connected to OS	To control and monitor enterprise resources connected to EOS
Where implemented	- Local - Inside a computer	- System-wide - On top of computers
Who use	Users of a computer	Business managers and enterprise system operators
Resource monitored /controlled	- Hard disk reader/writer - Mouse / keyboard - Printer / scanner - Monitors - Data storage devices - etc.	- Computing devices - Machining devices (NC machines, robots, transport systems, autonomous production systems, artificial agents etc.) - Humans (managers, designers, shopfloor operators)
Operations monitored / controlled	- Input commands - CPU computing operation - Print / scan - Read / write data - etc.	- Input commands - Engineering operations - Manufacturing operations - Management operations - etc.
Processes	- Internal processes to coordinate and synchronize operations of internal and external components	- Business processes - EOS internal processes to coordinate and synchronize operations of EOS services

Table 1 : Analogy between computer OS and EOS

Both OS and EOS have the same objective as to act as an interface between system's users and system itself constituted by a set of resources. Although the resources monitored and controlled by OS and EOS are different in nature, they must all receive commands from OS/EOS to start the operations and feedback the status when

operations are terminated. Evidently the operations performed by a computer system are much simple and can be entirely automated, while operations performed by an enterprise system are not. It is important to note that an OS/EOS will only monitor and control the start and end of operations, not how those operations are performed (automatically or manually).

In the context of industry 4.0, the machines that will work as autonomous production systems, and more generally artificial agents will also need to access collective information (and commands) to efficiently operate.

To conclude, the EOS can be defined as "a set of IT services supporting execution, control and monitoring of enterprise operations by means of message exchange among enterprise resources, model enactment and interoperability capabilities".

2.7 Concept of enterprise interoperability

Broadly speaking, the enterprise interoperability is an ability that can support the communication and transactions between heterogeneous and networked enterprises based on shared business references. The interoperability is considered as significant if the interactions can take place at least at the three different levels: data, services and process, with a semantics defined in a given business context.

From technical point of view, interoperability is the ability to understand and use the information provided another system without interfacing efforts.

It is worth noting the difference between integration and interoperability. The enterprise integration is the process of ensuring the interaction between enterprise entities necessary to achieve domain objectives, while enterprise interoperability refers to the ability of interactions (exchange of information and services) between enterprise systems (Zhiying, Chen, and Zacharewicz 2012). Interoperability has the meaning of coexistence, autonomy and federated environment, whereas integration refers more to the concepts of coordination, coherence and uniformization.

Incompatibility is the fundamental concept of interoperability which is defined as the obstacle to establish seamless interoperation. The framework for enterprise interoperability initially developed under Interop NoE and adopted as an international standard has identified three main categories of incompatibilities called 'barriers' (Chen 2009).

- *Conceptual Barriers* are concerned with the syntactic and semantic incompatibilities of information to be exchanged.

- *Technological Barriers* are concerned with the use of computer or ICT to communicate and exchange information.
- *Organizational Barriers* are concerned with the incompatibilities of organization structure and management techniques implemented in different enterprises such as the way of assigning responsibility and authority.

Furthermore, interoperations in the enterprise context can take place at the various levels of concern as shown in the framework (see figure 8), namely:

- *Data interoperability:* it is concerned with finding and sharing data originating from heterogeneous information bases and which can moreover reside on various machines with various operating systems and data bases management systems.
- *Service interoperability:* it is concerned with identifying, composing, and inter-functioning of various applications (designed and implemented independently) by solving the syntactic and semantic differences, as well as finding connections to various heterogeneous databases.
- *Process interoperability:* it aims at connecting distinctive process description to form collaborative processes and perform check, simulation and execution. Usually different process description languages are used to define different process models for different purposes.
- *Business interoperability:* it refers to working in a harmonize way at the levels of organization and company in spite of the different modes of decision-making, working methods, legislations, culture of the company and commercial approaches.

To solve interoperability problems, three basic admitted approaches can be used (ISO 14258 1999; Chen 2009).

- *Integrated approach:* it requires a typical configuration for every single constituent system. Divers models are interpreted in the common format. This format must be as rich as the constituent system models.
- *Unified approach:* it requires a typical predefined format in meta-level. This format covers across the constituent models, providing a means for establishing semantic equivalence.
- *Federated approach:* it requires that the models must be dynamically

accommodated rather than having a predetermined meta-model. To establish interoperability, parties must accommodate 'on the fly'.

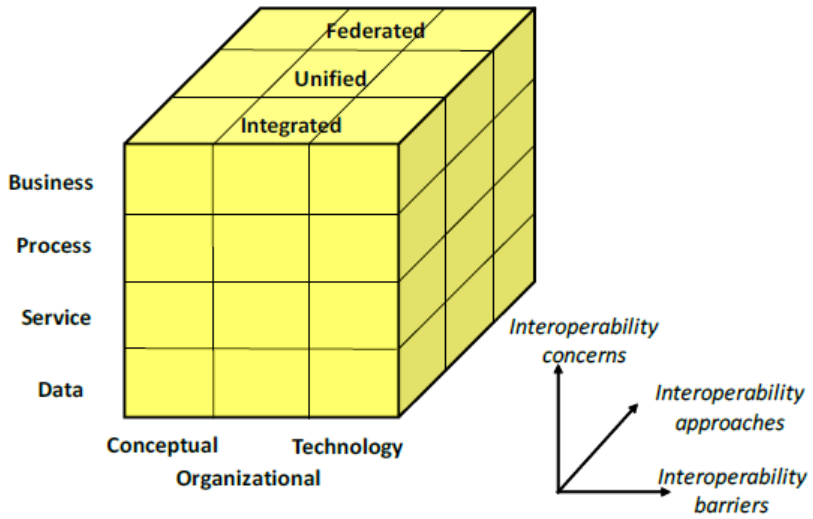

Figure 8: INTEROP Enterprise interoperability Framework

2.8 Methods, Architectures and Models for Enterprise Interoperability

Several models and techniques have achieved some success in developing Systems Interoperability. However, none of them proposes the complete solution for all the interoperability issues.

2.8.1 Model Driven Architecture (MDA)

MDA is a vendor-neutral approach to interoperability designed to promote the use of models providing a systematic architecture to model a system. The three major objectives of MDA are portability, interoperability and reusability (Zhiying, Chen, and Zacharewicz 2012). The MDA provides a set of guidelines for the structuring of specifications supporting model-driven engineering. It begins with the outstanding and long established idea of isolating the specification of the operation of the framework from the points of interest of the way the framework utilizes the capacities of its product execution (OMG 2003). It builds on six basic concepts: System, Model, Architecture, Viewpoint, View and Platform.

The MDA approach contributes on building an interoperable ICT model, from enterprise models to technology models. Those models are able to be aligned by using common meta-model. MDA also provides flexibility and adaptability to accommodate changes at a higher abstraction level. Furthermore, Model transformation ensures the

interoperability achievement and/or agreement from higher level to infrastructure (lower level) (OMG 2003). Besides that, it allows document transformations on the fly, and can contribute to new approaches for semantic interpretations on information exchanges. However, no matter how many advantages MDA has, there are still many people doubt on its performance in practice (Bézivin et al. 2003).

2.8.2 Architecture Driven Modernization (ADM)

MDA is notable for advancing the utilization of models and their transformations to design and implement distinctive data systems. The fundamental thought proposed in the MDA approach is to interpret from an abstract platform-independent model (PIM) communicated in UML into a more solid platform-specific model (PSM) from which the code still needs to be generated (OMG 2010). Reversing the MDA lifecycle, ADM is finding models from the coding level of legacy information system, for example, UML models, Knowledge Discovery Meta-display (KDM) and Abstract Syntax Tree Meta-demonstrate (ASTM). KDM and ASTM are intended to fulfill someone interested on finding more particular models from a legacy system (OMG 2003).

ADM shows its strong power in obtaining information from the legacy systems, but many people doubt on the validity of this information for achieving federated enterprise interoperability (Zhiying, Chen, and Zacharewicz 2012). ADM met the same model transformation problems as MDA. In addition, most of the current researches are focus on obtaining static models from the existing systems which cannot fully describe the systems. Most of the time, the reversed models can only be a guideline for the system reconstruction. Thus, the model reverse engineering did not achieve its real intention (De Lucia, Fasolino, and Pompella 2001).

2.8.3 Enterprise Interoperability Standards

This section presents two major existing enterprise interoperability standards (RMI and HLA); these two models are mainly detailed in order to explain their compatibility with the proposed EOS and how much they can ensure the business interoperability in case they are implemented with EOS.

2.8.3.1 Remote Method Invocation (RMI)

RMI was developed and updated in order to allow the software engineers to

compose object-oriented programming in which questions on various computers can collaborate in a distributed network (Buss and Jackson 1998).

As appeared in figure 9, the RMI system comprises of three layers:

- *The stub/skeleton layer:* client-side stubs (proxies) and corresponding server-side skeletons. The stub appears to the calling project to be the program being required for a service.
- *The remote reference layer:* remote reference behavior that can be distinctive relying upon the parameters passed by the calling program. (e.g. invocation to a single object or to a replicated object)
- *The transport layer:* connection set up and management and remote object tracking

The client utilizes the stub (proxy) to summon a strategy on the remote server. The nearby stub is an implementation of the remote interfaces of the remote object. It holds a reference to the remote object and advances the conjuring solicitations to the server by means of the remote reference layer. The remote reference layer is in charge of doing the semantics of the summon. The transport layer takes accountable for connection set-up and management. It likewise monitors the remote objects (the targets of remote calls) and dispatches them to the transport's address space.

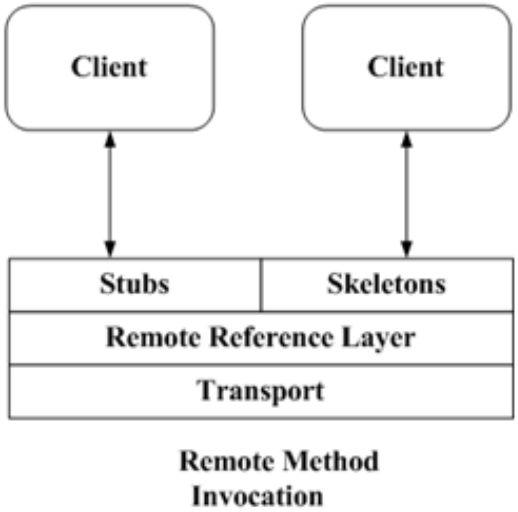

Figure 9: RMI Structure

RMI can strongly support the interoperation of software application on in a distributed environment; however, it cannot provide advance simulation services, such as integrated time management, interest specification, ownership management and data distribution services. Without these services, it is very hard to create a flexible and

adaptable interoperability environment.

2.8.3.2 High Level Architecture (HLA)

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 (Elvesæter et al. 2007).

The Run Time Infrastructure (RTI) is the supportive middleware for the distributed simulation. It is the principal segment of HLA. It gives an arrangement of software services for the dynamic data administration and inheritance, in which federates facilitate their operations and trade information during a runtime execution. As indicated by the HLA interface specification, RTI offers six management services: Federation management, Time management, Declaration management, Object management, Ownership management and Data distribution management.

As shown in figure 10, A HLA federate has two parts, federate code and local RTI component code (LRC):

- *Federate Code:* has to extend and implement RTI with Local RTI Component Code from the C++ library LibRTI: Federate Ambassador contains pure virtual functions for each possible callback.
- *LRC:* provides the services for the federate through communication with the RTI executive component. Those services can be obtained by calling the member functions of Class RTI: RTIAmbassador, which is contained in the LibRTI.

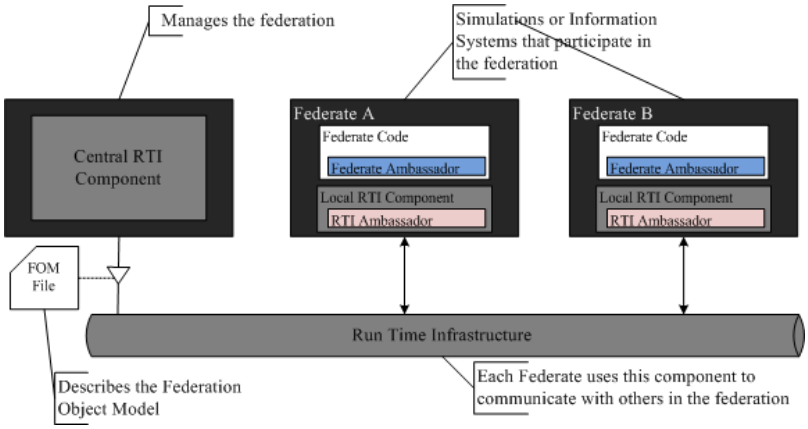

Figure 10: High Level Architecture

Various HLA components, functionalities and services support the interoperation of distributed simulation, but HLA FOM (Federation Object Model) is playing the primary role among them via following ways:

- *Unify information exchange:* FOM defines the shared vocabulary of a federation, which allows federates to work with one another in a defined manner (Pokorny, Stratton, and Smith 2006).
- *Overcome platform differences:* In the HLA simulation, FOM plays as a recipe for the reconstitution of received information into its intended format.

The Distributed Simulations ensured by HLA are necessary useful in order to analyze systems and may help in the enterprise operating systems "EOS" by:

- Managing and controlling executions of processes inside the operating system;
- Reducing time for better performance of the EOS.
- Modelling different parts of the EOS in order to synchronize the execution of different events in the EOS.

Unlike the above concepts and standards, HLA is mainly used in our doctorate research because of providing excellent services for "EOS" such as the capability of achieving interoperability across disparate platform, the reusability of simulation models, time management and secure simulation environment.

2.8.3.2.1 Run Time Infrastructure (RTI)

The Run Time Infrastructure of HLA plays a very significant role and can be considered as valuable solution for EOS in order to integrate internal applications and interoperate with other events to exchange defined messages. It may reduce the development expenses, attenuate the risk and accelerate time to value leading to increasing interoperability among Enterprise Operating Systems.

A few business RTI programming tools exist together, for example, Pitch portable RTI (pRTI), MAK Real-time RTI, poRTIco, CERTI, and OpenRTI.

- CERTI:

CERTI is a free open source software involving a library "LibRTI" linked with each HLA federate, as well as two processes: 1) RTI Ambassador "RTIA" which exchanges messages over the network via TCP sockets and satisfies federate requests. 2) RTI Gateway "RTIG" which manages the creation/destruction of federation executions and delivers messages to the interested RTIAs avoiding a true broadcasting.

CERTI RTI ensures distributed simulations by making fast modifications, adding real-time simulation and federating an international user community. It supports HLA

Evolved and runs on several platforms including various flavors of Linux, Microsoft Windows and Solaris.

CERTI guarantees a secure interoperation of simulations belonging to various mutually suspicious organizations by implementing Trusted Third Party security architecture "TTP". Moreover, it includes features required in order to achieve and increase high-performance simulation of federations running on the same shared memory or on cluster platforms (Noulard, Rousselot, and Siron 2009).

- OpenRTI:

OpenRTI is an open source IEEE 1516 standard Run Time Infrastructure, written by Mathias Fröhlich. OpenRTI operates through several scenarios:

  - The client/server scenario: Having one server process and some federates in several processes on some machines in a network.
  - The slave servers under a parent server scenario: Messages are only routed to specific and required servers.
  - Non server scenario: When all federates are living in one program and may be in different threads, messages are just exchanged in that program.

To select between the different modes of operation above, several protocols are being used by OpenRTI:

  - thread: Just use the in memory communication between threads. This is the default as it does not require much setup.
  - rti: This is the preferred TCP/IP protocol variant for networked communication.
  - pipe: Uses the rti binary protocol on a named pipe. Can be used for machine local communication in presence of tight packet filters.

The implementation is prepared to some degree to move protocols into a user provided shared library that could be loaded at runtime (Callahanp 2013).

- poRTIco:

poRTIco is a fully supported, free open source, cross-platform HLA RTI, implemented in two languages: JAVA and C++, and consists of three components: LRC, Connection Binding and Run Time Infrastructure. poRTIco is licensed under the terms of the Common Developer and Distribution License "CDDL" and is actively

developed and maintained by its team of core contributors (poRTIco 2009; Portico 2013; R. M. Fujimoto 2000).

Focused on modularity and flexibility, poRTIco provides an extensible environment to support HLA simulation development and research and has the potential to greatly expand the use of distributed simulation and training within industries.

As the poRTIco is an open source without central monitor of the HLA federation execution, the central federate is designed to play this role.

poRTIco is chosen for this doctorate research, in light of the fact that Portico is a fully supported, open source, cross-platform HLA RTI implementation. Outlined in light of modularity and flexibility, poRTIco is expected to provide a creation review RTI execution and an environment that can support continued research and development (Doshi and Castellote 2006; R. Fujimoto 2015).

2.9 Process Modeling

The Process Modeling is the key aspect of business process re-engineering. This concept is a mechanism for describing and communicating the current and intended future state of a business process by explaining the key concepts needed to describe what happens in the development process.

The Process modeling addresses the process aspects of an enterprise business architecture, leading to an all-encompassing enterprise architecture. It is responsible of tracking what actually happens during a process, defining the desired processes and how they should/could/might be performed, providing explanations about the rationale of processes, and establishing an explicit link between processes and the requirements that the model needs to fulfill (Rolland 1993).

2.9.1 Business Process Model and Notation (BPMN)

Business Process Model and Notation (BPMN) is a standard for business process modeling that is presented as a method of illustrating business processes in the form of a diagram similar to a flowchart.

The primary goal of BPMN is to provide a standard notation readily understandable by all business stakeholders. These include the business analysts who create and refine the processes, the technical developers responsible for implementing them, and the business managers who monitor and manage them.

It is mainly used in order to support business process management, for both technical and business users, by providing a notation that is intuitive to business users, yet able to represent complex process semantics (White 2013).

2.9.1.1 BPMN Editors

Several software and modules are being used in order to generate a BPMN graphical diagram such as Bonita Studio, Modelio, Visual Paradigm, Metatask, Wrike, Yaoqiang, etc.

Yaoqiang is chosen for this doctorate research, in light of the fact that Yaoqiang BPMN Editor is an open source graphical editor for business process diagrams, compliant with OMG specifications.

This platform is mainly used for generating a BPMN graphical diagram in order to execute from its back-end an XML file for ensuring the implementation of the HLA distributed simulation based on the federated approach (Falcone et al. 2017).

2.10 EOS requirements and functionalities

This section presents an outline of the requirements and functionalities for developing an Enterprise Operating System. They are inspired from and based on the work specified in a pre-European standard known as prENV 13550 "Advanced Manufacturing Technology - Systems Architecture - Enterprise Model Execution and Integration Services" (AMICE 1993). This Pre-European standard benefits from the result of European Esprit project CIMOSA (Kosanke 1995). Five categories of requirements representing five basic functions of EOS have been identified as follows (CEN 1999):

- Enterprise Resources Management (ERM)
- Enterprise Process Management (EPM)
- Enterprise Information Management (EIM)
- Presentation Management (PM)
- Interoperability Management (IM)

2.10.1 Enterprise Resources Management (ERM)

The ability to send commands to appropriate available resources to execute operations and possibly to receive feedback is essential for an EOS. Enterprise

Resource Management is the efficient and effective deployment and allocation of an organization's resources including:

- Human resources: The sets of individuals who make up the workforce of an organization and business sector.
- Machine resources: The types of machines and tools containing one or more parts that use energy to perform an intended action. Machines are usually powered by mechanical, chemical, or electrical means.
- IT resources: The collections of physical elements that constitutes computer and communications systems as well as software applications.

ERM provides a real-time and global view of the 'occupation' of the resources in a company by monitory the enterprise resources system-wide (available, occupied, out-of-order…) (CEN 1999). Main required functionalities of ERM are: checking the availability of the resources and pre-assigning it when available; responding to Event Handling's request to pre-allocate the critical capabilities required by a process; signaling the Process Management that those critical resources are reserved; matching the required capabilities to the capabilities of existing available resources; selecting the appropriate resource as defined by its capabilities; allocating and de-allocating resources; releasing the assigned resource inputs; signaling the Enterprise Process Scheduling that resources have been assigned or de-allocated; managing time and performance management support of enterprise functions and operations; and ensuring that the right resources are allocated to the right place at the right time.

It is to note that only the resources connected to EOS are to be monitored and managed by EOS, for example those ones involved supporting defined processes. IT and automated machine resources are to be connected directly to EOS. Human resources are connected to EOS via appropriate portable interfaces.

2.10.2 Enterprise Process Management (EPM)

One of the important goals of EOS is to support enterprise model execution. An Enterprise Process is a set of sequenced enterprise activities and operations. There are two types of processes to be executed by Enterprise Process Management (EPM) of an EOS:

- Business Processes: these are processes defined by enterprise managers to fulfil engineering or manufacturing missions of the enterprise in order to meet market

demands. These processes are external to the EOS.

- Computing Processes: these are processes defined by EOS developers to fulfil coordination and synchronization activities of EOS. These processes are internal to the EOS to ensure EOS functioning.

Business processes defined by business users will be triggered by events to perform enterprise daily operations. These processes (once created) can be stored in the repository, updated and re-used. Examples of these business processes are: customer order negotiation and confirmation; new product design, production planning, raw material purchasing, finished product delivery, etc.

As for computing processes, the EPM is responsible for interpreting process behavioral, information and resource requirements. The main required functionalities of EPM include: checking authorization to execute a process; sending commands that will trigger the start of processes; retrieving process descriptions from the run-time repository; recording ending statuses of processes (e.g., done, not done, failed…); notifying the rule sets contained with processes to Event Management Registration; activating or terminating the appropriate process; responding to Rule Interpretation notifications when a process is notified as terminated; providing fault management support for enterprise activities; invoking the Condition Monitor service to register conditions that need to be periodically monitored; monitoring the progress of process execution and key performance indicators; responding to unexpected events and exceptions by requesting Rule Interpretation to invoke the pre-registered exception-handling Enterprise Activity; invoking Presentation and Information Management services as necessary to acquire relevant information, and interacting with the Enterprise Resource Management in order to allocate resources, interpret and receive resource statuses (CEN 1999; J. Gray and Reuter 1993).

Noting that the Enterprise Process Management functionalities will be applied and executed through the Yaoqiang module implemented as a BPMN diagram's generator.

2.10.3 Enterprise Information Management (EIM)

This set of functionality is fundamental for all enterprise integration infrastructures and platforms. Enterprise Information Management supports information and data exchanges between all internal entities of the EOS as well as external entities connected to the EOS such as enterprise resources and business

managers (Chen, Youssef, and Zacharewicz 2015; CEN 1999). EIM is required for optimal use of information and its exchange within the organization; for instance, to support decision-making, process execution and day-to-day operations that require the availability of data and knowledge.

The main required functionalities of EIM are: ensuring the centralized management of reference data used; automating and arranging master data management; excluding data duplicates; ensuring and maintaining consistency with the system-wide run-time name space; supporting transparent access to system and user data stored in several heterogeneous database systems; supporting transparent access to data sources and appropriate data storage facility needed for running EOS; supporting maintenance of the consistency and integrity of enterprise data; supporting data and object queries expressed in a standard format; supporting flexible change of the enterprise's information including data definition, configuration control and enforcement of data access rights, and ensuring information and data confidentiality and security to protect from non-authorized access.

2.10.4 Presentation Management (PM)

Presentation Management is concerned with the interactive aspects of an EOS that ensure the interactions between internal and external worlds of the EOS. PM produces a set of interfaces which make it easy (self-explanatory) and efficient to organize and coordinate the communication and information flows between Enterprise Resources connected to the EOS and internal entities of the EOS including the Human, Machine and Application Dialogue services (Chen, Youssef, and Zacharewicz 2015; CEN 1999).

- Human dialogue service: Refers to the interface between the EOS and human agents (managers and operators) receiving and sending information (command, data, instruction, status…) from and to the EOS. The information can be displayed on a computer screen or a mobile device.
- Machine dialogue service: Refers to the interface between the EOS and machine resources (e.g. NC machines, robots, transport systems and any other material transformation devices) receiving and sending information (command, data, status…) from and to the EOS.
- Application dialogue service: Refers to the interface between the EOS and

computing devices (e.g. computers, applications, databases…) receiving and sending information (command, data, status…) from and to the EOS.

This function is mainly required for: transposing the control and information inputs into the appropriate forms for the involved functional entities and assigned resources; applying Object Views to extract that part of the information needed by the involved resources; requesting the involved functional resource to execute the enterprise activity; supplying the results of the execution; providing status information on functional entities; visualizing an execution trace as a log of completed events; allowing authorized personnel to enter events and activities as defined in the enterprise model; modifying the contextual parameters of a simulatable model by the resource at run-time in order to support decision-making, process analysis and exploration of the consequences of alternative scenarios, and interpreting responses from the resources.

2.10.5 Interoperability Management (IM)

Interoperability Management is a prerequisite feature for an EOS. It is a precondition for a successful implementation of the Enterprise Operating System by allowing different content management systems to inter-operate and ensuring that the EOS can ”talk” to both Business Software Applications and the Embedded Software provided for various machine device controllers and sensors.

The federated interoperability approach (Chen, Youssef, and Zacharewicz 2015) seems to be the best solution for EOS development. This approach facilitates ‘on-the-fly’ interoperations between entities with a dynamic method through self-adaptation and accommodation. IM must at least provide the following two functionalities:

- On-line interoperability establishment service: It allows a plug-and-play ability to establish the interoperability whenever an external resource is connected to the EOS. IM can automatically detect possible mismatch and perform necessary mappings.
- Off-line interoperability engineering service: To establish interoperability between a newly connected resource and the EOS, some human interventions are needed to make necessary changes to solve non-interoperability problems.

The federated approach aims at bridging the gap from interoperability concepts to the implementation of interoperable enterprise systems development. It facilitates and coordinates the communication between heterogeneous distributed information systems of the enterprises by allowing quick interoperability establishment, easy-pass, and

dynamic environment update.

It is to note that IM and PM have different roles. PM is concerned with how information is presented (shown) in the display devise of a resource (table, graphic, lines,…) and how to connect EOS to the resource. IM is to ensure that information exchanged between EOS and its external elements can be interpreted/understood correctly without syntax, semantics and technology incompatibilities.

2.11 Conclusions

This chapter presented at first the basic definitions and architectures related to the Computer Operating System "OS" including its related functionalities and components which can highly help in the development an Enterprise Operating System. After that, concepts, objectives and models related to the Distributed Simulations and Enterprise Interoperability are described, analyzed and reviewed highlighting the big role of these concepts for "EOS" by providing real-time execution and monitoring.

Moreover several existing theories, methodologies and approaches related to the development of an Enterprise Operating System are described highlighting that tremendous efforts have been spent to develop enterprise integration and enterprise operating system modules; however, these approaches are still not satisfactory due to the lack in managing and guaranteeing the daily enterprise operations interoperability, monitoring and control.

Furthermore, this chapter presents an outline of the requirements and functionalities for developing an Enterprise Operating System which are inspired from and based on the work specified in a pre-European standard known as prENV 13550 "Advanced Manufacturing Technology - Systems Architecture - Enterprise Model Execution and Integration Services" (AMICE 1993).

Chapter 3. State-of-the-Art

This section provides a state-of-the-art of relevant existing works that have been carried out since the 1980's to support enterprise integration and interoperation. These approaches are analyzed in relation to the set of EOS requirements defined in the previous section to assess their effectiveness with respect to EOS. A comparison study is also provided so that their relevance to EOS can be appropriately determined. Pointing out that the ERP module covers and includes the below works and models, however it is presented as a black box; it could not be modified, updated, customized.

3.1 Overview of relevant works

Enterprise integration is a technical field of enterprise architecture, which focused on the study of topics such as system interconnection, electronic data interchange, product data exchange and distributed computing environments (F. Vernadat 1996). It aims at integrating divers and heterogeneous enterprise applications so that they can work as a 'whole'. Many works have been performed in this field, this section will give an overview on the most important ones.

3.1.1 Common Object Request Broker Architecture (CORBA)

The Common Object Request Broker Architecture "CORBA" is a standard founded in 1989 by the Object Management Group "OMG" and designed for facilitating and enabling the communication of systems deploying on diverse platforms, written in different languages and implemented on different operating systems and computing hardware.

CORBA is designed to link disparate applications together, which means that distributed, heterogeneous application can communicate with each other in a location and language independent manner (McCarty and Cassady-Dorion 1998). The remote client application requests the public interface in the remote server by utilizing the Interface Definition Language (IDL). There is an IDL stub at the client side and an IDL skeleton at the server side. The IDL provides a programming language neutral method for indicating the specifics of an interface. It can likewise be utilized by different systems to create the vital stub code that will facilitate distributed communication.

This communication can only be carried out within the Object Request Broker (ORB), which is achieved by defining a generalized communications protocol – the Inter-ORB

Protocol (IIOP). This protocol standardizes the format of communications that are passing between the distributed CORBA based applications. This protocol likewise permits clients written in any programming language and on any platform to communicate with one another.

CORBA provides flexible data typing for reducing human errors, high level of detail in error conditions, and data to web interface. It uses an object-oriented model and offers benefits such as inheritance, information hiding, reusability and polymorphism. However CORBA is seen more as an enterprise application integration platform rather than an Enterprise Operating System (OMG 2012; McCarty and Cassady-Dorion 1998).

3.1.2 ENV 13550

ENV 13550 is a pre-European standard based on CIMOSA enterprise modeling and enterprise integration concepts, developed in 1990's to support the Computer Integrated Manufacturing "CIM". The main objective is to support system-wide business process monitoring and control. In other words, ENV 13550 defines the requirements to express the capabilities of environments for developing, executing and integrating enterprise models on an open IT based platform. It is defined at a high level of abstraction and does not specify how services are to be implemented. There is neither industrial application nor commercial product developed on the market.

ENV 13550 focused on Enterprise model execution; however, integration of divers' heterogeneous applications is not well supported as CORBA (CEN 1999; Kosanke 1995; AMICE 1993). Interoperability issues were not addressed in ENV 13550.

3.1.3 Enterprise Service Bus (ESB)

The Enterprise Service Bus "ESB" is a software architecture model defined in 2002 as a set of rules and principles to promote and enable interoperability between heterogeneous environments. The concept has been developed in analogy to the bus concept found in computer hardware architecture combined with the high-performance computer operating systems, and used for designing and implementing communication between mutually interacting software applications in a distributed computing domain especially in enterprise application integration and mediation (Flurry and Clark 2011; Chappell 2004).

An enterprise service bus implements a communication system between mutually

interacting software applications, software architecture for distributed computing, and a special variant of the more general client-server model. ESB promotes agility and flexibility with regard to high protocol-level communication between applications.
The primary duties of an ESB are to: monitor and control routing of message exchange between services, resolve contention between communicating service components, and control deployment and versioning of services (Sethi, Raynaud, and Faure-Vincent 1995; Vollmer 2011).

3.1.4 Workflow Management (WfM)

In parallel to various initiatives for developing integration architectures and platforms, enterprise model execution also gained wide attention in the mid 90's with workflow management (WfM).
The Workflow Management Coalition "WfMC" is a non-profit organization formed to define standards for the interoperability of Workflow Management Systems "WfMS". It was founded in May 1993 with original members including IBM, Hewlett-Packard, Fujitsu, and ICL. It focuses principally around process definition file interchange by using the standard "XPDL" which present a process design format for storing the visual diagram and process syntax of business process models as well as extended product attributes. "WfM" is generally used in order to coordinate and sequence business processes, such as loan approval, insurance reimbursement and other office procedures.
Like ENV 13550, the aim of WfM is process model execution, which is one of the main functions of an EOS (AMICE 1993; WfMC 1999).

3.1.5 Enterprise Application Integration (EAI)

Going one step further than CORBA, Enterprise Application Integration (EAI) platforms are integration frameworks made of a collection of technologies and services to provide integration of systems and applications that reside on different operating systems and use different database solutions across an enterprise (e.g. SCM applications, ERP systems, CRM applications, payroll…) (AIIM 2000; Gable 2002; Linthicum 1999). EAI architecture is typically based on four components including a centralized broker (or hub) that handles message routing, security services, communication services and connectors. The latter are used to interface each application to the EAI hub so that it can communicate with the other applications. The EAI approach focuses on enterprise application integration by means of normalized

message exchange (preferably XML). It was not dedicated to enterprise operations monitoring and control as foreseen for EOS. In addition, EAI products are based on tight integration approaches, leading to so-called monolithic architectures, rather than on loosely coupled interoperability, which is more suited for EOS because of its agile nature (i.e. components can be easily added, removed or modified in the architecture).

3.1.6 Open Software Foundation/Distributed Computing Environment (OSF/DCE)

The Open Software Foundation "OSF" is a non-profit organization founded in 1988 by several sponsoring members such as Apollo Computer, IBM and Digital Equipment Corporation and has been incorporated into The Open Group to foster, identify and develop technologies that could serve as industry and international standards. OSF developed a widely-implemented platform for distributed computing called the Distributed Computing Environment "DCE".

DCE is a middleware oriented toward heterogeneous systems, used to allow different operating systems and different hardware platforms to distribute processing and data across the enterprise by providing a coherent environment for developing and running applications. It supports the sharing of information and provides services to mask the complexity of multivendor network environments (Fagg, London, and Dongarra 1996; Moon and Lee 2006; Turner 2012).

3.1.7 Open Distributed Processing (ODP)

An important initiative under the frame of ISO/IEC was the development of a Reference Model for Open Distributed Processing (RM-ODP). This international standard was published in 1990 to describe and build widely distributed systems and applications in a multi-vendor environment (ISO/IEC 2009). The ODP framework integrates aspects related to the distribution, interoperation and portability of software systems, so the operating systems, programming languages, databases and management systems are transparent to the user, and it manages also complexity and maintains consistency of information (Zurawski 2004).

ODP and DCE are similar approaches aiming at supporting stem-wide distributed communication and sharing of information. This is a core functionality to be provided by EOS. However, both RM-ODP and DCE do not support enterprise model execution and they are dedicated to IT systems and not aimed at enterprise operations monitoring

and control.

3.2 Relevance of existing approaches to EOS

This section presents a comparison study of existing approaches reviewed in the previous section and evaluates their relevance with respect to EOS. The criteria used for the evaluation are based on the five categories of requirements and functionalities of EOS defined in section 2.10.

The following notations are used: '+++' means that there is a strong correlation and that the approach highly fulfils the EOS criteria, '+' denotes that there is weak correlation and '++' is in between, '-' means that the approach does not meet the criteria. Table 2 gives an evaluation on the degree of relevant of the existing approaches to EOS.

	Enterprise Resource Management	Enterprise Process Management	Enterprise Information Management	Presentation management	Interoperability Management
CORBA	+	+	+ + +	+ + +	+
ENV 13550	+++	+ ++	+ + +	+ + +	-
ESB	+ +	+	+ +	-	-
WfM	+ +	+ +	+ + +	+ + +	+ +
EAI	++	++	+++	+	+
OSF/DCE	-	+	+ + +	+ + +	+ +
ODP	+	-	+ + +	+ + +	+ +

Table 2. Comparison of existing approaches against EOS requirements

As shown, none of the existing approaches can meet the EOS requirements in a satisfactory way. On the other side, we noticed that ENV 13550 and WfM are the most relevant ones. ENV 13350 is designed as an integration platform focusing on enterprise model execution while WfM is more oriented to monitor and control workflow. The two approaches are complementary. The following sections will give more details on the evaluation.

3.2.1 Enterprise Resource Management

Concerning Enterprise Resource Management, most of the existing approaches only focus on IT resource management (monitoring and control). Only ENV 13550 and to a lesser extent WfM deal with non-IT enterprise resources such as 'Human' and 'Machine'. Table 3 shows the evaluation for the various approaches regarding to Enterprise Resource Management.

	CORBA	ENV 13550	ESB	WfM	EAI	OSF/DCE	ODP
Monitor status of resources	Yes	yes	Yes	Yes	Yes	Partially (keeping track for the status of resources is difficult in the distributed environment)	Partially (only by implementing the TBRMS)
Command resources	No	no	No	No	Yes	No	Yes
Search available resources on the request	Yes	no	Yes	Yes	Yes	Partially (keeping track for what resources are available is difficult in the distributed environment)	Partially (difficult without the Trader component)
Match the required capabilities to resource	No	no	Yes	Yes	Yes	Yes	No
Allocate/de-allocate resources	Yes	yes	Yes	No	Yes	No	Partially (in case of the presence of exporters)
Report status of resources	Yes	yes	Yes	Yes	Yes	No	Partially (based on the Trader component of the ODP)

Table 3. Evaluation against Enterprise Resource Management

CORBA provides a distributed resource management service that monitors slightly the information about resource usage and availability and ensures that adequate resources are available for use when and where they are required along to the end-to-end path of a computation through the distributed system (Chang 2005; Zaraté 2013).

ENV 13550 also allows monitoring status of resources (e.g. available, occupied…). It commands connected resources by sending execution orders through process management service. When the defined resource is available, it can allocate the resource to the operation according to the model and de-allocate the resource when the operation is finished. When a resource is freed after an operation, a report is sent to change the resource status from ‘occupied’ to ‘available’.

An ESB contains a module called “Request Handler” to connect users to the ESB and

to different parts of the framework. It forwards the data source requests from the user to the service manager and uses the selection and restriction processes to find the best data source for each resource (Fagg, London, and Dongarra 1996; Rademakers and Dirksen 2008).

WfM includes a "role coordinator" component which is responsible for managing the resources that can perform a particular role and their related state information (Henning 2006). When an activity of a case needs to be performed, a new activity manager is created at the preferential host of the resource assigned by the role coordinator to that activity. The "role coordinator" has access to the history server which stores information about completed cases and control databases. This information allows the "role coordinator" to answer all kind of queries (i.e. resource with most experience, last resource generating an event…) (WfMC 1999).

Concerning EAI, it allows resource access to products and information and enables them totally to control business processes involved in interactions between different business applications by matching the needs of each resource to the process available and compatible. It provides quick response time and better coordination between enterprise resources as well as high availability (AIIM 2000).

While in DCE the resource management is very limited, there is no single entity that has full control of the resources including the availability, the status, the location, the information and the properties of each resource. The connection and communication between different resources is implemented through the Remote Procedure Call "RPC". The RPC runtime library is actively involved in sending and receiving the remote procedure calls, finding the necessary server services and communicating between different resources and the server.

As for ODP, it is difficult where the resources are scattered throughout the distributed computing environment, they are not confined to a single location and no single entity has full control of all resources. However, ODP includes the "Trader" component which manages partially the distributed system, and monitors the resources' specifications such as location, naming of the software entities, naming of users and security policies for access control (Moon and Lee 2006).

3.2.2 Enterprise Process Management

Regarding Enterprise Process Management (see table 4), most of the existing approaches include Process Management. However, ENV 13550 and WfM provide the

most interesting concepts and principles to define EOS, especially in the sense that EOS is focusing on the business process execution defined by the decisions makers.

	CORBA	ENV 13550	ESB	WfM	EAI	OSF/DCE	ODP
Execute process model	Yes	yes	Yes	Yes	Yes	Yes	Yes
Execute commercial process modeling software	Yes	no	Partially	No	Yes	Yes	No
Monitor process execution	Yes	partially	Yes	Yes	Yes	Yes	Yes
Send 'request' to Resource management	No	yes	Yes	Yes	Yes	Yes	Partially (only in case of the presence of TBRMS)
Trigger process 'start'	Yes	yes	No	No	Yes	Yes	No
Interpret 'resource status' from Resource management	Yes	yes	Yes	Yes	Yes	Partially (difficult in case of large number of resources in a distributed environment)	Partially (only by impleme nting the TBRMS)

Table 4. Evaluation against Enterprise Process Management

CORBA includes a central building block called Object Request Broker (ORB), which delivers the requested objects and returns the results to the resources even if the process is executed on a different platform. It can determine if a process failed for reasons in case of not receiving an exception (Henning 2006).

ENV 13550 helps making the enterprise's workflow and activities more effective, more efficient and more organized by handling and managing the enterprise business process. It aims at dynamically controlling the process model run-time execution by sending commands triggering the start of processes, recording the ending statuses, sending

required capabilities to Resource Management to find/allocate the available qualified resources, receiving a signal in order to release resource, and running/executing process models from commercial and non-commercial modeling tools.

An ESB includes an engine controlled by the process description that coordinates the collaboration of the services connected to the bus in order to execute the related processes (Sethi, Raynaud, and Faure-Vincent 1995); it contains process configuration elements such as endpoints, fault handlers, process mapping and tracking in order to define how a process flows between services. ESB process can be used by more than one resource at the same time, and multiple ESB processes can use a single service.

WfM engines have a "process coordinator" component that manages the description of a particular process and creates a new case coordinator for new orders instantiating and transferring the plan to that object (Sethi, Raynaud, and Faure-Vincent 1995). This process is configured with its specific data in order to be executed using the appropriate applications, and then the process coordinator sends the appropriate information to the appropriate resource's task list, notifying the role coordinator that the current activity is terminated and informing who is the selected resource to perform the next activity. This component keeps track of all case coordinators that are still active and propagate any process changes to all cases. It is centralized in a way that all processes can be located on the same resource or they can be disseminated on different resources in order to achieve the maximum distribution.

In EAI products, a "Process Layer" is used for automated processes and to ensure the proactive control of the entire process from instantiating a predefined workflow type all the way to its completion (Gable 2002). Moreover, it provides a business process oriented solution that is configurable across applications and that supports automated deployment by integrating and distributing business processes. It provides the necessary isolation and abstraction between the internal processes and the processes across organizations.

Concerning DCE, it has the "Access Control" model, which provides an object level protection and auditing of each process (Chen, Doumeingts, and Vernadat 2008). The user library allows processes to be added or deleted by sending messages to the related resource and permits to a process to start or terminate by using the operating system services (Fagg, London, and Dongarra 1996). It has three scheduling algorithms used to execute processes (FIFO, Round Robin and Default) and it uses the "Mutex" model in order to prevent multiple processes from accessing the same resource at the same time.

ODP also includes a basic concept defining the rules that govern the collective behavior of a set of processes, the period in which a process exists and its terminating behavior (Chang 2005). The life cycle of this concept is defined in terms of its establishing behavior, the period in which a process exists, and its terminating behavior.

3.2.3 Enterprise Information Management

Managing system-wide information is essential in enterprise integration and all existing approaches can support the Enterprise Information Management with different levels as shown in table 5. Nevertheless, due to the heterogeneous data sources in an enterprise, interoperable solutions are more suitable and fundamental to support this function.

	CORBA	ENV 13550	ESB	WfM	EAI	OSF/ DCE	ODP
Provide system-wide info. exchange facility	Yes	yes	Yes	Yes	No	Yes	Yes
Ensure data consistency and integrity	Yes	yes	Yes	Yes	Yes	Yes	Yes
Provide transparent access to data sources	Yes	yes	Yes	Yes	Partially (EAI does not ensure the access to web data sources)	Yes	Yes
Ensure data confidentiality and security	Yes	yes	Yes	Yes	Partially (EAI includes the centralized broker for security, however it faces issues with several applications)	Yes	Yes
Provide an appropriate run-time data storage	Yes	yes	Partially (only for the state of information and not for the deployed services)	Yes	Yes	Yes	Yes

Table 5. Evaluation against Enterprise Information Management

CORBA includes a local information collection component, which records all

information including the progress of activities, status of processes and data exchanges. It manages the information in the system repository, with all information stored about end-to-end processes and resources (Wainer, Filho, and Madeira 2000). The Vault and "Security Context" objects take part in the establishment of secure associations, which guarantee confidentiality and integrity of messages exchanged between resources.

ENV 13550 provides access to system-wide information and data interchange facilities from all documents between enterprise resources connected to the Enterprise Operating System. It aims at ensuring data confidentiality and security to prevent any unauthorized access, requesting to provide required services for establishing interoperability between non-interoperable data sources and supporting flexible change of information including data definition and configuration control, and providing an appropriate data storage facility needed for running EOS.

ESB solutions encapsulate the information offered by their component applications in a meaningful way by using an "Enterprise Message Model" which routes the information transmitted and received by the ESB to the appropriate application after transforming the message into a format that the application can interpret. They also monitor and control routing of information exchange between services (Rasta 2013; Chappell 2004).

Concerning WfM, workflow engines include the "case coordinator" component that keeps track of the case status as it moves along, collects cache information left by activity managers in the hosts of the system, stores the collected data in the history server, centralizes all information concerned with each case by managing the activities of that case, detecting failures, coordinating its recovery procedures, answering to queries about the case and notifying the process coordinator when a case is closed. This component creates a synchronization activity for each "and-join" specified in the process definition by adding their addresses and names to its plan (WfMC 1999).

EAI based commercial tools (such as Tibco for example) can also ensure the proper routing and transformation of messages between applications in multiple systems by selecting, targeting, converting and mapping data to be used. This layer addresses the mismatch of data either at the lower level of data type representation or at the higher level of mismatched data structures and it also transforms the external events in messages and information to ensure the integration between different applications (Linthicum 1999).

DCE provides an end-to-end view of business information by providing common definitions of enterprise data to be integrated in the internal system architecture. It

contains a graphical user interface based management environment that allows a secure remote management of any resource and process in an enterprise network and helps resources to monitor the aspects of the system and its applications.

In the case of ODP systems, they interconnect the distribution and processing of information and support the exchange of data across organizations in the enterprise. They guarantee a backup of information to recover in case of failure and allow resources to perceive immediate access to data. When resources interact with the system, their interactions are managed by the ODP in order to prevent any violation to the integrity of information.

3.2.4 Presentation Management

Presentation Management is not a core function of EOS but necessary to interact with all resources and business users connected to EOS. ENV 13550 presents the most interesting concepts, because it allows dialogue with all types of enterprise resources (human, machine and IT) (see Table 6).

	CORBA	ENV 13550	ESB	WfM	EAI	OSF/DCE	ODP
Provide 'human' dialogue (and display) service	Yes	yes	Partially (display service and not human dialogue service)	Yes	Yes	Yes	Yes
Provide 'machine' communication interface	Yes	yes	No	Yes	No	Yes	Yes
Provide 'application' communication interface	Yes	yes	Yes	Yes	Partially (EAI interface may be unstable on the level of applications communication)	Yes	Yes

Table 6. Evaluation against Presentation Management

CORBA uses the middleware services locating in the middle of the platform and application layers which are responsible for Presentation Management by providing a set of application programming interfaces in order to allow resources to operate using any type of platforms. Through the Interface Description Language “IDL”, the middleware describes an interface in a language-independent way that manages the communication between different platforms.

ENV 13550 also has Presentation Management providing a set of interfaces and

commands between internal and external worlds of the platform. It organizes the information flow between the platform and the Enterprise Resources and interacts with the Human, Machine and Application dialogue services.

ESB has no Presentation Management but an Eclipse-based graphical development environment for designing, testing and running ESB flows (Vallejo, Romero, and Molina 2012). It consists of editors for development and a management console for providing run time management facility of deployment to the ESB repository and clusters. It helps for integration to connect applications, data and devices with integration connectors (Wyszkowski 2011).

As for WfM, it contains a "worklist handler" that manages the interaction between the user and the work list maintained by a workflow engine. It enables work items and notifications of work status to be passed between the workflow management system and the resources. The user interface is implemented as a task list similar to an e-mail reader which notifies the resource of new activities that he supposed to perform. This allows the role coordinator to assign the appropriate resource according to the current specified policies. This user interface allows the resource to customize its preferred external applications, the policies to customize its preferred external applications, the policies for sorting the coming activities and the preferential host.

EAI platforms only have a front-end to cluster applications in order to provide a single consistent access interface, to ensure the consistency of information and to simplify and automate business processes (CEITON 2014). However, the EAI user interface is unable to handle more than few screen interfaces at a given time, which may prove unstable if the mechanisms are not set up and developed correctly (Jin 2009).

DCE contains a graphical user interface based management environment that allows a secure remote management of any resource and process in an enterprise network and helps resources to monitor all aspects of the system and its applications. The management console maintains a centralized database of service and object rules.

ODP possesses an interface that interacts as an abstraction of the behavior of the workflow that consists of a subset of the resources with a set of processes to define when the identified interactions can occur.

3.2.5 Interoperability Management

Last but not least, interoperability is a key function to allow diverse heterogeneous enterprise resources connected to the EOS to perform enterprise

operations. Interoperability in existing approaches is not developed to a mature and satisfactory level. A set of interoperability utility services needs to be implemented to support EOS (see Table 7).

	CORBA	ENV 13550	ESB	WfM	EAI	OSF/DCE	ODP
Ensure plug & play of commercial applications	Partially (problems only occur when using peer-to-peer communication)	No	No	No	Partially (the new generations of EAI tools are moving much closer to the plug and play approach)	Yes	Yes
Ensure data interoperability of divers data sources	Partially (lack of interoperability is noticed in presence of the deficiency of automation)	No	Yes	Yes	No	Partially (difficult with large number of resources and applications)	Yes
Provide syntax interoperability (format, table…)	No	No	No	Yes	No	Yes	Yes
Support semantic interoperability (use of ontology..)	Yes	No	Yes	Yes	Yes	Yes	Partially (ontologies for policy sets relevant for business processes is needed)

Table 7: Evaluation against Interoperability Management

CORBA includes the ORB Interoperability Architecture, which is defined as a tool for the enterprise system environment in which a single interface can be developed for each resource having multiple processes that can be used (Omicini, Petta, and Pitt 2013). It provides a conceptual framework for defining the elements of interoperability and identifying its compliance points. However, lack of interoperability is occasionally noticed because of the deficiency of automation and the fact that earlier CORBA specification did not mandate any particular data formats or protocols for ORB communications.

ENV 13550 is not designed for supporting interoperability between enterprise resources.

ESB establishes an Interoperability Management System to connect platforms to each

other and organize the share of resources (Konstantas et al. 2006). It is placed between different systems and act as a translator between them in order to implement transportation protocol conversion, message routing and data mapping as well as providing information security and monitoring mechanisms.

WfM possesses the “synchronization activity” feature that organizes the workflow between resources by managing the “and-joins” and “or-joins” notifications. The synchronization activity waits for all notifications (and-join) or the first notification (or-join) from its preceding activities before starting the following output and activities. It also defines the “Gateway Activity” feature, which is responsible of the bidirectional workflow data, control and process definition conversions between different workflow management systems (Saha, Mukherjee, and Bandyopadhyay 2011).

EAI tools provide utilities and infrastructure services for integration of enterprise applications that enables information sharing and business processes. However, data sources cannot normally operate in heterogeneous environments and they are facing problems of communication and execution related to their respective frameworks (May 2001). EAI systems do not support service interface heterogeneity and several standards are used in order to achieve limited kind of semantic interoperability between enterprises.

DCE facilitates the development and deployment of interoperable applications in the distributed heterogeneous networked environment and it also helps for the integration of legacy applications with new distributed platforms (Xu 2014).

ODP systems are based on open solutions to provide general connections among software entities and ensure the cooperation and sharing of information. It is based on well-developed enterprise modeling languages and a distributed system architecture, which jointly position ODP as a perfect framework for modeling large cross-organizational and cross-jurisdictional systems that communicate over the Internet (Weichhart et al. 2016). However, careful architecting of distributed processing systems is needed to mechanize the communication and coordination of distributed information relevant to the enterprise (Zurawski 2004).

3.3 Conclusion

This section presents a summary of the relevance and insufficiencies of the existing approaches (see Figure 11) based on the EOS requirements and functionalities and according to the evaluation and estimation described in Table 2.

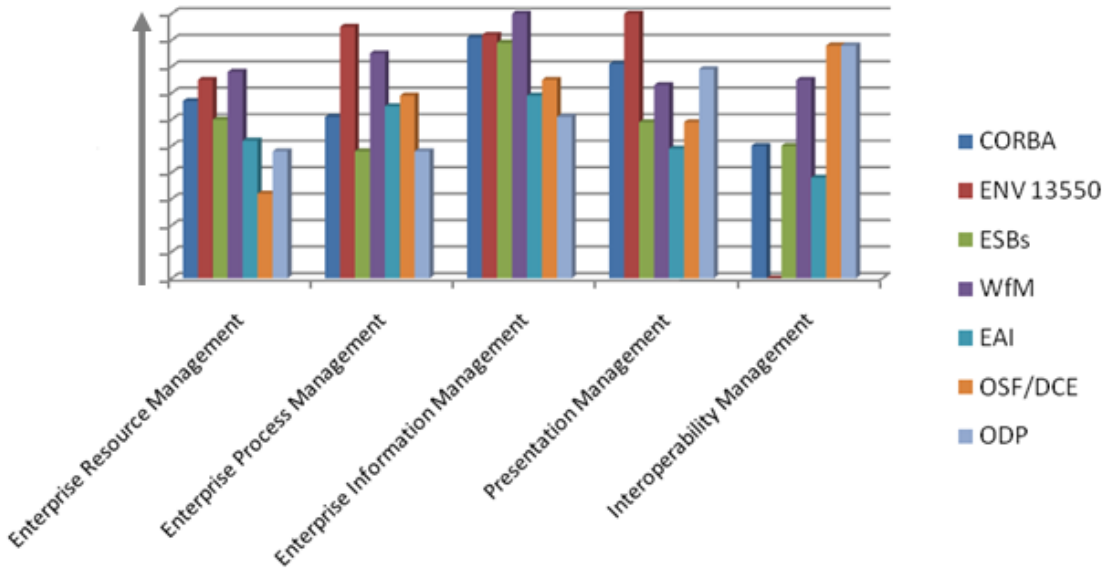

Figure 11: Relevance and insufficiencies of the existing approaches.

Considering Enterprise Resource Management, most of the existing approaches only focus on IT resource management (monitoring and control). However, ENV 13550 and WfM also deals with other non IT enterprise resources such as 'Human' and 'Machine' and allow discovering and matching required capability to existing available resources.

Referring to Enterprise Process Management, most of the existing approaches deal with Process Management. However, ENV provides the most interesting concepts and principles to manage EOS processes defined by the business users and orchestrated to ensure internal working of platforms.

Regarding Enterprise Information Management, all existing approaches support this function to some extents but due to the heterogeneous data sources, data interoperability is not yet well implemented.

Concerning the Presentation Management, ENV 13550 presents the most interesting concepts, as it allows dialoging with all types of enterprise resources (human, machine and IT).

For Interoperability Management, existing approaches focus on Enterprise Integration rather than loosely coupled interoperability.

In conclusion, the existing works and models are all relevant for the design and development of an EOS but they do not support and cover all the requirements and functionalities needed by EOS. Each of them has its particular objectives with specific focuses. CORBA and EAI are seen more as enterprise application integration platforms rather than Enterprise Operating Systems. Compared to CORBA, ENV 13550 moved a

step towards an EOS. Unfortunately, the work of ENV 13550 is discontinued. It must be noted that service orientation has gained interest in the middle of the 90's and has been used as a basic principle to develop enterprise integration and interoperability platforms. The ESB developed for this purpose cannot be directly used as an EOS but they provide interesting concepts and principle to develop and build the EOS. Last but not least, and at a technical level, some other approaches are also interesting to take into account to develop EOS technical architectures. We can mention as an example HLA (High Level Architecture) that allows building federated interoperability capability in the EOS (Knight et al. 2002).

Chapter 4. Contribution: EOS architectures

4.1 Introduction

This section presents the conceptual, technical and implementation architectures of an Enterprise Operating System that are designed to meet the requirements and functionalities identified in chapter 2 - section 2. The conceptual architecture defines what core functions are needed in EOS, while the technical architecture specifies how these functions work together to accomplish the EOS mission and the implementation architecture describes what technology and concepts are being used to implement the conceptual and technical architectures.

4.2 Position of EOS based on interoperability

Based on the Enterprise Interoperability Framework detailed and described in figure 7, this doctorate research can be positioned as mentioned in figure 12 and used as an interoperability interface connected to the Enterprise Operating System in order to set up interoperability rapidly among existing enterprise information systems.

EOS adopts unified and federated approaches by aiming at shifting from the integration to the full federation concept by facilitating 'on-the-fly' interoperations between entities and supporting the communication and transactions between heterogeneous and networked enterprises based on shared business references; it also strives at establishing interoperability through the conceptual and organizational barriers in the four different levels of concerns (data, service, process and business) by adjusting business rules to control operations and tasks, facilitating the alignment and real-time communication with enterprise activities, providing a mechanism to control the authority of accessing the services, ensuring data bases connection, and providing a data distribution service for implementing on-the-fly information mapping.

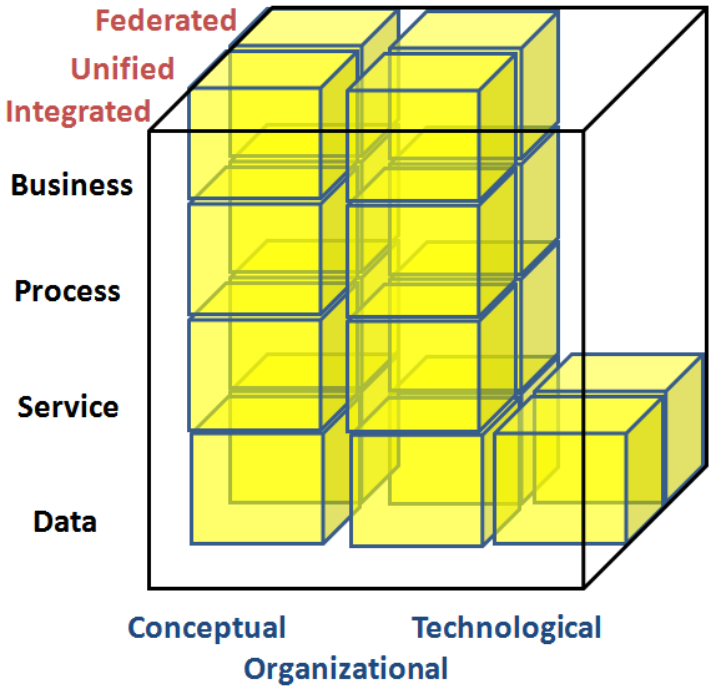

Figure 12: EOS positioned in the Interoperability framework

The intersection of an interoperability barrier and an interoperability concern is the set of interoperability problems having the same barrier and concern (see table 8). In order to constitute the solution for the interoperability problem, the interoperability approaches are imperative (Salinesi and Pastor 2011).

		Conceptual	Organizational	Technological
Federated	**Business**	Use standards to facilitate the alignment with other models based on the on-the-fly mapping concept	Adjust business rules to control operations and tasks	
	Process	Facilitate the alignment and real-time communication with processes and enterprise activities	Define and adjust procedures of work	

	Service	Provide a platform-independent, technology-independent, and language-independent service for the participants	Provide a mechanism to control the authority of accessing the services	
	Data	Address the on-the-fly mapping issue of information with different format (syntactic) and meaning (semantic)	Provide a mechanism to manage the ownership of the data and the authority of obtaining information	Provide a data distribution service for implementing on-the-fly information mapping
Unified	**Business**	Use standards to facilitate the alignment with other models	Adjust business rules to control operations and tasks	
	Process	Use standards to facilitate the alignment with other models	Define and adjust procedures of work	
	Service	Use standards to facilitate the alignment with other models	Put in place and adjust guidelines for service exchanges	
	Data	Use standards to facilitate the alignment with other models	Adjust rules and methods for data interoperability management	Ensure data bases connections and remote access

Table 8: Interoperability issues relevant to EOS

The implementation of EOS will use the existing interoperability solutions to support the development of federated and unified approaches of enterprise interoperability. In particular, the HLA architecture will be used to rapidly develop HLA based interface for achieving federated enterprise interoperability.

4.3 Conceptual architecture

A conceptual architecture is a representation of the main functions of a system from the point of view of its use. It is independent of how it technically works and is implemented using a specific technology. Figure 13 describes the proposed EOS Conceptual Architecture. Unlike ERP, this EOS will be used as a system-wide platform that allows the decisions makers to connect to and communicate through the enterprise systems (hardware, software, network, machines, human operators…) in an effective and efficient way (Youssef, Zacharewicz, Chen, and Zhiying 2017; Chen, Youssef, and Zacharewicz 2015).

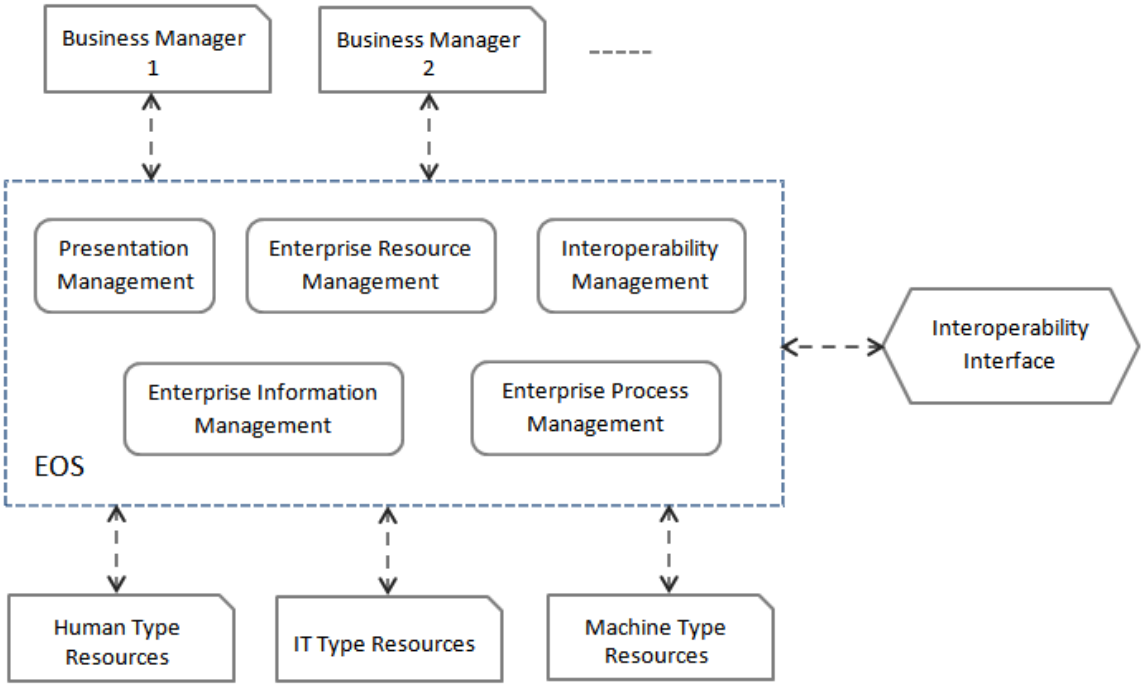

Figure 13: EOS Conceptual Architecture

As shown in figure 13, business users and the three types of resources are outside the EOS. They are connected to the EOS to send and receive commands (or orders) and information (e.g. data, feedbacks, statuses...). To perform enterprise operations (i.e. to design, manufacture and sell required products to customers) in a coordinated way, the resources must be commanded and controlled by business managers via the EOS:

- *Human type resources* are human actors, operations of whom can be monitored and controlled by the EOS. They are commercial and purchasing agents, product designers, business developers, production managers, shop floor operators, etc.
- *IT type resources* are computer and communications systems including data processing, storage devices and enterprise applications such as, for examples, MRPII planning software, shop floor scheduling software, CAD system, sale forecasting software, CRM software, inventory management software, etc.
- *Machine type resources* are production or transport machines, technical devices, pieces of equipment and tools containing one or more parts that use energy to transform raw material to products such as, for examples, automated/manual transfer lines, conventional and NC machines, robots, etc.

Besides the three basic types of resources, two other components need to be connected to an EOS. They are Business Managers and Interoperability interface:

- *Business managers* are not monitored and controlled by the EOS; they are the decision-makers who define what and how enterprise operations will be done, manage the activities and send commands to enterprise resources via the EOS.
- *Interoperability Interface* allows interoperability engineers to deal with some particular interoperability problems on-line or off-line. It also enables interoperability between EOS internal and external components connected to the EOS.

As shown in figure 13, an EOS is made of five modules providing required services to ensure the functioning of the EOS:

- *Enterprise resource management* dynamically monitors the connected resources system-wide (available, occupied, out-of-order…), matches the required capabilities to the capabilities of existing available resources; and ensures that the right resources are allocated to the right place at the right time.
- *Enterprise process management* executes business processes defined by business managers, sends commands triggering the start of processes, records ending statuses of processes and monitors the activity and state of individual enterprise processes as well as some key performance indicators.
- *Enterprise information management* provides a centralized management of reference data used, automates and arranges master data management, provides

transparent access to data sources and an appropriate data storage facility needed for running the EOS and guarantees information and data confidentiality and security to protect from non-authorized access.

- *Presentation management* is a set of services with appropriate interfaces that allow business users and other enterprise resources to connect to EOS and receive/send information.
- *Interoperability management* is a set of services that provide necessary mapping between heterogeneous resources to make them interoperable through EOS.

4.4 Technical architecture

A technical architecture describes how components or services of a system interact to fulfill the required functions. The technical architecture is still independent of a specific technology for implementation. The figure 14 illustrates and presents the EOS layers' workflow where the enterprise operations are executed and generated through the EOS internal components (Youssef, Zacharewicz, and Chen 2016; Youssef et al. 2016).

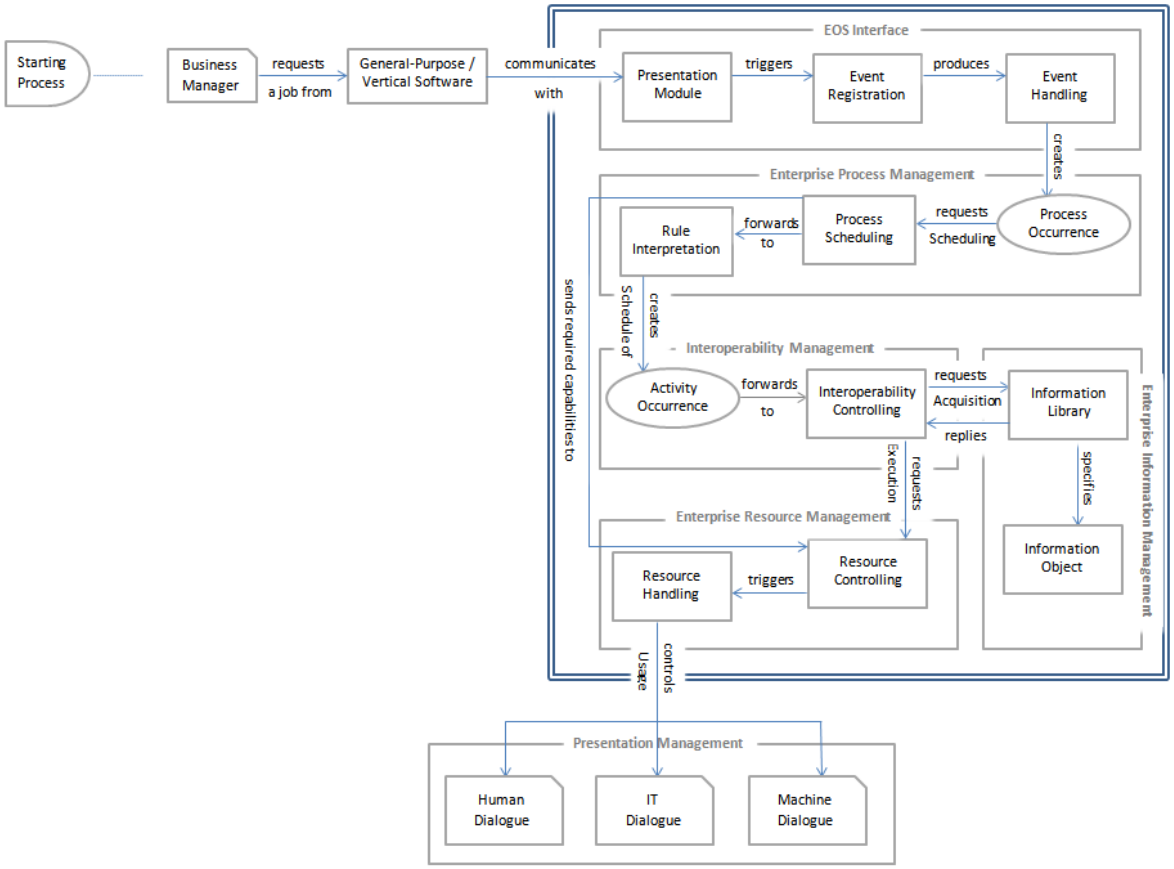

Figure 14: EOS technical architecture

At first, each decision maker accesses the General-Purpose and Vertical software interfaces to request a new job from the day-to-day list of activities and operations. The software sends related commands and data to communicate with the EOS front-end interface named Presentation Module in order to execute the requested job.

The Presentation Module then interprets the run-time entities, creates the events which will be registered with their associated information by the Event Registration Component using the Run-Time Repository Service and produces event occurrences for the Event Handling component.

Next, the Event Handling manages the event occurrence priorities, queues and traceability, provides the order identifier and creates the Process Occurrence. The Process Occurrence requests to scheduling from the Process Scheduling component which interprets the process behavior, information and resource requirements. This sub-service checks the authorization to execute the process, retrieves process descriptions from the Run-Time Repository, invokes the Enterprise Resource Management to allocate the required resource capabilities and forwards the details to the Rule Interpretation component.

Later, the Rule Interpretation component provides functionality to retrieve the sequencing and conditional rules associated with the identified Enterprise Process, maintain a state record of all enterprise activities and respond to detected events in order to initialize or terminate the activity.

Subsequently, the Activity Occurrence schedule created by the Rule Interpretation is forwarded to the Interoperability Component which is responsible of requesting from the Enterprise Resource Management to assign resources allocated by the Process Scheduling Component, invoking the Enterprise Information Management to acquire the object states and specify the information object required, requesting from the Enterprise Resource Management to: assign resources allocated by the Process Scheduling component, release the involved resources when terminating an activity and notify the termination of the current activity to the Rule Interpretation component.

The Resource Controlling component checks the availability of the resources and pre-assigns them when available, responds to the Interoperability Controlling requests in order to assign agents and responds to the Process Scheduling requests to allocate and de-allocate resources. The Resource Handling component ensures the resource optimization and the resource mapping by selecting the appropriate resources after matching the capabilities required and by taking into consideration the time,

performance and priority.

The Presentation Management services are controlled by the Resource Handling Component for handling Human, Machine and IT Dialogues:

- *The Human Dialogue* provides functionality for presenting in appropriate format the current status and the past history of events, allowing authorized persons to intervene manually in order to modify the contextual parameters at run-time.
- *The Machine Dialogue* supports the necessary features in order to provide access to the various functional capabilities of the machine. It provides the functionality required for receiving and interpreting responses from the machine.
- *The IT Dialogue* provides functionality for interrogating application program interfaces to determine its capabilities, providing support for the integration of the functional entities implemented by existing IT application programs.

4.5 Implementation architecture

Implementation architecture describes how to physically realize a system. It defines the technology and concepts used to implement the technical architecture. In other words, implementation architecture identifies the needed technological components and their interactions in order to build a system.

4.5.1 Implementation approach

Broadly speaking, for a given technical architecture, various different technologies can be used for implementation. In this proposal, the HLA (High Level Architecture) supported by the US Department of Defense is chosen (IEEE 1516 2010; Portico 2013). The main reasons for this choice are 1) its capability to easily enable federated interoperability between heterogeneous applications and 2) its ability to deal with large distributed (simulation) environments. Besides, HLA is also in line with service orientation based implementation that is one of the main characteristics of EOS principles. Indeed, an EOS is mainly an IT infrastructure made of many services. HLA provides one possible implementation support particularly convenient to EOS. Nevertheless, the implementation of EOS is not restricted to HLA.

As shown in figure 15, all EOS federates are connected together through a HLA Federation provided by IEEE in order to publish and subscribe events between federates. This type of structure is used to facilitate inter-federate communications and

to support efficient information exchange when participating in a distributed federation execution (Youssef, Zacharewicz, and Chen 2016).

The HLA technology allows the EOS federates to communicate data and to synchronize actions among one another based on the federated interoperability features by defining how federates can connect to the RTI, create, join and manage federations, save and restore federation states and define a system to synchronize federates to the same time (Portico 2013).

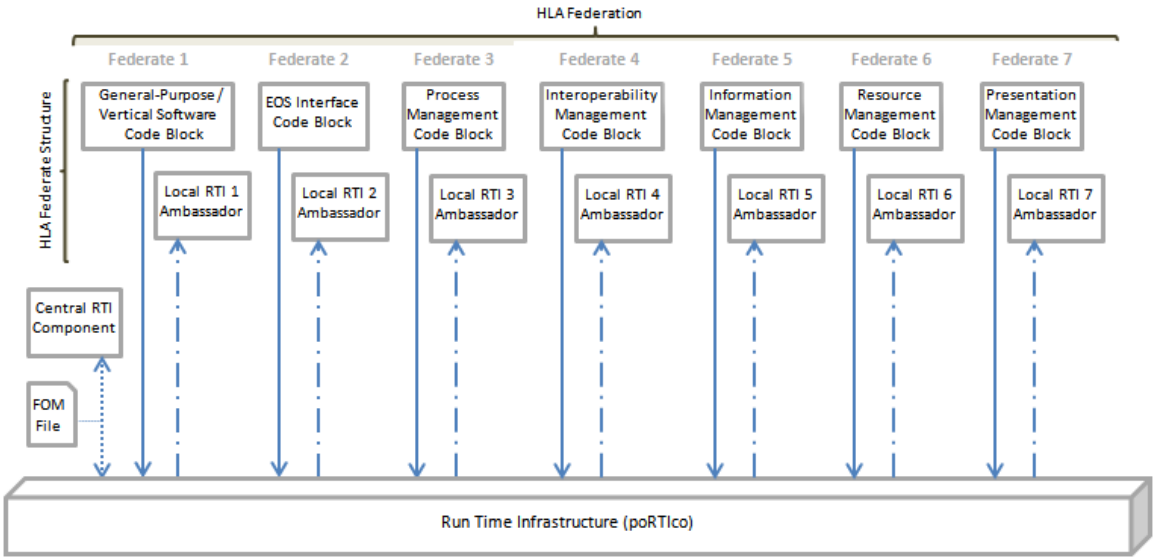

Figure 15: EOS implementation architecture

poRTIco RTI component is the fundamental component presented as an open source, cross-platform used to implement the High Level Architecture to coordinate and interoperate federates' operations and exchange data. This middleware contains a Central RTI component "CRC" connected to the Local RTI component "LRC" of each federate in order to convert requests into messages to be transmitted between federates. It supports HLA simulation development to greatly expand the use of distributed simulation between all components and federates of the EOS.

The Central RTI Component manages the federation by communicating with the LRC of each federate to update, reflect, send and receive data between federates.

Each HLA federate ensures the execution of two parts: (1) the federate code block, which is the source code of the federate developed, implemented and connected to the Local RTI Component using the .Vb and .Jar languages from the C# and the Java libraries "LibRTI" to form a complete federate, and (2) the local RTI component code "LRC" that provides the services for the federate through communication with the RTI executive component and the other federates (poRTIco 2009).

The Federation Object Model “FOM” file describes the set of object classes, attributes and interactions that are shared across the federation. “FOM” contains a copy of the HLA standard Management Object Model, which is a collection of classes and interactions. This file is documented using the HLA OMT format.

Broadly speaking, 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.

It must be noted that this Implementation Architecture is fully implemented using heterogeneous development languages (Java, Vb.net, SQL…) based on the Interoperability and Uniformity principles to provide a set of domain-independent APIs used to access capabilities and features, and to exchange data between federates using the XML format.

4.5.2 HLA FOM Generation

HLA FOM is playing the primary role for supporting the interoperation of distributed simulation by: 1) defining the shared vocabulary of a federation, which allows federates to work with one another in a defined manner, and 2) helping overcome platform differences by providing the right mechanism in order to ensure the reconstitution of received information into its intended format.

HLA FOM systematizes the global scenario of the interoperation, e.g. definition of primary entities and basic interactions, and then specify this scenario into HLA and JAVA related model, e.g. HLA FOM and correspondent JAVA object bean (Zhiying, Chen, and Zacharewicz 2012).

As shown in figures 16, 17 and 18, the MoDisco software tool has been selected in our case study in order to generate at first the KDM model, after that a readable UML file in XML format, and finally generating the HLA FOM file. This tool is an open source plug-in of the Eclipse GMT (Generative Modeling Technologies) component for model-driven reverse engineering used in order to allow practical extractions of models from legacy systems.

MoDisco (for Model Discovery) objective is to allow practical extractions of models from legacy systems. Due to the generally unique nature and heterogeneity of legacy systems, there are diverse approaches to extract models from such systems. MoDisco proposes a nonexclusive and extensible model-driven approach to model discovery. A basic framework and an arrangement of guidelines are provided to the Eclipse

contributors to bring their own solutions to discover models in various kinds of legacy. It is accessible on the Eclipse Website http://www.eclipse.org/MoDisco/downloads/ with the most recent version 0.11.0 released on June 26th, 2013 (MoDisco 2013). After the installation of MoDisco tool, the right-click popup menu of Eclipse will be changed by including another menu bar with MoDisco logo. By tapping the correct mouse catch on one anticipate in the "Bundle Explorer", a popup menu with a menu bar marked "MoDisco" (as highlighted in figures 16 and 17) will appear. Following the options insides this bar, you can obtain KDM model and Java model. After KDM model is obtained, a popup menu with a menu bar named "MoDisco" can be initiated by tapping the correct mouse catch on KDM model item. This item can be found under the structured tree of selected project. Following the alternatives internal parts of "MoDisco" bar, a menu bar labeled "Discover UML model from KDM model" (as highlighted in figure 18) can be found, which can be used to get UML model (MoDisco 2013).

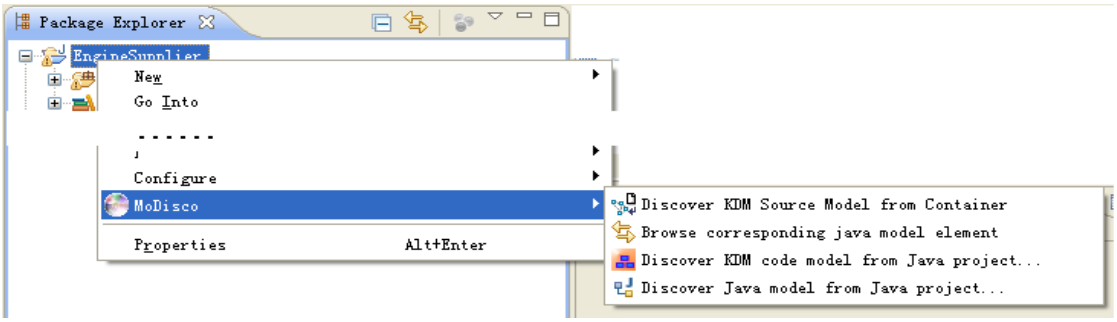

Figure 16. MoDisco features

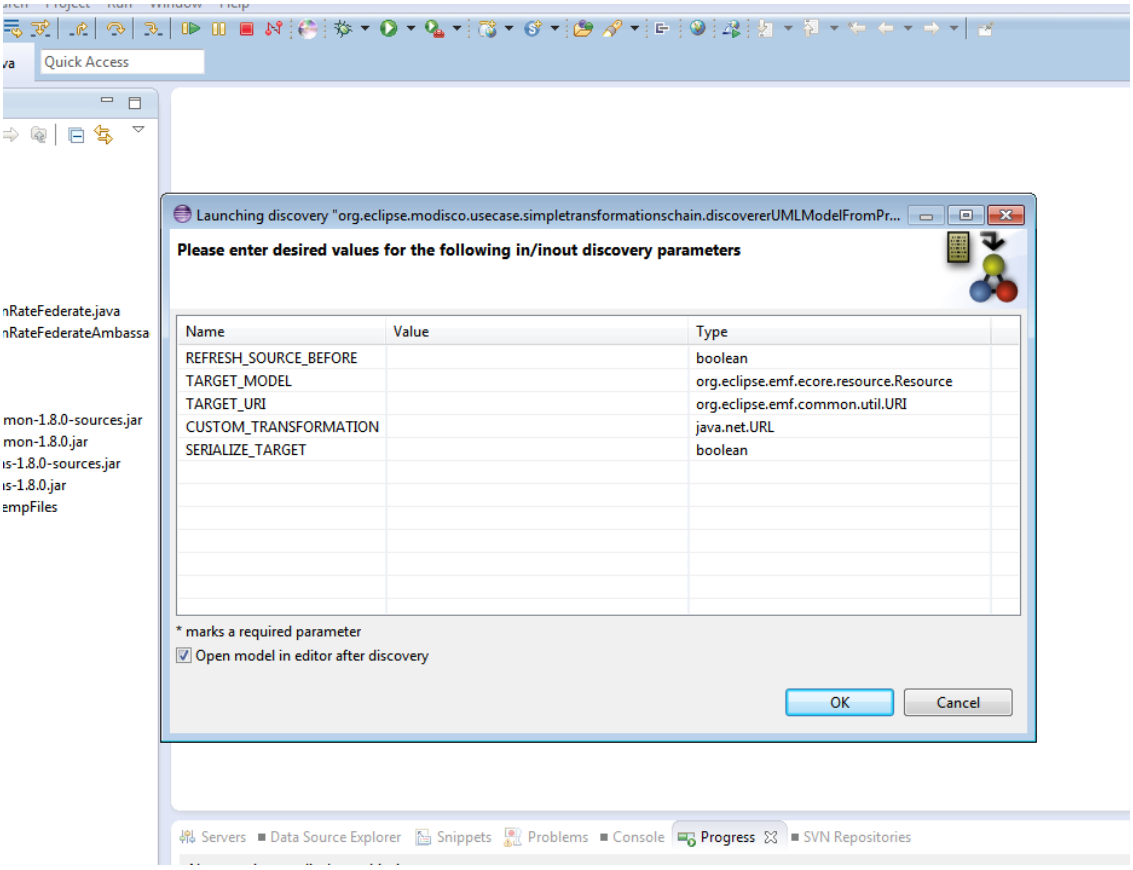

Figure 17. MoDisco configuration and specifications

Figure 18. MoDisco tool usage to generate the UML file

The UML file is generated through the "KDM to UML converter" after the usage of two existing discoverers: 1) JavaDiscoverer which discovers KDM models from java sources, and 2) CSharpDiscoverer which discovers models from C# sources.

As shown in figure 19, the converted UML models include Packages, Interfaces, Classes, and also the properties and operations of classes and associations and dependencies among the classes.

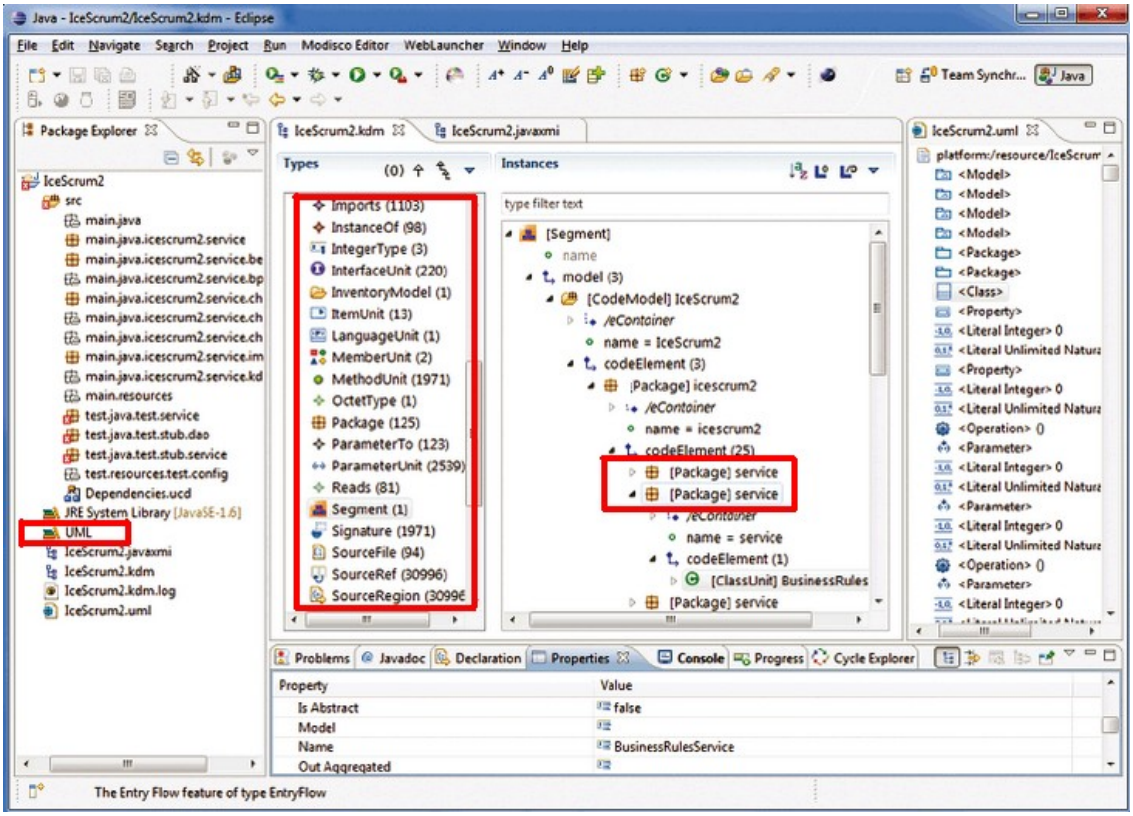

Figure 19. UML Model

In order to generate HLA FOM, a critical phase must be occurred by selecting the similar entities among federates participating in this simulation after comparing their UML models. However, the .uml file would contain bulk information, which is not remarkable for HLA FOM generation, such as external java package and common java data types. Thus, an “Analyze UML” sub-module is required to select the valuable information, and organize these information.

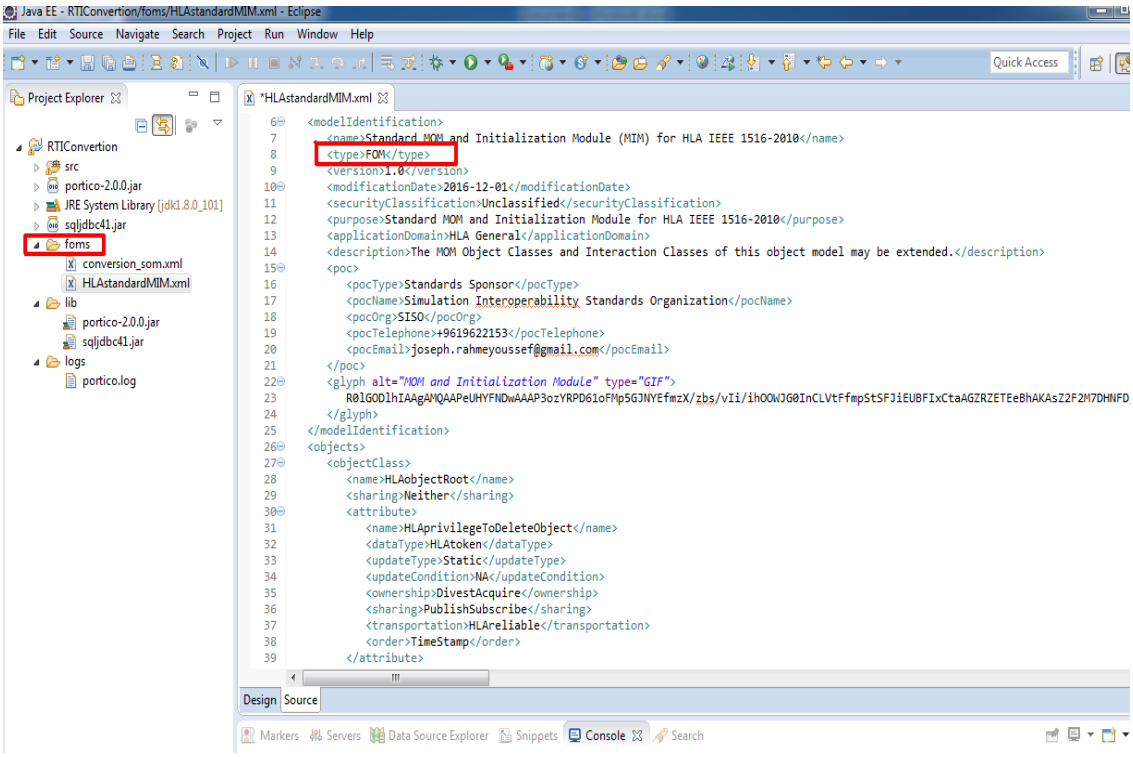

Figure 20. Generation of FOM File

After the analysis of the UML Model, the useful attributes are selected and the models are transferred into HLA FOM file by following different RTI FOM file format as shown in figure 20. According to the poRTIco RTI FOM file format, UML:Class matches to Objects:Class, and UML:Attribute matches to Objects:Attribute (Zhiying, Chen, and Zacharewicz 2012).

4.6 Conclusion

This chapter presented at first the position of this thesis based on the interoperability concept, where EOS is mainly covering the unified and federated approaches especially for the conceptual and organizations barriers in the data, service, process and business levels.

After that, the illustrative architectures related to the Enterprise Operating System are presented and detailed in order to show the different internal components of EOS and how this proposed platform can interact with the peripherals and external components by executing through the Run Time Infrastructure called poRTIco and after generating a FOM file that is playing a big role in supporting the interoperation of distributed simulation.

Chapter 5. Applications: Case studies

This section provides two examples to illustrate concepts and principles of the proposed EOS. The two examples deal with two different application domains: manufacturing and service. The service case deals with the banking sector. It is implemented using the HLA technology for the purpose of proof of concept from the implementation point of view. The manufacturing case focuses on conceptually illustrating how enterprise operations are monitored and controlled through the use of the EOS.

5.1 Banking sector – Case example

The illustration case is concerned with an example in banking operations domain. A simulation system in the Banking and Finance environment performing exchange rate definition and update operations is presented, validated and implemented as a real-world system. The exchange rate is defined as a rate at which a country's currency is exchanged in terms of another currency. Bank exchange rates are constantly changing on every business day based on current market conditions.

In our case, the currencies exchanged are EURO and USD.
Every participating application in this system is called "federate" that interacts with other federates through the Run Time Infrastructure "RTI".
In our case "poRTIco" open source software was chosen as a Run Time Infrastructure that can support heterogeneous environments. It is designed with modularity and flexibility and provides a set of software services for the dynamic information management and inheritance, in which federates coordinate their operations and exchange data during a runtime execution.

Figures 21 and 22 show the EOS Conceptual and Technical architectures of the Bank daily Exchange Rate update operations.

As shown in figure 21, the EOS internal components are connected together and to the external components in order to create a new activity (exchange rate update), analyze this activity, turn it as a process and execute it through special resources and information (Youssef, Zacharewicz, Chen, and Vernadat 2017).

Figure 22 presents the intercommunication architecture between the EOS components through the Run Time Infrastructure in order to execute the exchange rate new activity.

Each EOS component has a special role during the execution:

- *EOS Interface:* EOS interface takes the charge of transmitting the new requested job (exchange rate update) created by the Bank's Branch Manager. After that this interface restricts any financial tasks before this modification to prevent any duplicate or errors on the Bank's daily transactions. Later on, it creates the process of Exchange Rate's update by defining the related database tables, attributes and specifications.
- *Enterprise Process Management:* This EOS component is responsible of ensuring that the Manager's access rights are mentioned as the right privileges to modify the currencies Exchange Rates. After the confirmation, it retrieves the Exchange Rate's currencies (EURO and USD) and the conditional rules associated with this identified operation process, and after that the "PM" maintains a state record of this activity including the requestor information, attributes modified, date and time.
- *Enterprise Interoperability Management:* After the creation and generation of the process, the "IM" has two synchronized jobs: 1) requesting from the Resource Management to assign the specified software, teller and printer for exchange rate execution, and 2) invokes the Enterprise Information Management to acquire the new Exchange value. After the process execution, this component requests from the Resource Management to release the involved resources when terminating.
- *Enterprise Information Management:* When the "IM" notifies the "EIM" about this new task, this EOS component acquires the new Exchange Rate value by specifying the involved information and functions related to this update and then replies to the Interoperability Management with the object states and information objects required.
- *Enterprise Resource Management:* When the job is ready to be executed, this component checks the availability of the Bank's related software, teller and printer in order to pre-assign it when available. Then, it responds to the Process Scheduling requests to allocate and de-allocate these resources and after that it controls the Presentation Management services and the related dialogues to ensure that the right resources are executing the exchange rate update in the right way and at the right time.

- *Enterprise Presentation Management:* As the final stage, this component has the role of generating and executing the new job by modifying the exchange rate on the Bank's core banking system. Then, printing the operation approval for being signed by the authorized personnel (Bank's Branch Manager), and finally notifying the teller to use the new exchange rate value.

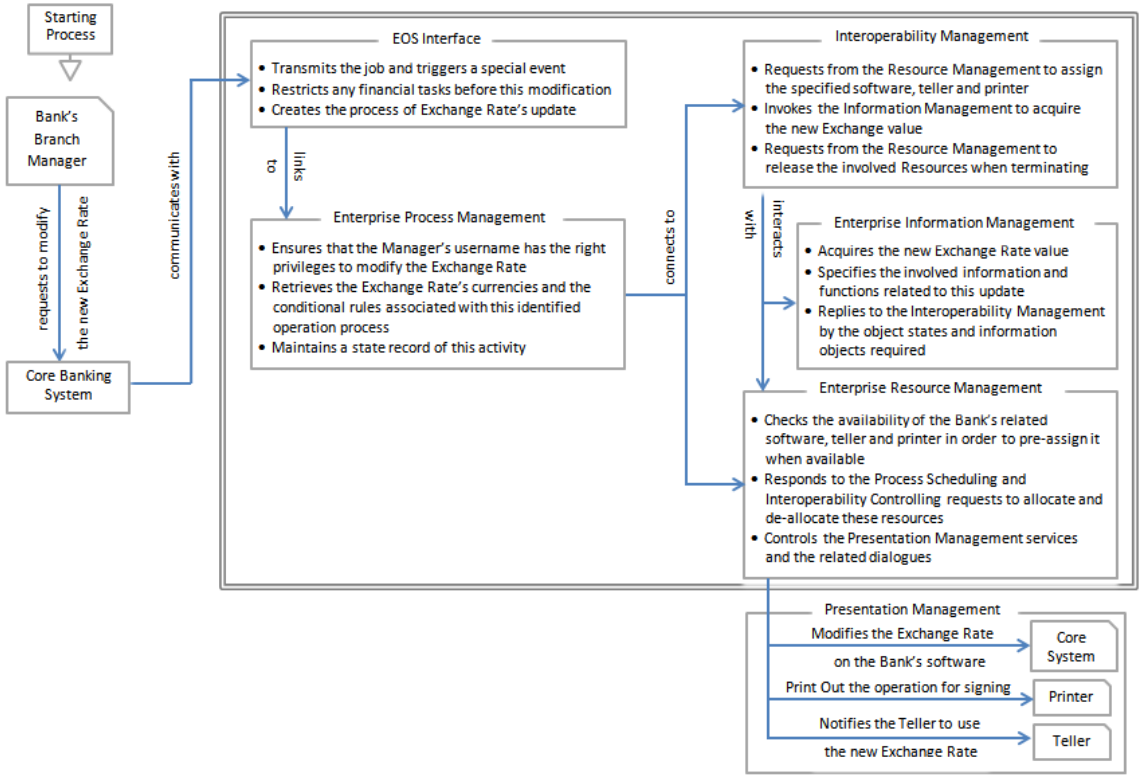

Figure 21: Conceptual and technical architecture of the bank exchange rate update

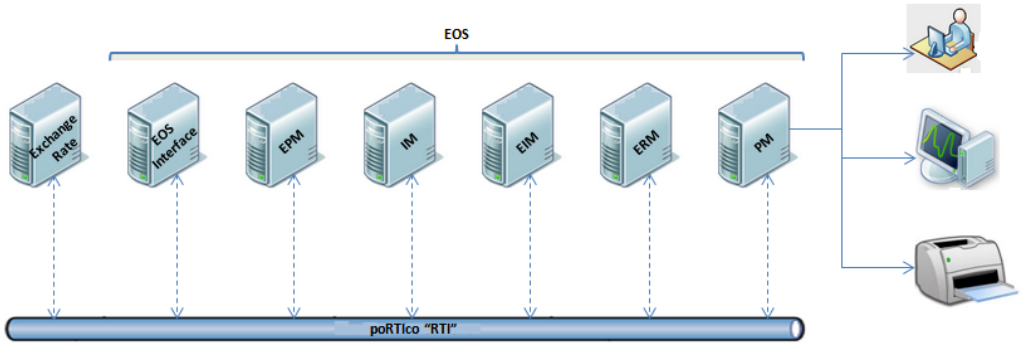

Figure 22: Federates connected to RTI through EOS

Figure 23 shows the EOS Implementation architecture noting that this architecture is designed using HLA technology. As shown in this figure, in the Implementation architecture the seven federates are connected together and they are communicating through poRTIco in order to publish and subscribe events between modules. The "Exchange Rate form" of the Core Banking System and "Presentation Management"

federates are developed and implemented using the Vb.net programming language on the Visual Studio 2010 platform. The other federates, “EOS Interface”, “Enterprise Process Management”, “Interoperability Management”, “Information Management” and the “Enterprise Resource Management” are developed and implemented using the Java language on Eclipse version 6 platform.

- The Exchange Rate form of the Core Banking System is connected to the related SQL storage database. It is used as a user interface that allows the Bank’s Branch Manager to request the modification of the daily currencies rate between EURO and USD.
- The EOS Interface is used to transmit the request job, restrict any operation on the previous rate from any teller and create the process.
- The Enterprise Process Management is used to ensure that the login username and password are compatible with the required privileges and retrieve the exchange rate attributes.
- The Interoperability Management has two roles: (1) to preserve the correct rate value from the Information Management federate and (2) to request from the Resource Management federate to assign the specified external resources (Rate Application, Printer and Teller) to execute the operation.
- The Information Management is responsible of sending the new exchange rate value to the Interoperability Management federate.
- The Enterprise Resource Management is used to respond to the Interoperability Management federate by checking the availability of the related resources and pre-assign it when available.
- The Presentation Management is the federate that modifies the exchange rate on the Rate Application, prints out the operation receipt and notifies the teller to use the new exchange rate.

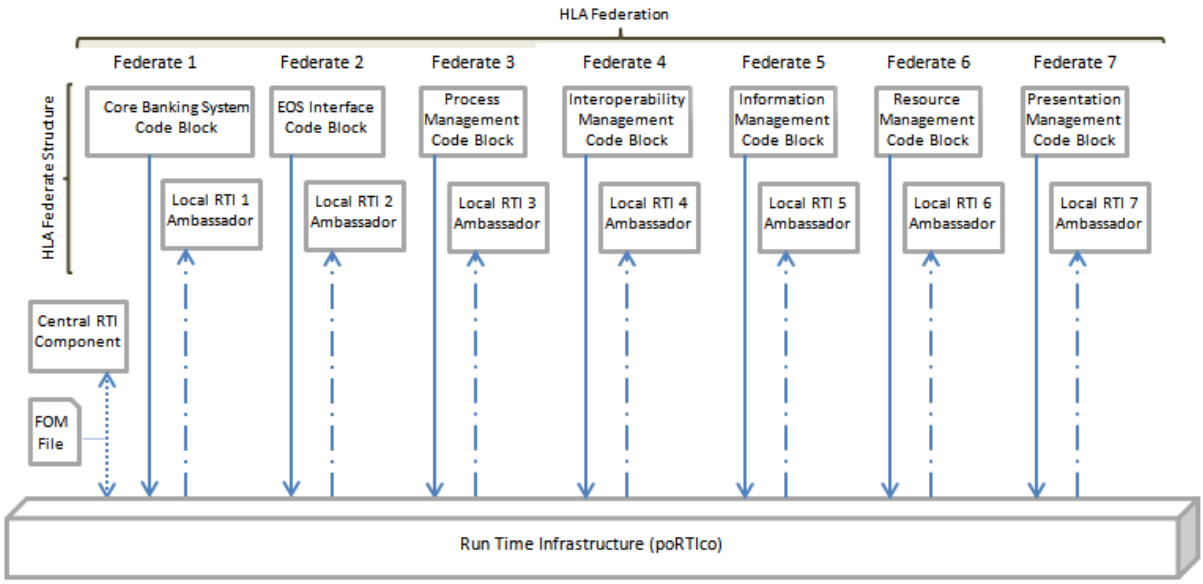

Figure 23: Implementation Architecture of the Bank Exchange Rate update

As shown in figure 24, the BPMN diagram of this case study generated through Yaoqiang software displays the orchestration and interchange of messages and tasks between EOS components and the external resources on a clear and organized queue based on the priority of each task (Falcone et al. 2017).

This BPMN diagram provides an XML scheme and file to extend BPMN towards business modeling and executive decision support. As shown in figure 25, the XML file is representing the workflow parses including the executing of the enterprise's processes through the related resources. It serves as a common language, bridging the communication gap that frequently occurs between business process design and implementation.

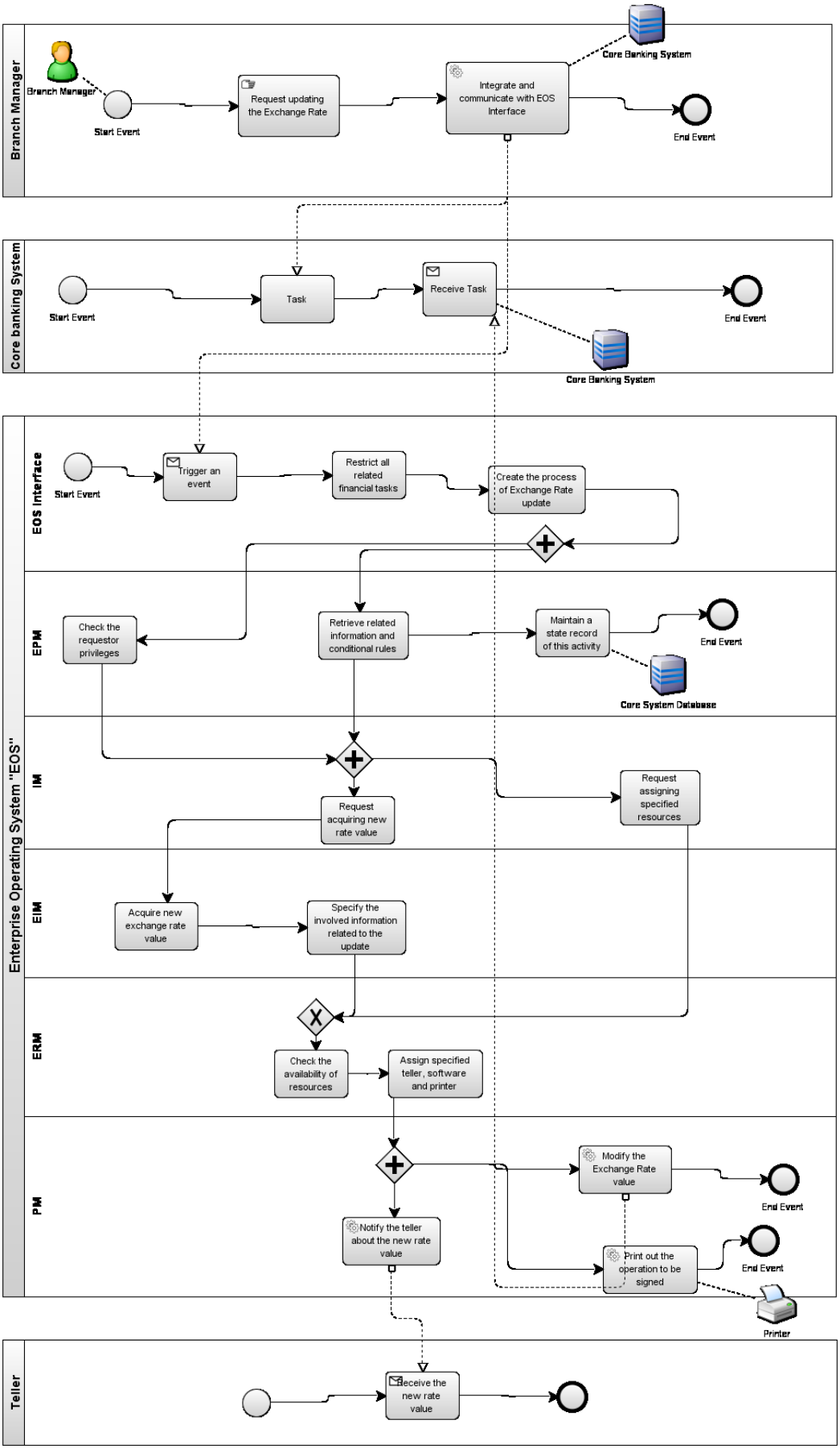

Figure 24. BPMN diagram related to the Banking sector case study

Figure 25. BPMN XML file

As for the technical development and implementation phase, figures 30, 31, 32 and 33 in appendix 1 show the EOS interface, Enterprise Resource Management, the Enterprise Process Management and the Presentation Management federates that are developed in different heterogeneous environments and integrated with poRTIco by calling the member functions of Class RTI:RTIAmbassador, which is contained in the LibRTI, and by extending and implementing the RTI:FederateAmbassador that uses pure virtual functions to send messages and requests to the RTI(Youssef, Chen, and Zacharewicz 2017).

- *The EOS interface federate* (figure 30 in appendix 1) contains the "reflectAttributeValues" function and its related parameters used in order to restrict any access for Read/Write on the database and change the status to "ON" as a notification for the Enterprise Operating System that a new job is taking place.
- *The Enterprise Process Management federate* (figure 31 in appendix 1) contains the "getSelectedItem" function that is developed and deployed in order to retrieve the user access rights, compare it and ensure that he has the right privileges for exchange rate update activity. The "actionPerformed" function is mainly responsible of defining and storing the logs and specifications related to the process execution.
- *The Enterprise Resource Management federate* (figure 32 in appendix 1) contains the "receiveInteraction" function that is executed and generated in order to receive all execution's information including the new value of exchange rate and the assigned resources' characteristics.

- *The Presentation Management federate* (figure 33 in appendix 1) comprises the "btn_print_Click", "btn_modify_Click" and "btn_notify_Click" functions that are developed and implemented in order to assign the related resources and execute the job through the related Human, Machine and IT dialogues.

As shown in appendix 2 (figures 34 and 35) and in appendix 3, KDM is generated as a meta-model for representing existing federates and their specifications, as well as relationships among the function models in the system. It is developed in order to ensure the interoperability and data exchange between existing Exchange Rate federates.

The second phase of generating the FOM file is by discovering the UML model from the KDM model" (as shown in appendixes 2 (figures 36 and 37) and 4). The UML model generated from the MoDisco Tool is saved in XML format and as an .uml file. Each item of the tree structure has a "xmi:id" and "name" to be distinctively identified, and is also used for class dependency and association. On the other hand, the .uml file would contain bulk information, which is not interesting for HLA FOM generation, such as external java package, common java datatypes, and etc. Thus, an "Analyze UML" sub-module is essential to select and arrange the valuable information.

The FOM File (as highlighted in appendixes 2 (figures 38 and 39), 5 and 6) comprises two xml files: "conversion_som.xml" and HLAstandardMIM.xml files that are generated from MoDisco through the UML file in order to define the structure of all information that is available to be exchanged among federates. In this simulation, FOM plays as shared concepts between all federates of the HLA federation, which represents the established consensus of the collaborative federates.

Noting that this poRTIco project (RTIBasic) located in the path "C:\workspace\RtiBasic" can execute through the Eclipse platform "Luna\RTIconversion\src\edu\bordeaux\jrahme".

At first, the "Main" class executes the "run_federate" function in order to start the federation by creating the RTIAmbassador and retieving the required information from the FOM file. After that the federates join the federation and run the "publish&subscribe" function. Next, the "Conversion" class performs the "Exchange Rate Operation" function. Afterward, the "Conversion_Rate" function draws the interface and query the database in order to retrieve the exchange rate new value. And

finally the “updateattributevalue” function sends the conversion model to the RTIAmbassador in order to notify all assigned resources to perform their tasks and operations. In order to confirm and approve the validity of the Exchange Rate update, this experimental case study has been fully implemented through several heterogeneous environments using the .net and java programming languages based on the Interoperability and Uniformity principles to provide a set of domain-independent APIs used to access capabilities and features and to exchange data between federates using the XML format.

It must be noted that the scenario described above approved the integration of several heterogeneous federates with poRTIco (as appeared in appendixes 2 (figure 40) and 7). Exchange rate has been transmitted, updated and executed through the EOS requirements.

5.2 Manufacturing sector – case example

This section presents a simplified case example to illustrate how an EOS behaves in a manufacturing enterprise to monitor and control enterprise operations (see figure 26). The example focuses on business operations performed by the EOS without detailing its internal and technical functioning.

The studied enterprise is a SME and works on ‘Production to order’. The main products are ‘Modular shelves’ with standard and customized (with new design) options. The resources of the company are of the three types that one can encounter in most of the companies (computing, machining and human):

- Commercial service: Excel software (PC)
- Production Management: MRPEasy software (PC)
- Design service: DeltaCAD software
- Delivery: DISPATCH! software
- Machining: NC machine
- Assemble: Human operators
- Painting: Robots (3 units)

The main problems encountered in the enterprise are: (i) impossibility to know in real time what is happening in the workshop; manufacturing orders are often late due to the

unavailability of resources; (ii) too much dead-time between operations due to inadequate planning and synchronization of these operations and (iii) some enterprise software applications are not interoperable (e.g. MRPEasy, DeltaCAD and DISPATCH!), it is necessary to arrange the structure of some data tables and perform the translation of some terms used in different software.

The motivation of the enterprise to implement an EOS is to be able to more efficiently monitor and control its resources and the daily operations carried out on these resources. More specifically, the aims of the case example are to:

- show that the resources can work together and be coordinated by the EOS, in particular:
  - Resources are monitored in real time (available, occupied, etc.)
  - Resources assignment is done in real time
- show that heterogeneous applications can interoperate through the EOS, in particular:
  - Semantic interoperability (data meaning)
  - Syntax interoperability (data structure)

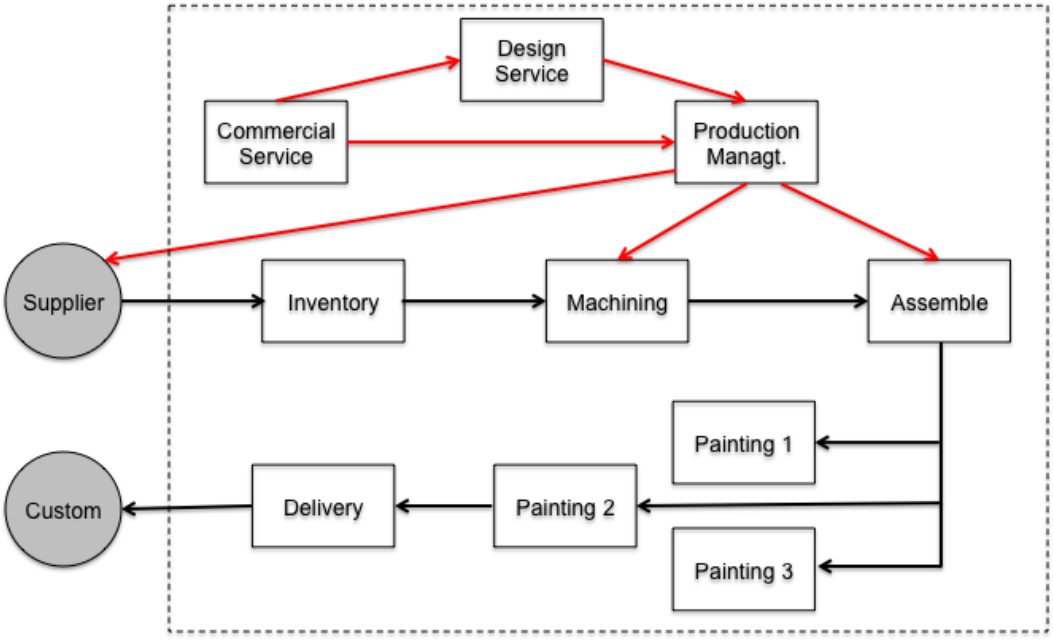

Figure 26: Structure of the enterprise with main information flow (Red line) and physical flow (Black line)

The commercial service receives customer orders and registers them using a simple Excel software. Orders requesting standard products will be directly sent to Production Management to plan their production while those requiring a customization will be first

processed by Design Service and then sent to Production Management. Production Management will issue purchasing orders (send to external supplier) and manufacturing orders for production to NC machine and assembly line. Purchased raw materials will be first stored in the inventory, and then moved to the machining workstation (NC machine) to manufacture shelves components. After that, the assembly operations are done by human operators to obtain shelves. The assembled shelves are pained using a robot (only one robot is needed at a time for one order). Then the finished products are delivered to customers according to a pre-defined delivery plan. Figure 26 illustrates how the EOS deals with one particular customer order that necessities a redesign for a customized shelve.

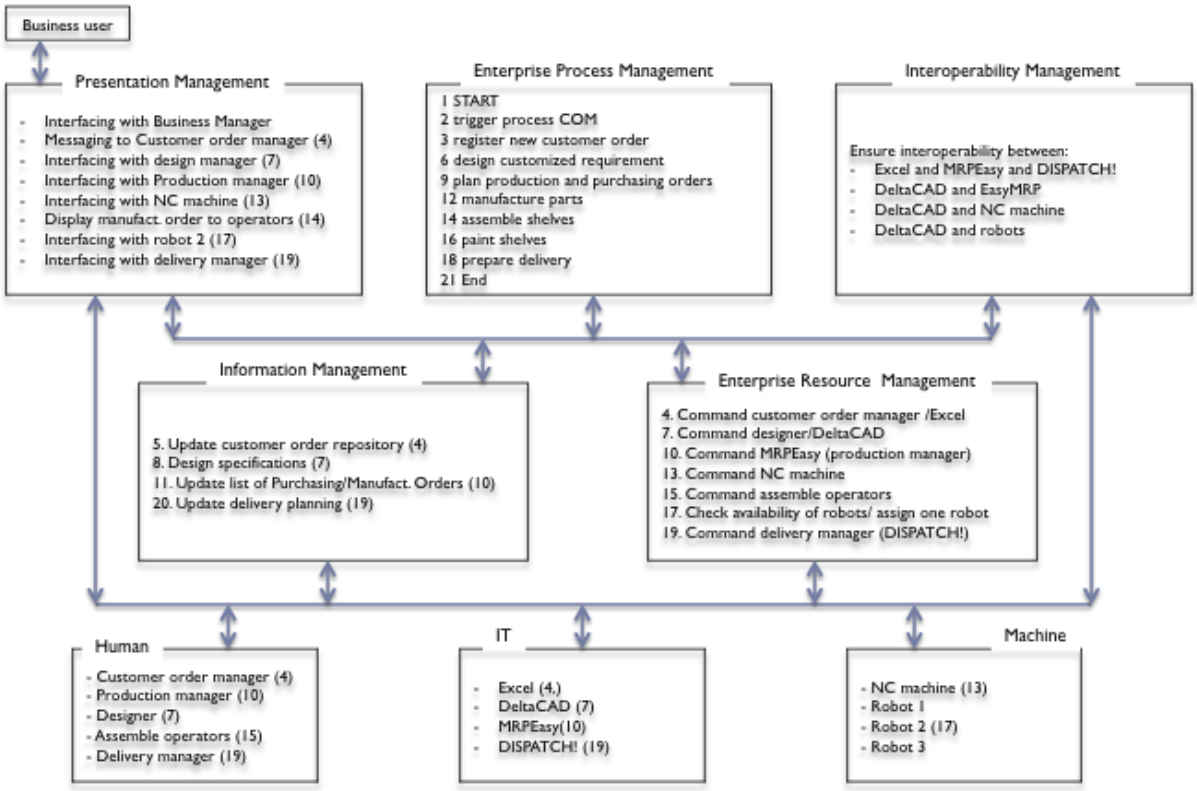

Figure 27: Manufacturing case scenario

The purpose of the presented scenario is to illustrate how the envisioned EOS can coordinate and command the needed operations to produce required products. The set of operations are numbered in sequence as shown in figure 27. The resource needed and assigned by the EOS to perform each requested operation (with the corresponding operation number in bracket) is also indicated.

1) Arrival of a new customer order with a new design requirement. The business manager decides to validate the order and asks to start the operations (EPM).
2) EPM triggers the COM (Customized Order Management) process (it is a pre-

defined process stored in the business process repository)

3) EPM interprets the first operation ‘register new customer order’ and sends a request to ERM
4) ERM informs the customer order manager via the ‘Presentation Management’ interface about the request
5) The Excel file of customer order list is updated (IM)
6) EPM interprets the next operation of the COM process: ‘design product’ to meet the customized requirements
7) ERM checks the availability of required resources and triggers Designer / DeltaCAD application
8) Design specifications are saved in a set of files with the help of Information management (IM)
9) EPM interprets the next operation: ‘plan purchasing and manufacturing orders’
10) ERM checks the availability and triggers Production manager/MRPEasy application
11) Purchasing and Manufacturing Order lists are updated with the help of IM. Purchasing order is sent to external suppliers
12) EPM interprets the next operation ‘manufacture parts’
13) ERM checks the availability of required resource and triggers ‘NC machine’ via Presentation Management. The ‘NC machine’ receives the technical specifications of parts via IM to manufacture the needed parts
14) EPM interprets the next operation: ‘Assemble shelves’.
15) ERM checks the availability of human operators and sends a ‘start to work’ command via the Presentation Management interface. Operators receive technical specifications of shelves via IM to perform the assemble operations
16) EPM interprets the next operation: ‘Paint shelves’
17) ERM checks the availability of ‘robots’ and assign one available robot (robot 2) to the task
18) EPM interprets the last operation: ‘prepare delivery’
19) ERM informs the delivery manager and ask him to plan delivery using DISPATCH!
20) The delivery plan is updated (IM)
21) End of the process

Figures 28 and 29 illustrate the BPMN diagram and the generated XML file related to the Manufacturing sector's case study. These figures display the flowchart and priorities of process execution through the Interoperability Management component and all other EOS components (Falcone et al. 2017).

The five EOS requirements are presented in the Enterprise Operating System pool (as shown in figure 28), that are connecting and communicating with the enterprise resource's in order to execute the enterprise's activities and operations.

As shown in figure 29, the process XML file consists of the top part that contains the definition of the different manufacturing nodes and their properties, and the lower part that contains all graphical information, like the location of the nodes. This file contains parameters related to the process execution (type, name, id and package name).

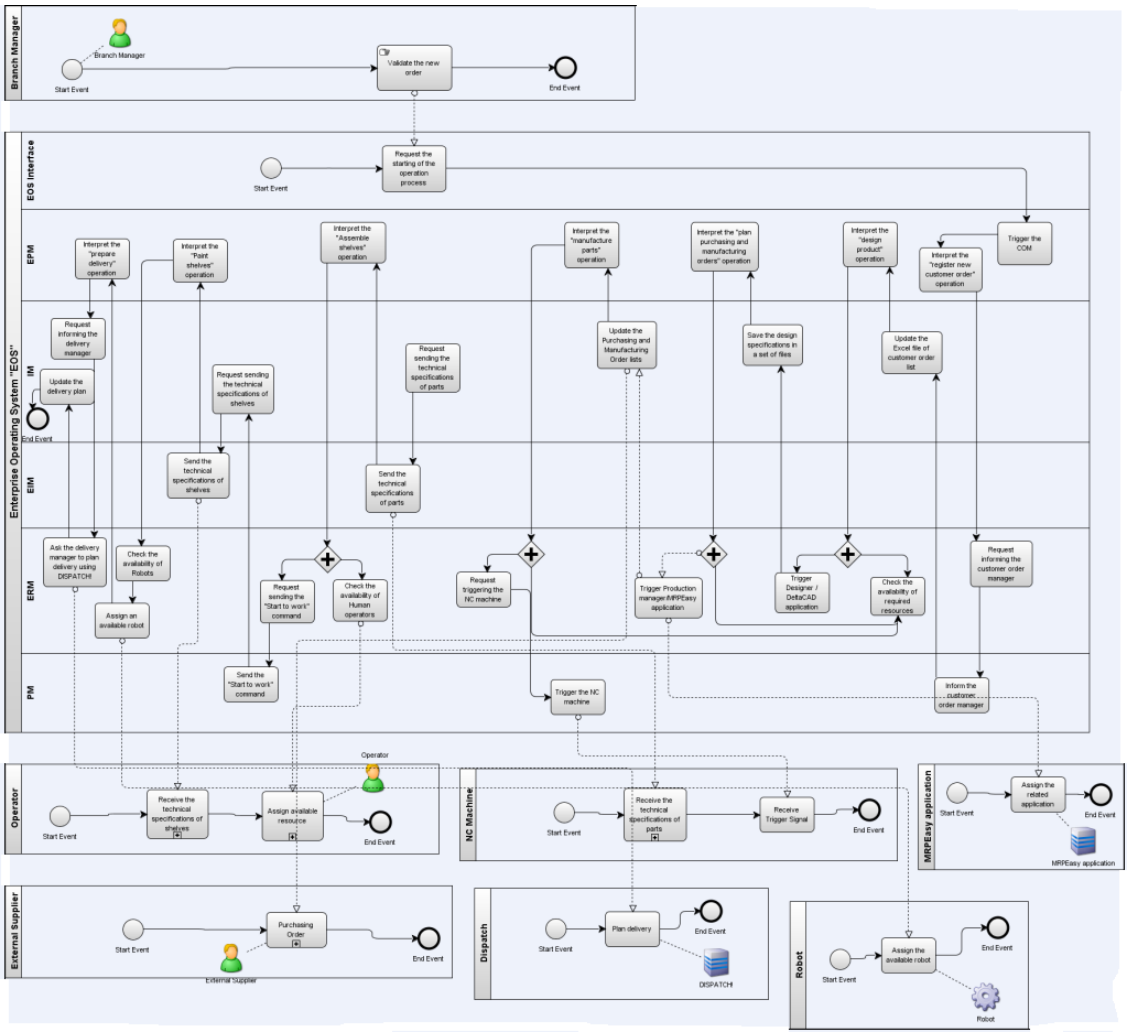

Figure 28: BPMN diagram related to the Manufacturing sector case study

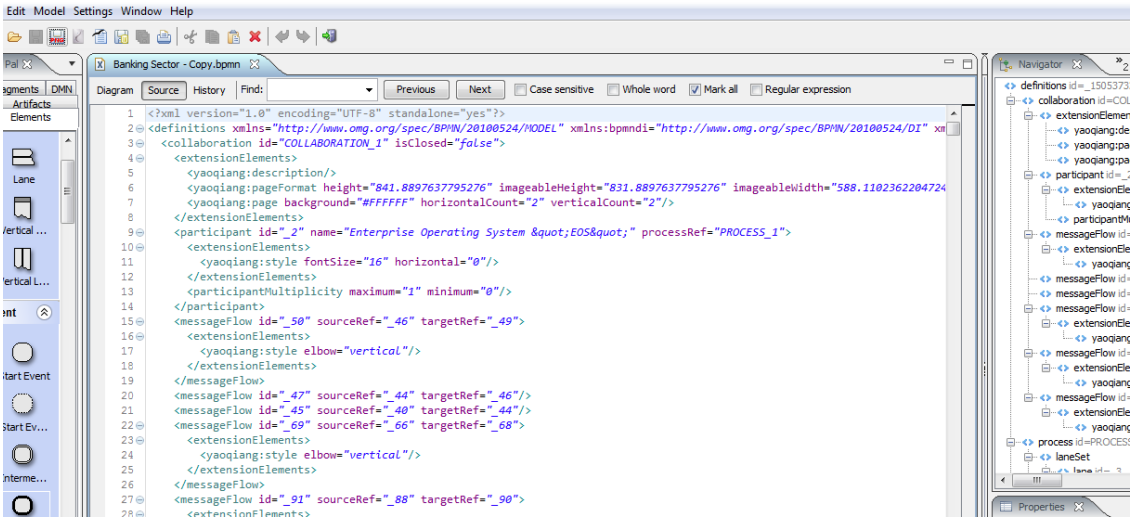

Figure 29: BPMN XML file related to the Manufacturing sector case study

It must be noted that the scenario described above has hidden some technical details on information exchanges between EOS functions. For example:

- at the end of each operation, a feedback information is sent to the EOS by the resource concerned to de-allocate the resource and trigger the next operation defined in the process
- some interoperability engineering activities are required before executing some operations. A set of Interoperability Utility services in IM module is responsible for making necessary mappings and harmonization between the software applications

5.3 Conclusion

This chapter presented two case examples in both manufacturing and service sectors to illustrate how to implement an EOS and how it behaves to monitor and control operations.

The first case in the banking sector has been fully detailed and implemented using several heterogeneous environments where federates were developed with different programming languages and integrated with portico using HLA technology based on the Interoperability and Uniformity principles in order to prove the efficiency and effectiveness of EOS in the service sector.

The second case study in the manufacturing sector focused on illustrating how various EOS functions work together to run an enterprise. It shows that basic functions of ERP, Workflow and enterprise integration are coordinated by the EOS.

Chapter 6. General conclusion and perspectives

The aim of this thesis was to tentatively present requirements, methodologies and architectures for developing an Enterprise Operating System (EOS) for the emerging generation of new enterprise systems as envisioned in the Fourth Industrial Revolution. A detailed state-of-the-art analysis has been performed to assess existing relevant works. It has been concluded that none of the existing approaches/ initiatives was previously developed as an Enterprise Operating System. Although they all contribute to some extents to support enterprise operation management, they failed to act as a system-wide interface between business managers who monitor and control enterprise systems as a whole, and enterprise resources performing local daily enterprise operations. The proposed EOS tends to reconciliate two different but complementary approaches / initiatives for enterprise management and control that exist in the market: IT platforms / infrastructure and ERP based application packages as assessed in the state-of-the-art.

Chapter 1 identified and defined the scope and objectives of the research carried out in this thesis. Firstly, it presented the context, background and requirements including the economic and industrial context and the research background. And then, it presented the research motivations and priorities. Afterwards, it analyzed the current situation of enterprise management and control and expatiated on the research challenges and the contribution of the thesis. Finally, it pointed out the objective and expected results of this doctoral research according to the research challenges.

Chapter 2 made a survey of the existing concepts, definitions and architectures that are relevant to Operating System, Simulation and Interoperability. Firstly, it presented the definitions and architectures of Computer Operating System including the structure and the different components. After that, the concept of Distributed Simulation and Enterprise Interoperability approach are defined and identified from the views of conceptual, organizational, and technological barriers. Later, it reviewed the existing relevant models for Enterprise Interoperability to identify the relevant concepts, methods, and principles that can be useful suggestions for solving Enterprise Interoperability including the model driven technologies, software and application distribution frameworks, and the High Level Architecture “HLA” into details.

Afterwards, a set of requirements and functionalities related to EOS are identified in

Chapter 3, and a survey of previous relevant works is presented and results are compared to the requirements of EOS by comparing the previous models, methodologies and technologies in the same domain to point out their advantages for promoting enterprise monitoring and control, and also their shortfalls that do not satisfy with the requirements of EOS.

Chapter 4 defined the position of the proposed EOS based on the Enterprise Interoperability Framework. Then it presented the main contribution of this research work. Based on the requirements and state-of-the-art results, the conceptual, technical and implementation architectures are outlined including all internal and external components of EOS:

- Conceptual architecture: it is a representation of the main functions of a system from the point of view of its use. It is independent of how it technically works and is implemented using a specific technology.
- Technical architecture: it describes how components or services of a system interact to fulfill the required functions. The technical architecture is independent of a specific technology for implementation.
- Implementation architecture: it describes how to physically realize a system by defining the technology and concepts used to implement the technical architecture.

Chapter 5 explained how EOS can be implemented in both service and production environments. After that, it presented two case studies in the Banking and Manufacturing sectors showing that EOS can be implemented in several sectors and fields same as executed in these two case studies:

- Banking Sector: The illustration case is concerned with an example in banking operations domain. A simulation system in the Banking and Finance environment performing exchange rate definition and update operations is presented, validated and developed as a real-world system. This experimental case study has been fully implemented in several heterogeneous environments based on the Interoperability and Uniformity principles to provide a set of domain-independent APIs used to access capabilities and features and to exchange data between federates using the XML format.
- Manufacturing sector: This example illustrates how an EOS behaves in a

manufacturing enterprise to monitor and control enterprise operations. The example focuses on business operations performed by the EOS without detailing its internal and technical functioning. The purpose of the presented scenario is to illustrate how the envisioned EOS can coordinate and command the needed operations to produce required products. The set of operations are numbered in sequence and the resource needed and assigned by the EOS to perform each requested operation is also indicated.

The requirements presented in this thesis are based on and inspired from some existing relevant approaches, in particular ENV13550 with necessary generalization and extension to focus on the core functions of an Operating System for production enterprises. EOS will mainly provide an alternative for Small and Medium-sized Enterprises "SMEs", support establishing enterprise interoperability in a heterogeneous environment and facilitate re-use of models and re-engineering sub-systems based on models. Noting that the Interoperability is still a conceptual concept in this doctorate research, it is used as a prototype to ensure the interconnection between the enterprise heterogeneous resources and the central orchestrator "EOS".

The proposal is challenging and its success mainly depends on three factors: 1) The acceptance of EOS in industry as a standard to develop an ecosystem providing varieties of enterprise applications compatible to EOS. 2) The compatibility of existing applications with EOS and how to create an EOS's compatible configuration that can interact with any prior existing applications in the market. 3) The 'Interoperability' service that allows other heterogeneous non-EOS compatible applications to run on EOS.

EOS is mainly developed and implemented as one simplified central orchestrator component connected to several peripheral devices and external components. It is critical for SMEs in order to ensure highly flexible, dynamically configured and real time controlled enterprise.

The reason for proposing to develop an EOS is first of all to contribute a part of the response to the foreseen enterprise evolution towards a new generation of modular structured agile 'smart factory' based on Cyber Physical System principles. Traditional factory automation, empowered with Internet of Things (IoT), will operate a challenging industrial migration towards the Fourth Industrial Revolution (or Industry 4.0). The massive use of connected sensors and mobile devices connected to or

assisting humans will make possible to collect timely real time information throughout the enterprise. Using the real-time data on enterprise operations, events and resources, an EOS will allow the dynamic scheduling of enterprise operations with on-the-fly resources allocation for ensuring the flexibility and real time control of enterprise functions.

In line with the development of the future generation enterprise manufacturing systems based on Internet of Things (IoT) and Cyber Physical System (CPS) principles, EOS helps improving the business industry and technology development by:

- Ensuring that all enterprise's functions and requirements are covered during the EOS execution
- Ensuring that the interoperability approach is operating correctly
- Accelerating the development process of enterprise tasks and phases
- Reducing the time and cost of enterprise activities and operations
- Reducing the enterprise's systems and deploying only the required heterogeneous applications.

Once EOS is adopted in the industry, a foreseen ecosystem would be developed in consequence to provide varieties of enterprise applications compatible to EOS, just like what happened with Apple's ecosystem to IOS (iPhone Operating System) and Google's one to Android.

As for the perspectives, a future work will be achieved to develop the technical architecture and functionalities of the Interoperability concept based on the prototype developed in this thesis, in order to test it on real and complex case studies.

IN addition, a Graphical User Interface "GUI" should be developed which views the directory structure and requests the services from the "EOS" by allowing the use of icons or other visual indicators to interact with electronic devices rather than using only text via the command line. Moreover, an "EOS" password protection tool needs to be added to keep unauthorized users out of the system by maintaining activity logs and providing backup and recovery routines for starting over in the event of a system failure.

Another perspective is to develop an international standard. The main concern is to create a general consciousness and consensus both in manufacturing and IT industries to accept EOS as a standard in order to develop and commercialize compatible enterprise software applications. To achieve this objective, it is necessary to proceed in

steps in a progressive way:

- Involve industry and appropriate standardization working groups to collect and consolidate requirements on EOS
- Define standard specifications of EOS based on the requirements and build demonstration industry use cases
- Create user interest groups and associations to disseminate EOS and provide technical support and consultancy to end users.

To start, the CEN 13350 initially drafted to support Enterprise Model Execution and Integration can be used as a basis with necessary extensions and adaptations towards a sound Enterprise Operating System.

References

AIIM. 2000. “Enterprise Applications - Adoption of E-Business and Document Technologies.” AIIM BookStore, 2000. http://www.techstreet.com/direct/ExSum_NA.pdf.

Al-Shaikh, Sara. 2005. “Operating System.” The Computer Language Company Inc. / Encyclopedia. http://www.pcmag.com/encyclopedia/term/48510/operating-system.

AMICE. 1993. “CIMOSA- Open System Architecture for CIM, 2nd and Extended Revision, Esprit Consortium AMICE.” Springer-Verlag Berlin.

Anguish, Scott. 2016. “Operating System Structure.” “WWDC 98 Summary” paper. https://en.wikipedia.org/wiki/Architecture_of_OS_X.

ATHENA. 2003. “Advanced Technologies for Interoperability of Heterogeneous Enterprise Networks and Their Applications.”

Aurousseau, M., E. Ballot, A. Bernard, D. Brissaud, S. Caroly, Y. Frein, B. Grabot, and V. Rocchi. 2013. “FUTURPROD - Les Systèmes de Production Du Futur.”

Bassi, Sachin Kumar. 2017. “The Evolution of the MVS Operating System.” Computer Hope. http://www.computerhope.com/os.htm.

Beal, Vangie. 2015. “Operating System.” Webopedia Press. http://www.webopedia.com/TERM/O/operating_system.html.

Bézivin, J., S. Gérard, P-A Muller, and L. Rioux. 2003. “MDA Components: Challenges and Opportunities.”

Bic, Lubomur F., and Shaw, Alan C. 2003. “Operating Systems.” Wikipedia. https://en.wikipedia.org/wiki/Operating_system.

Burnson, F. 2015. “Enterprise Resource Planning Software - Buyer Report.” Software Advice™. Stamford, England.

Buss, A., and L. Jackson. 1998. “Distributed Simulation Modeling: A Comparison Of HLA, CORBA And RMI.” Winter Simulation Conference.

Callahanp, Hooray. 2013. “Developing with HLA.” FlightGear wiki. http://wiki.flightgear.org/Developing_with_HLA#OpenRTI.

CEITON. 2014. “‘Front-End and Back-End EAI’, CEITON Technologies.” Wikipedia. https://en.wikipedia.org/wiki/Enterprise_application_integration.

CEN. 1999. “prENV 13550 ”Advanced Manufacturing Technology - Systems Architecture - Enterprise Mode1 Execution and Integration Services”.” Secrétariat du CEN TC 310 (dot. CEN TC 310 N 706). Paris, France. “Association Française de Normalisation.”

Chang, J.F. 2005. “Business Process Management Systems, Strategy and Implementation.” CRC Press.

Chappell, D.A. 2004. “Enterprise Service Bus.” O’Reilly Media, ACM Digital Library. ©2004. ISBN: 0596006756.

Chen, D. 2009. “Framework for Enterprise Interoperability.” Congrès International de Génie Industriel (CIGI2009).

Chen, D., G. Doumeingts, and F. Vernadat. 2008. “Architectures for Enterprise Integration and Interoperability: Past, Present and Future. Computers in Industry.”

Chen, D., J.R. Youssef, and G. Zacharewicz. 2015. “Towards an Enterprise Operating System – Requirements for Standardisation, Proceedings of IWEI 2015.” IWEI Workshops 2015.

CIMOSA. 2009. “CIMOSA Association e.V. - About Us.”

De Lucia, A., A. Fasolino, and E. Pompella. 2001. “Decisional Framework for Legacy System Management.”

Deitel, Harvey M., Paul Deitel, and David Choffnes. 2003. “Operating Systems.” Pearson/Prentice Hall.

Diercksen, V. 2012. “What Is ERP?” http://www.erpsoftwareblog.com/2012/03/do-you-know-the-history-of-erp/.

Doshi, Rajiv, and Gerardo-Pardo Castellote. 2006. “A Comparison of HLA and DDS.” Real-Time Innovations.

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

ESoCE-Net. 2012. “COIN Project - Enterprise Interoperability and Collaboration - Results.”

Fagg, G.E., K.S. London, and J.J. Dongarra. 1996. “Taskers and General Resource Managers.” Third European PVM Users’ Group Meeting, EuroPVM ’96. Springer, Berlin, Heidelberg.

Falcone, A., A. Garro, A. D’Ambrogio, and A. Giglio. 2017. “Engineering Systems by Combining BPMN and HLA-Based Distributed Simulation.”

Fiedler, Ryan. 1983. “The Unix Tutorial / Part 3: Unix in the Microcomputer Marketplace.” Retrieved 30 January 2015.

Flurry, G., and K. Clark. 2011. “Enterprise Service Bus.” IBM – DeveloperWorks. © Copyright IBM Corporation.

Franklin, Curt, and Dave Coustan. 2000. “How Operating Systems Work.” http://computer.howstuffworks.com/operating-system.htm.

Fujimoto, Richard. 2015. “The High Level Architecture: Introduction.”

Fujimoto, R.M. 2000. “Parallel and Distributed Simulation Systems.” Wiley

Interscience.

Gable, J. 2002. “Enterprise Application Integration.” Information Management Journal (Magazine Article). questia Trusted online research.

Grabot, B. 2012. “GDR MACS, Prospectives STP, Sciences et Techniques de La Production de Biens et de Services.”

Gray, George. 2014. “Real-Time Control System Software.” Wikipedia. https://www.revolvy.com/main/index.php?s=Real-time%20Control%20System%20Software&item_type=topic&sr=50.

Gray, J., and A. Reuter. 1993. “Transaction Processing: Concepts and Techniques.” Morgan Kaufmann Publishers (An Imprint Elsevier). San Francisco, California.

Henning, M. 2006. “Response to ‘The Rise and Fall of CORBA.’” ACM 2006 Article. http://queue.acm.org/detail.cfm?id=1142044.

Hermann, Pentek, and Otto. 2016. “Design Principles for Industrie 4.0 Scenarios.”

IEEE. 1995. “Standard for Distributed Interactive Simulation - Communication Services and Profiles.”

IEEE, . 2017. “IEEE Standard for Distributed Interactive Simulation - Application Protocols".”

IEEE 1516. 2010. “‘Standard for Modelling and Simulation High Level Architecture’. Framework and Rules.” IEEE Xplore – Digital Library. New York, USA. IEEE International Conference on, pp. 1-9, 2011.

ISO 14258. 1999. “Industrial Automation Systems – Concepts and Rules for Enterprise Models.” ISO TC184/SC5/WG1.

ISO/IEC. 2009. “ISO/IEC 10746:2009, Information Technology – Open Distributed Processing – Reference Model: Architecture.” International Standards Organization, Geneva, Switzerland.

Jin, X. 2009. “Research on the Model of Enterprise Application Integration with Web Services.” Proceedings of the 3rd WSEAS International Conference on COMPUTER ENGINEERING and APPLICATIONS (CEA’09). Beijing, 100081, China.

Jonah, K. 1999. “Enterprise Operating Systems.” GCN - Technology, Tools and Tactics for Public Sector IT. https://gcn.com/Articles/1999/04/ENTERPRISE-OPERATING-SYSTEMS.aspx?Page=1.

Knight, P., A. Corder, R. Liedel, J. Giddens, R. Drake, C. Jenkins, and P. Agarwal. 2002. “Evaluation of Run Time Infrastructure (RTI) Implementations.” Huntsville Simulation Conference. Huntsville.

Konstantas, D., J.P. Bourrières, M. Léonard, and N. Boudjlida. 2006. “Interoperability of Enterprise Software and Applications.” INTEROP-ESA International Conference. Geneva, Switzerland.

Kosanke, K. 1995. “CIMOSA - Overview and Status.” Computers in Industry 27.

Kuijpers, Nico, Johan Lukkien, Bas Huijbrechts, and Marco Brassé. 2005. “Applying Data Distribution Management and Ownership Management Services of the HLA Interface Specification.” SemanticScholar. https://pdfs.semanticscholar.org/278e/25d4e9ad0c71b06d4211f7498bcee0c07464.pdf.

Liffler, Markus, and Andreas Tschiesner. 2013. “The Internet of Things and the Future of Manufacturing | McKinsey & Company.” Mckinsey.com.

Linthicum, D.S. 1999. “Enterprise Application Integration.” Addison-Wesley Information Technology Series. Canada.

May, J.M. 2001. “Parallel I/O for High Performance Computing.” ACM Digital Library. Boston, United States of America.

McCarty, B., and L. Cassady-Dorion. 1998. “Java Distributed Objects. SAMS Publishing: Indianapolis.”

MoDisco. 2013. “MoDisco User Guide.” http://wiki.eclipse.org/MoDisco/Installation.

Moon, J., and S. Lee. 2006. “Design and Implementation of a Resource Management System Using On-Demand Software Streaming on Distributed Computing Environment.” ICCS’06 Proceedings of the 6th international conference on Computational Science - Volume Part I. Berlin, Germany.

Mstrohlein. 2011. “Enterprise Operating System Definitions.” IT/Business Alignment. http://www.agilebusinesslogic.com/blog/?p=284.

Noulard, Eric, Jean-Yves Rousselot, and Pierre Siron. 2009. “CERTI, an Open Source RTI, Why and How.”

OMG. 2003. “MDA Guide Version 1.0.1. Object Management Group.” www.omg.org/docs/omg/03-06-01.pdf.

OMG, OMG. 2010. “Architecture Driven Modernization (ADM): Knowledge Discovery Meta-Model (KDM) v1.2.” http://www.omg.org/spec/KDM/1.2.

OMG, OMG. 2012. “Common Object Request Broker Architecture (CORBA), Version 3.3.” Object Management Group. http://www.omg.org/spec/CORBA/3.3.

Omicini, A., P. Petta, and J. Pitt. 2013. “Engineering Societies.” Agents World 4th International Workshop London.

Patrick, Robert. 1987. “General Motors/North American Monitor For The IBM 704 Computer.” RAND Corporation.

Petrie, Charles. 1992. “Enterprise Integration Modeling.” MIT Press. Proceedings of the First International Conference.

Pokorny, T. 2012. “Openlvc/Portico.” https://github.com/openlvc/portico/blob/master/codebase/src/java/examples/ieee1516e/foms/RestaurantProcesses.xml.

Pokorny, T. 2016. “Portico.” https://sourceforge.net/projects/portico/?source=recommended.

Pokorny, T., D. Stratton, and P. Smith. 2006. “AOP and the HLA: Simplified Federation Development.” Fall Simulation Interoperability Workshop.

poRTIco. 2009. “Developer Documentation.” http://porticoproject.org/index.php?title=Developer_Documentation.

Portico. 2013. “Portico Space : Www.porticoproject.org.” https://portico.space/youre-the-architect.

Rademakers, T., and J. Dirksen. 2008. “Open-Source ESBs in Action.” Manning Publications. https://www.manning.com/books/open-source-esbs-in-action.

Rasta, K. 2013. “Data Quality-Based Resource Management in Enterprise Service Bus.” University of Agder.

Rolland, C. 1993. “Modeling the Requirements Engineering Process.” 3rd European-Japanese Seminar on Information Modelling and Knowledge Bases.

Rouse, Margaret. 2014. “Operating Systems: Three Easy Pieces.” http://whatis.techtarget.com/definition/operating-system-OS.

Russinovich, Mark E., and David A. Solomon. 2007. “Operating System Principles.” “Windows Internals” paper - 4th edition. http://blogs.msdn.com/b/hanybarakat/archive/2007/02/25/deeper-into-windows-architecture.aspx.

Saha, D., A. Mukherjee, and S. Bandyopadhyay. 2011. “Networking Infrastructure for Pervasive Computing, Enabling Technologies.” Springer Science & Business Media. Springer. Boston, United States of America. Kluwer Academic Publishers.

Salinesi, Camille, and Óscar Pastor. 2011. “Advanced Information Systems Engineering.” CAiSE 2011 International Workshops.

Sethi, A.S., Y. Raynaud, and F. Faure-Vincent. 1995. “Integrated Network Management IV.” Fourth international symposium on integrated network management, Springer US.

Shaughnessy, Haydn. 2011. “The 5 Pillars of the New Enterprise Operating System.” Forbes. https://www.forbes.com/sites/haydnshaughnessy/2011/10/05/the-5-pillars-of-the-new-enterprise-operating-system/#3ce8d0405a39.

Shaun. 2009. “Operating Systems – OS Concepts.” WordPress.com. https://computersciencesource.wordpress.com/2009/11/21/year-2-operating-systems-os-concepts/.

Silberschatz, A., P. Galvin, and G. Gagne. 2009. “Control-Based Operating System Design.” Operating System Concepts – 8th Edition. http://codex.cs.yale.edu/avi/os-book/OS8/os8c/slide-dir/PDF-dir/ch1.pdf.

Silberschatz, Abraham, Greg Gagne, and Peter Baer Galvin. 2006. “HYDRA: The Kernel of a Multiprocessor Operating System.” “Operating System Concepts” paper - Ninth Edition.

The Open Group. 2012. “SOA Reference Architecture Technical Standard: Operational Systems Layer.”

Turner, K.J. 2012. “Advances in Home Care Technologies, Results of the MATCH Project.” IOS Press.

Vallejo, C., D. Romero, and A. Molina. 2012. “Enterprise Integration Engineering Reference Framework and Toolbox.” International Journal of Production Research, Taylor & Francis.

Vernadat, F.B. 1996. “Enterprise Modeling and Integration.”

Vernadat, François. 1996. “Enterprise Modeling and Integration: Principles and Applications.” Chapman & Hall.

Vollmer, K. 2011. “The Forrester Wave™: Enterprise Service Bus for Application Development & Delivery Professionals.” Forrester Research.

Wainer, J., R.S. Filho, and E.R.M. Madeira. 2000. “CORBA Based Architecture for Large Scale Workflow.” Institute of Computing. Tokyo, Japan.

Weatherly, R., A. Wilson, and S. Griffin. 1993. “ALSP-Theory, Experience, and Future Directions.”

Weichhart, G., A. Molina, D. Chen, L.E. Whitman, and F. Vernadat. 2016. “Challenges and Current Developments for Sensing Smart and Sustainable Enterprise Systems.” Computers in Industry.

WfMC. 1999. “The Workflow Management Coalition Specification, Workflow Management Coalition, Terminology & Glossary.” Baidu. Winchester, United Kingdom.

White, S. A. 2013. “Business Process Modeling Notation v1.0.”

Wyszkowski, P. 2011. “ESB Application for Effective Synchronization of Large Volume Measurements Data.” University of Science and Technology. Kraków, Poland. ResearchGate.

Xu, L.D. 2014. “Enterprise Integration and Information Architecture, A Systems Perspective.” Toronto Public Library. Toronto, Canada. Book Depository.

Youssef, J.R., D. Chen, and G. Zacharewicz. 2017. “Developing and Enterprise Operating System (EOS)- State of the Art.” ICE/IEEE’17 - International Conference on Engineering, Technology and Innovation.

Youssef, J.R., D. Chen, G. Zacharewicz, and T. Zhiying. 2016. “Developing and Enterprise Operating System (EOS) with the Federated Interoperability Approach.” I3M’16 - International Multidisciplinary Modeling & Simulation Multiconference.

Youssef, J.R., G. Zacharewicz, and D. Chen. 2016. “Developing and Enterprise Operating System – Requirements and Architectures.” WETICE’16 - 25th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises.

Youssef, J.R., G. Zacharewicz, D. Chen, and François Vernadat. 2017. “EOS: Enterprise Operating Systems.” International Journal of Production Research – 2017.

Youssef, J.R., G. Zacharewicz, D. Chen, and T. Zhiying. 2017. “Enterprise Operating System (EOS) Framework: Federated Interoperability Based on HLA.” International Journal of Simulation and Process Modelling.

Zacharewicz, G., D. Chen, and B. Vallespir. 2008. “HLA Supported Federation Oriented Enterprise Interoperability, Application to Aerospace Enterprises.” International Simulation Multiconference EuroSISO.

Zacharewicz, G, D. Chen, and B. Vallespir. 2009. “Short-Lived Ontology Approach for Agent/HLA Federated Enterprise Interoperability.”

Zaraté, P. 2013. “Tools for Collaborative Decision-Making.” Wiley Online Library. http://onlinelibrary.wiley.com/doi/10.1002/9781118574690.ch2/summary.

Zhiying, T., D. Chen, and G. Zacharewicz. 2012. “Federated Approach for Enterprise Interoperability: A Reversible Model Driven and HLA Based Methodology.”

Zurawski, R. 2004. “The Industrial Information Technology Handbook.” CRC Press Book.

Annexes

1> Banking Sector – Federates Development

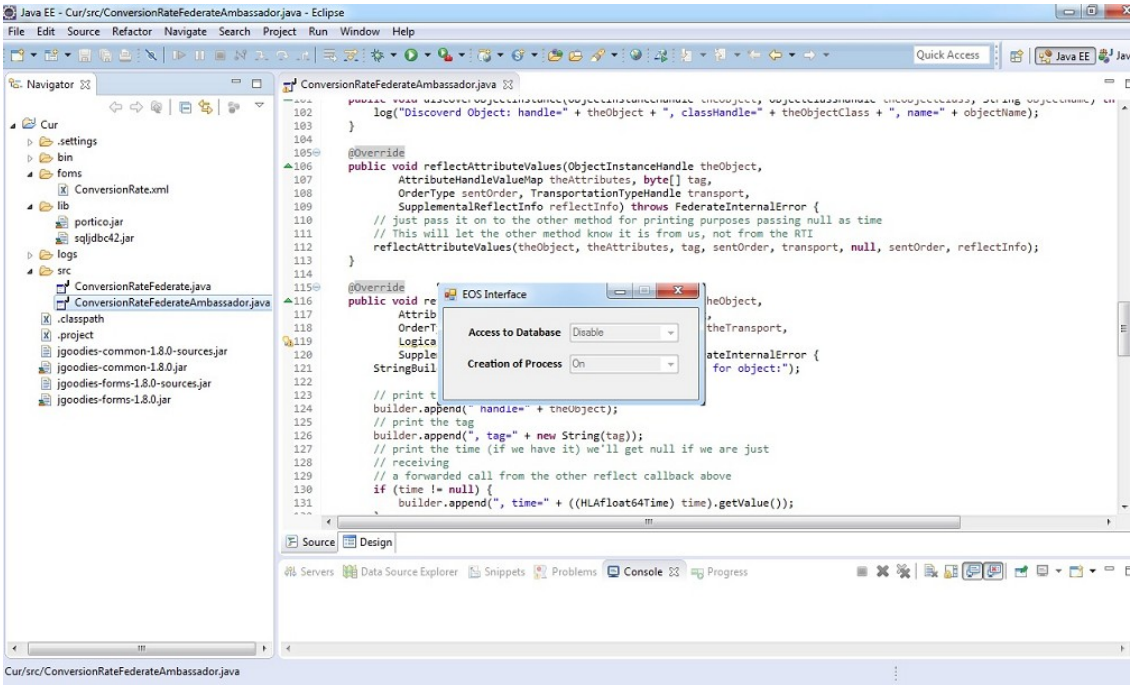

Figure 30: The EOS interface federate integrated with portico

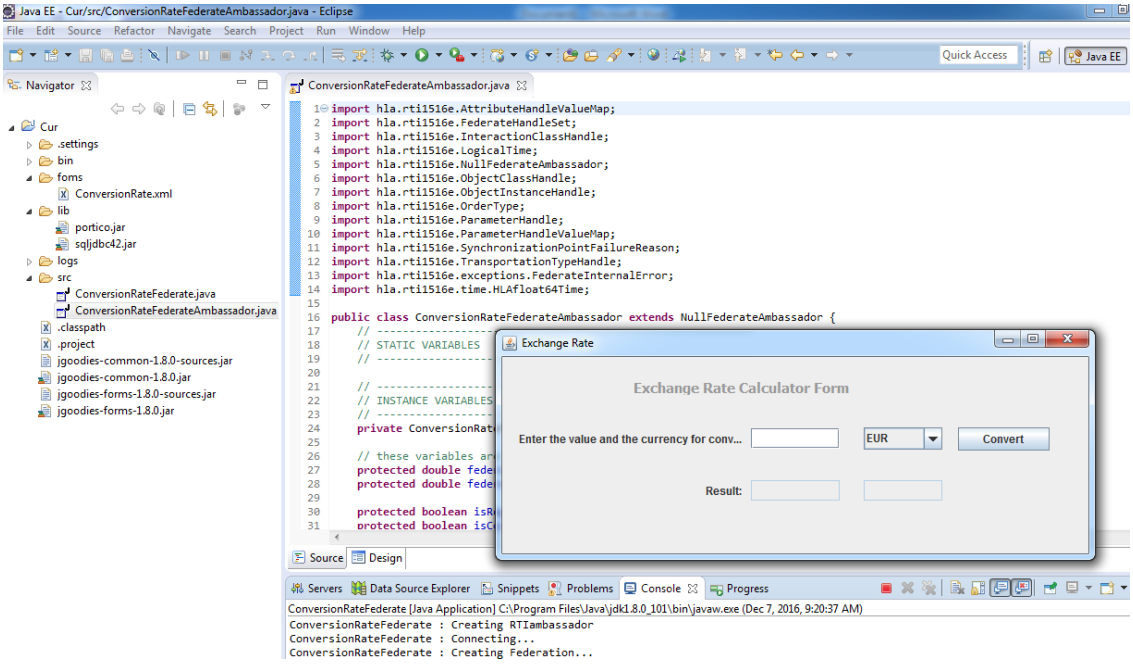

Figure 31: The Enterprise Process Management federate integrated with portico

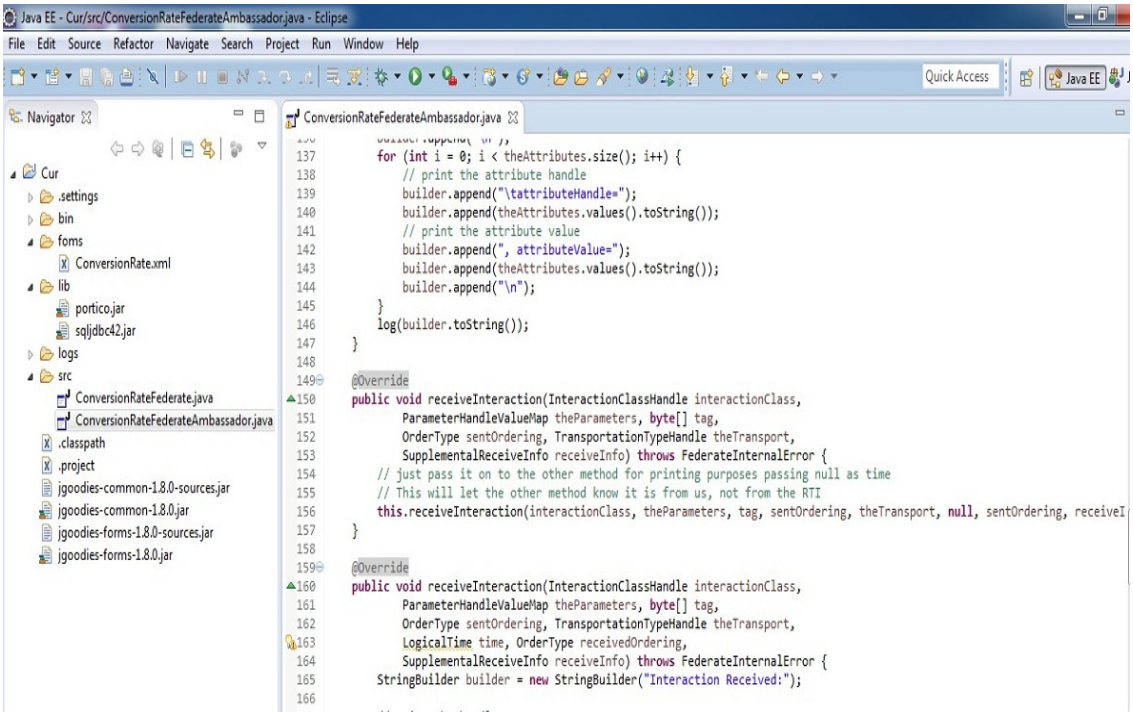

Figure 32: The Enterprise Resource Management federate integrated with poRTIco

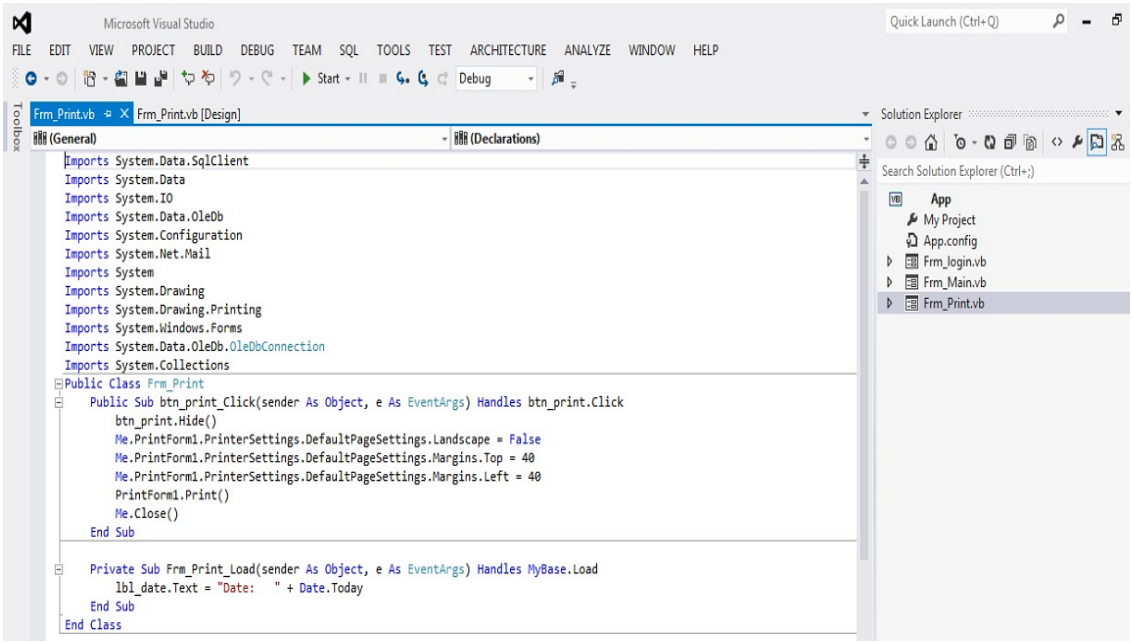

Figure 33: The Presentation Management federate integrated with poRTIco

2> Banking Sector – FOM File and Federates Inter-communications

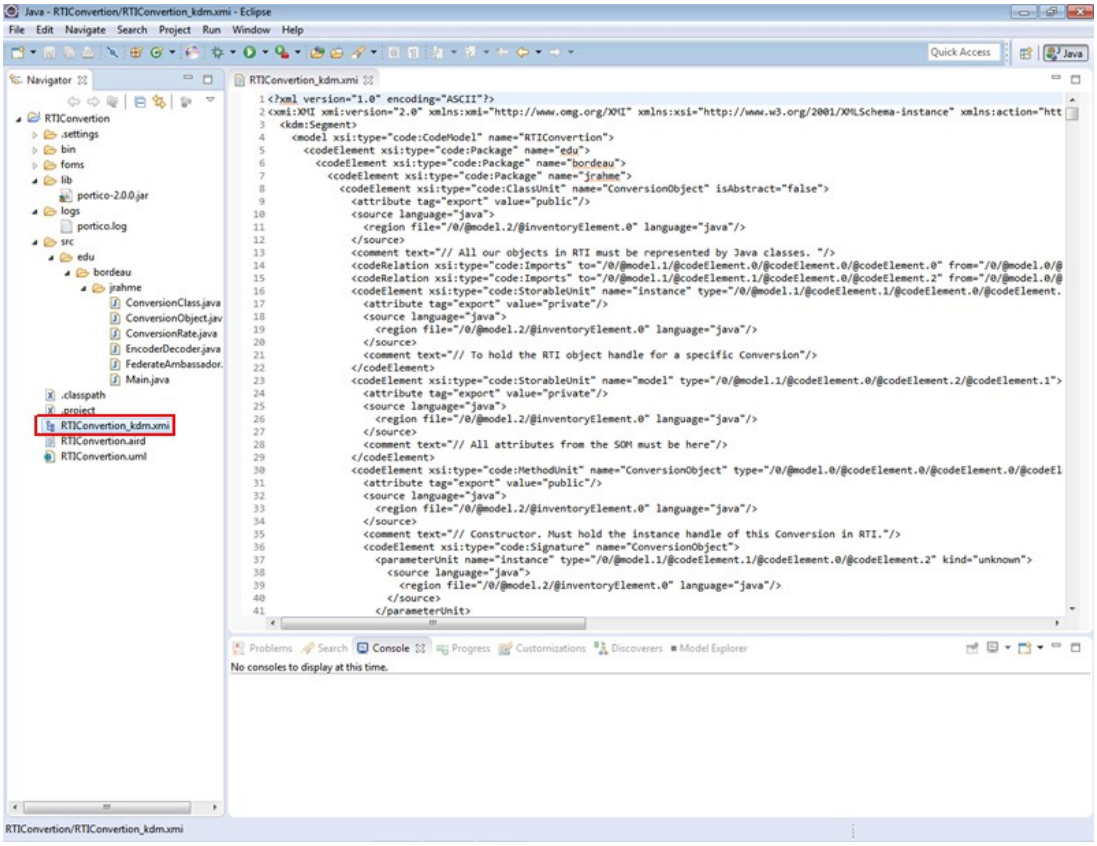

Figure 34: The KDM Code File

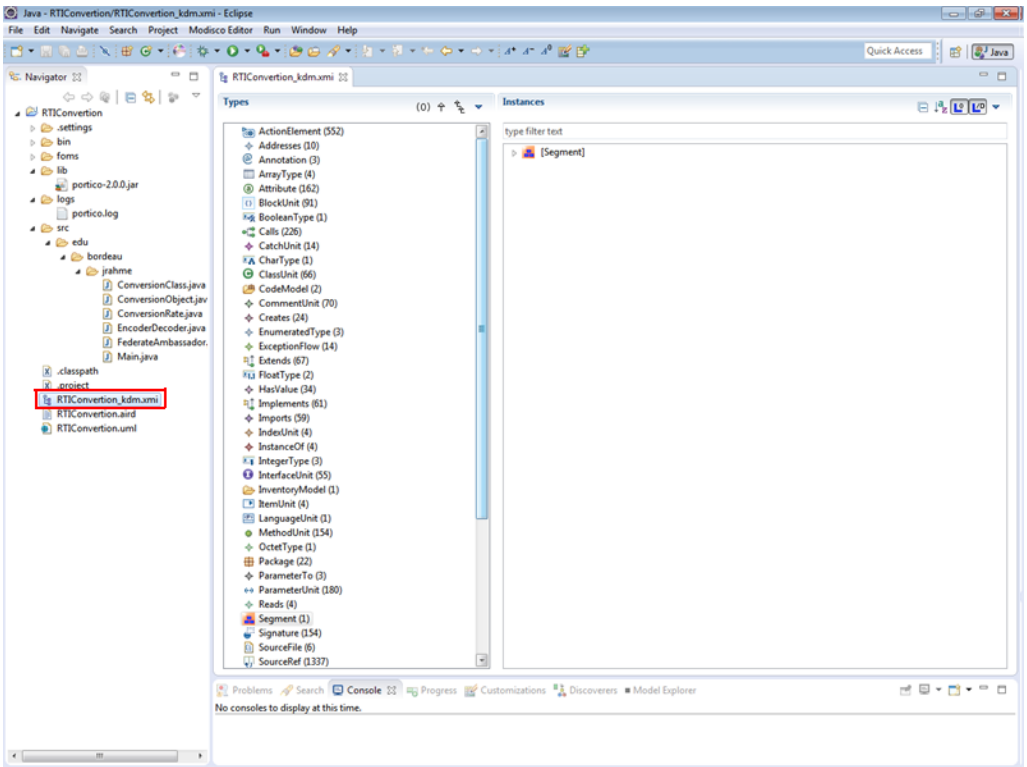

Figure 35: The KDM Model Design

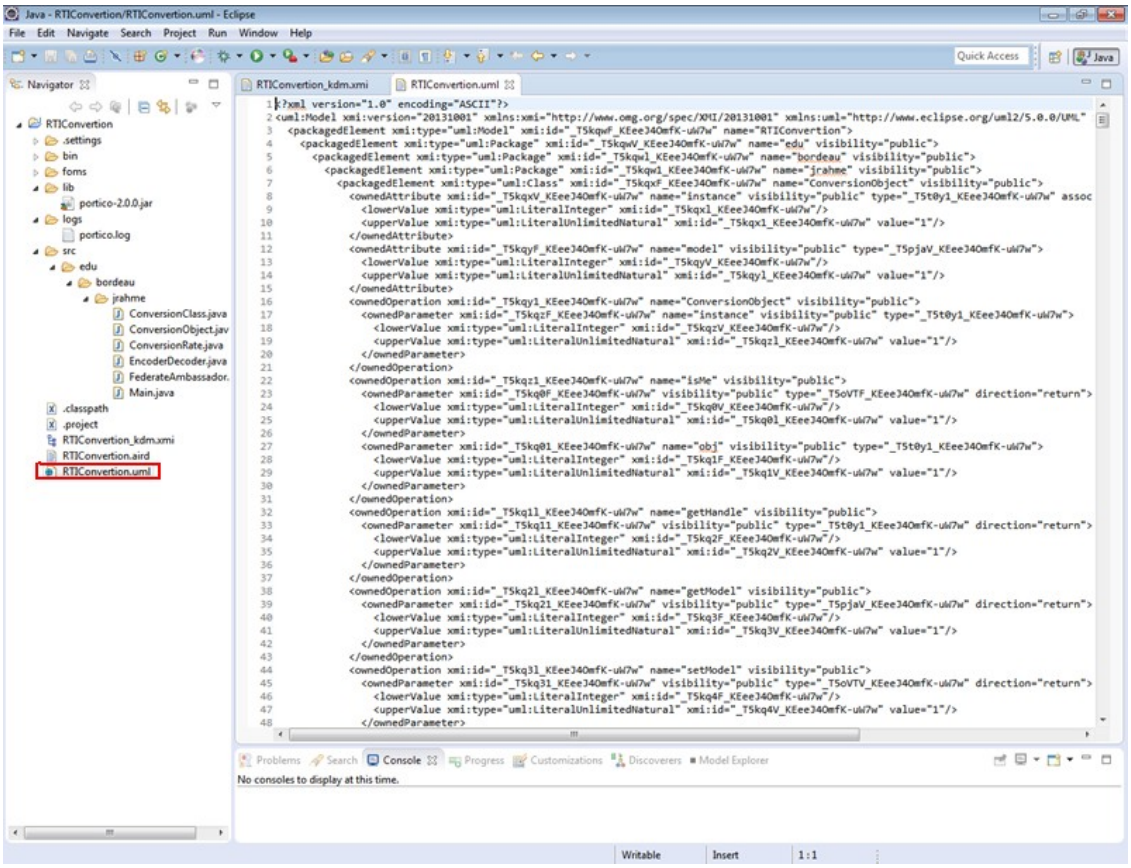

Figure 36: The UML Code File

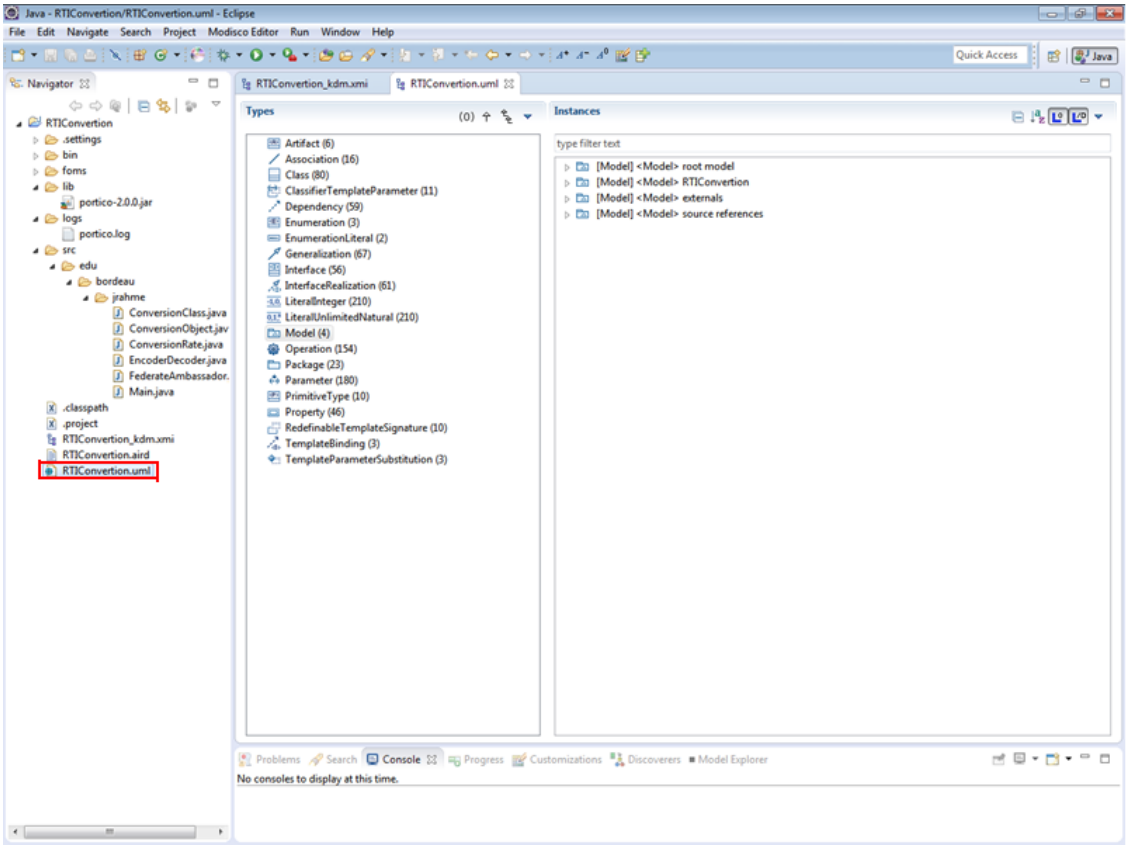

Figure 37: The UML Model Design

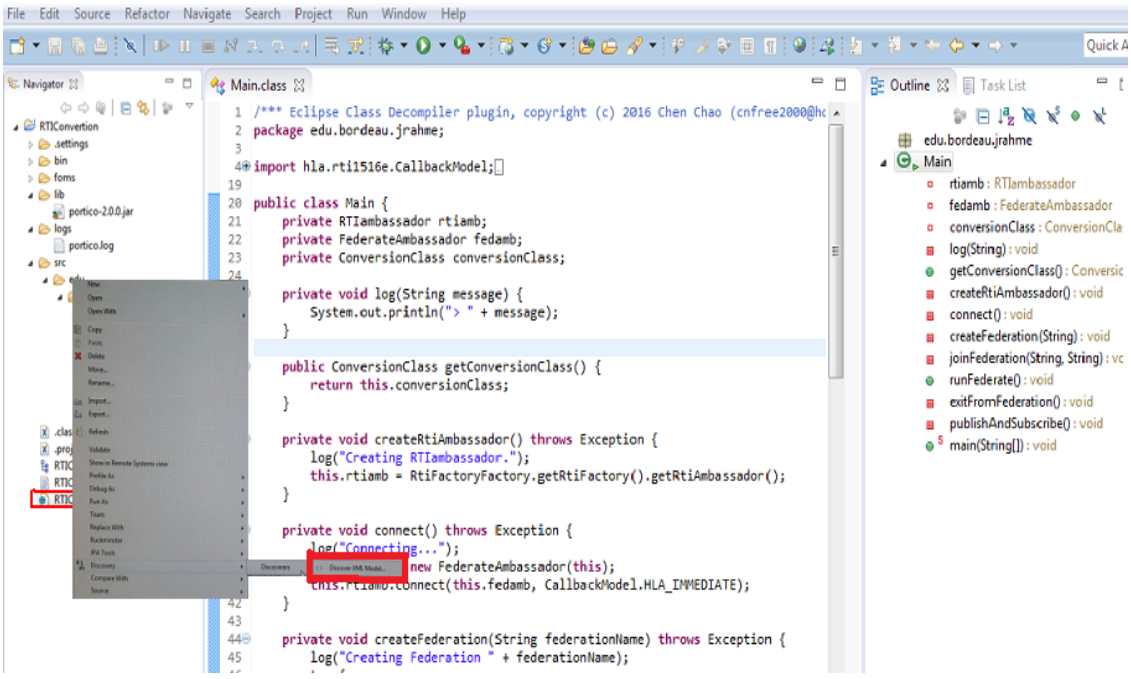

Figure 38: Discovery of XML Model for FOM file

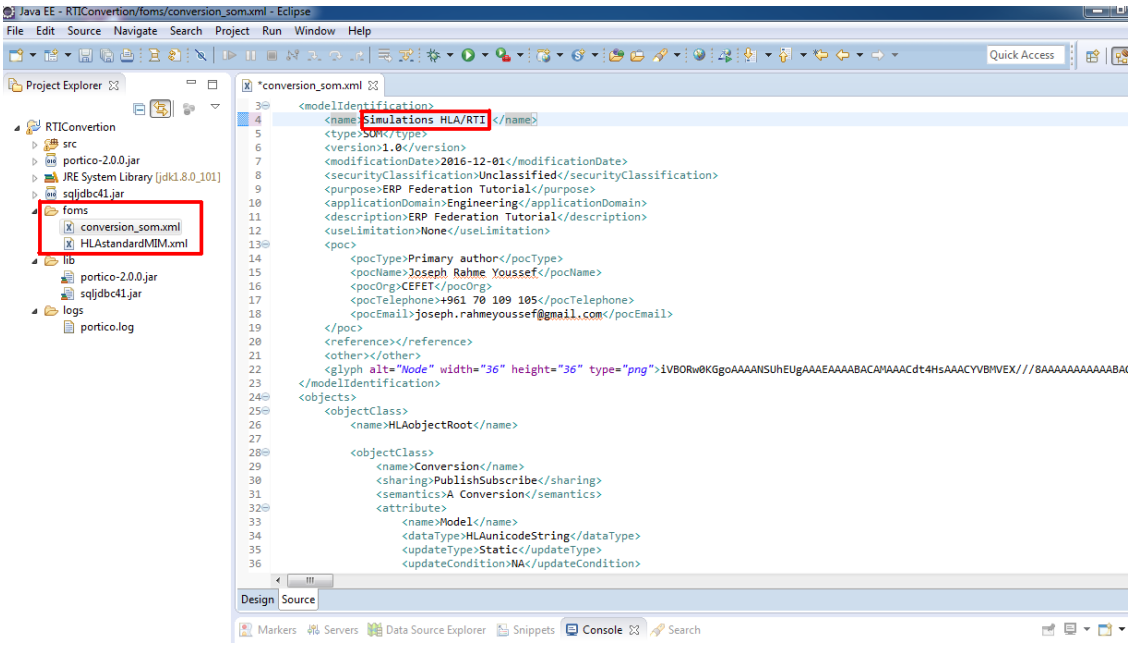

Figure 39: The FOM File generation

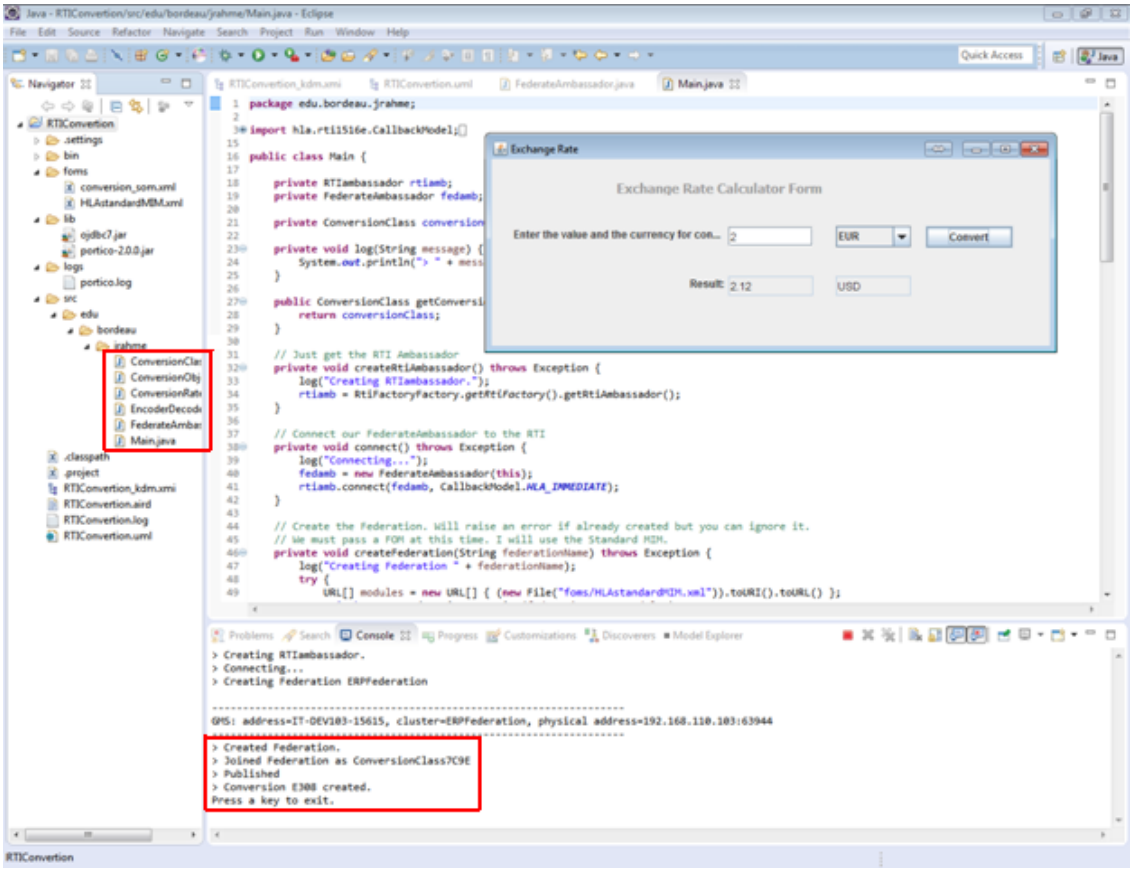

Figure 40: EOS Federates - Inter-Communication results

3> KDM file reversed by MoDisco

- Below is a sample of xml code for KDM file reversed by MoDisco Tool

```
RTIConvertion_kdm.xmi
---------------------
<?xml version="1.0" encoding="ASCII"?>
<kdm:Segment>
    <model xsi:type="code:CodeModel" name="RTIConvertion">
      <codeElement xsi:type="code:Package" name="edu">
        <codeElement xsi:type="code:Package" name="bordeau">
          <codeElement xsi:type="code:Package" name="jrahme">
            <codeElement     xsi:type="code:ClassUnit"     name="ConversionObject"
isAbstract="false">
              <attribute tag="export" value="public"/>
              <source language="java">
                <region file="/0/@model.2/@inventoryElement.0" language="java"/>
              </source>
              <comment text="// All our objects in RTI must be represented by
Java classes. "/>
              <codeRelation                                  xsi:type="code:Imports"
to="/0/@model.1/@codeElement.0/@codeElement.0/@codeElement.0"
from="/0/@model.0/@codeElement.0/@codeElement.0/@codeElement.0/@codeElement.0"/>
              <codeRelation                                  xsi:type="code:Imports"
to="/0/@model.1/@codeElement.1/@codeElement.0/@codeElement.2"
from="/0/@model.0/@codeElement.0/@codeElement.0/@codeElement.0/@codeElement.0"/>
              <codeElement      xsi:type="code:StorableUnit"      name="instance"
type="/0/@model.1/@codeElement.1/@codeElement.0/@codeElement.2">
                <attribute tag="export" value="private"/>
                <source language="java">
                  <region                  file="/0/@model.2/@inventoryElement.0"
language="java"/>
                </source>
                <comment text="// To hold the RTI object handle for a specific
Conversion"/>
              </codeElement>
              <codeElement       xsi:type="code:StorableUnit"        name="model"
type="/0/@model.1/@codeElement.0/@codeElement.2/@codeElement.1">
                <attribute tag="export" value="private"/>
                <source language="java">
                  <region                  file="/0/@model.2/@inventoryElement.0"
language="java"/>
                </source>
                <comment text="// All attributes from the SOM must be here"/>
              </codeElement>
              <codeElement   xsi:type="code:MethodUnit"   name="ConversionObject"
type="/0/@model.0/@codeElement.0/@codeElement.0/@codeElement.0/@codeElement.0/@co
deElement.2/@codeElement.0" kind="constructor">
                <attribute tag="export" value="public"/>
                <source language="java">
                  <region                  file="/0/@model.2/@inventoryElement.0"
language="java"/>
                </source>
                <comment text="// Constructor. Must hold the instance handle of
this Conversion in RTI."/>
                <codeElement xsi:type="code:Signature" name="ConversionObject">
                  <parameterUnit                                    name="instance"
type="/0/@model.1/@codeElement.1/@codeElement.0/@codeElement.2" kind="unknown">
                    <source language="java">
                      <region              file="/0/@model.2/@inventoryElement.0"
language="java"/>
```

```
                    </source>
                  </parameterUnit>
                </codeElement>
                <codeElement xsi:type="action:BlockUnit">
                  <source language="java">
                    <region                file="/0/@model.2/@inventoryElement.0"
language="java"/>
                  </source>
                  <codeElement  xsi:type="action:ActionElement"  name="expression
statement" kind="expression statement">
                    <source language="java">
                      <region              file="/0/@model.2/@inventoryElement.0"
language="java"/>
                    </source>
                    <codeElement   xsi:type="action:ActionElement"   name="ASSIGN"
kind="assignment">
                      <source language="java">
                        <region            file="/0/@model.2/@inventoryElement.0"
language="java"/>
                      </source>
                      <codeElement   xsi:type="action:ActionElement"   name="field
access" kind="field access">
                        <source language="java">
                          <region          file="/0/@model.2/@inventoryElement.0"
language="java"/>
                        </source>
                        <codeElement  xsi:type="action:ActionElement"  name="this"
kind="this">
                          <source language="java">
                            <region        file="/0/@model.2/@inventoryElement.0"
language="java"/>
                          </source>
                        </codeElement>
                        <actionRelation               xsi:type="action:Addresses"
to="/0/@model.0/@codeElement.0/@codeElement.0/@codeElement.0/@codeElement.0/@code
Element.1"
from="/0/@model.0/@codeElement.0/@codeElement.0/@codeElement.0/@codeElement.0/@co
deElement.2/@codeElement.1/@codeElement.0/@codeElement.0/@codeElement.0"/>
                      </codeElement>
                      <codeElement  xsi:type="action:ActionElement"  name="method
invocation" kind="method invocation">
                        <source language="java">
                          <region          file="/0/@model.2/@inventoryElement.0"
language="java"/>
                        </source>
                        <codeElement xsi:type="action:ActionElement" name="method
invocation" kind="method invocation">
                          <source language="java">
                            <region        file="/0/@model.2/@inventoryElement.0"
language="java"/>
                          </source>
                          <codeElement            xsi:type="action:ActionElement"
name="method invocation" kind="method invocation">
                            <source language="java">
                              <region      file="/0/@model.2/@inventoryElement.0"
language="java"/>
                            </source>
```

```
                          <codeElement          xsi:type="action:ActionElement"
name="method invocation" kind="method invocation">
                            <source language="java">
                              <region     file="/0/@model.2/@inventoryElement.0"
language="java"/>
                            </source>
                            <actionRelation             xsi:type="action:Calls"
to="/0/@model.1/@codeElement.0/@codeElement.0/@codeElement.0/@codeElement.0"
from="/0/@model.0/@codeElement.0/@codeElement.0/@codeElement.0/@codeElement.0/@co
deElement.2/@codeElement.1/@codeElement.0/@codeElement.0/@codeElement.1/@codeElem
ent.0/@codeElement.0/@codeElement.0"/>
                          </codeElement>
                          <actionRelation               xsi:type="action:Calls"
to="/0/@model.1/@codeElement.0/@codeElement.0/@codeElement.0/@codeElement.1"
from="/0/@model.0/@codeElement.0/@codeElement.0/@codeElement.0/@codeElement.0/@co
deElement.2/@codeElement.1/@codeElement.0/@codeElement.0/@codeElement.1/@codeElem
ent.0/@codeElement.0"/>
                        </codeElement>
                        <codeElement     xsi:type="code:Value"     name="number
literal" type="/0/@model.0/@codeElement.1/@codeElement.0" ext="1">
                          <source language="java">
                            <region      file="/0/@model.2/@inventoryElement.0"
language="java"/>
                          </source>
                        </codeElement>
                        <codeElement     xsi:type="code:Value"     name="number
literal" type="/0/@model.0/@codeElement.1/@codeElement.0" ext="5">
                          <source language="java">
                            <region      file="/0/@model.2/@inventoryElement.0"
language="java"/>
                          </source>
                        </codeElement>
                        <actionRelation                 xsi:type="action:Calls"
to="/0/@model.1/@codeElement.0/@codeElement.2/@codeElement.1/@codeElement.0"
from="/0/@model.0/@codeElement.0/@codeElement.0/@codeElement.0/@codeElement.0/@co
deElement.2/@codeElement.1/@codeElement.0/@codeElement.0/@codeElement.1/@codeElem
ent.0"/>
                      </codeElement>
                      <actionRelation                   xsi:type="action:Calls"
to="/0/@model.1/@codeElement.0/@codeElement.2/@codeElement.1/@codeElement.1"
from="/0/@model.0/@codeElement.0/@codeElement.0/@codeElement.0/@codeElement.0/@co
deElement.2/@codeElement.1/@codeElement.0/@codeElement.0/@codeElement.1"/>
                    </codeElement>
                  </codeElement>
                </codeElement>
                <codeElement  xsi:type="action:ActionElement"  name="expression
statement" kind="expression statement">
                  <source language="java">
                    <region              file="/0/@model.2/@inventoryElement.0"
language="java"/>
                  </source>
```

4> UML file reversed by MoDisco

- Below is a sample of xml code for UML file reversed by MoDisco Tool

```
<?xml version="1.0" encoding="ASCII"?>
<uml:Model xmi:version="20131001"
<packagedElement       xmi:type="uml:Model"       xmi:id="_T5kqwF_KEeeJ4OmfK-uW7w"
name="RTIConvertion">
    <packagedElement    xmi:type="uml:Package"    xmi:id="_T5kqwV_KEeeJ4OmfK-uW7w"
name="edu" visibility="public">
      <packagedElement   xmi:type="uml:Package"   xmi:id="_T5kqwl_KEeeJ4OmfK-uW7w"
name="bordeau" visibility="public">
        <packagedElement  xmi:type="uml:Package"  xmi:id="_T5kqw1_KEeeJ4OmfK-uW7w"
name="jrahme" visibility="public">
          <packagedElement  xmi:type="uml:Class"  xmi:id="_T5kqxF_KEeeJ4OmfK-uW7w"
name="ConversionObject" visibility="public">
            <ownedAttribute    xmi:id="_T5kqxV_KEeeJ4OmfK-uW7w"    name="instance"
visibility="public"                                  type="_T5t0y1_KEeeJ4OmfK-uW7w"
association="_T5oVJ1_KEeeJ4OmfK-uW7w">
              <lowerValue                          xmi:type="uml:LiteralInteger"
xmi:id="_T5kqxl_KEeeJ4OmfK-uW7w"/>
              <upperValue                  xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5kqx1_KEeeJ4OmfK-uW7w" value="1"/>
            </ownedAttribute>
            <ownedAttribute     xmi:id="_T5kqyF_KEeeJ4OmfK-uW7w"     name="model"
visibility="public" type="_T5pjaV_KEeeJ4OmfK-uW7w">
              <lowerValue                          xmi:type="uml:LiteralInteger"
xmi:id="_T5kqyV_KEeeJ4OmfK-uW7w"/>
              <upperValue                  xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5kqyl_KEeeJ4OmfK-uW7w" value="1"/>
            </ownedAttribute>
            <ownedOperation                       xmi:id="_T5kqy1_KEeeJ4OmfK-uW7w"
name="ConversionObject" visibility="public">
              <ownedParameter   xmi:id="_T5kqzF_KEeeJ4OmfK-uW7w"   name="instance"
visibility="public" type="_T5t0y1_KEeeJ4OmfK-uW7w">
                <lowerValue                        xmi:type="uml:LiteralInteger"
xmi:id="_T5kqzV_KEeeJ4OmfK-uW7w"/>
                <upperValue                xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5kqzl_KEeeJ4OmfK-uW7w" value="1"/>
              </ownedParameter>
            </ownedOperation>
            <ownedOperation      xmi:id="_T5kqz1_KEeeJ4OmfK-uW7w"      name="isMe"
visibility="public">
              <ownedParameter                     xmi:id="_T5kq0F_KEeeJ4OmfK-uW7w"
visibility="public" type="_T5oVTF_KEeeJ4OmfK-uW7w" direction="return">
                <lowerValue                        xmi:type="uml:LiteralInteger"
xmi:id="_T5kq0V_KEeeJ4OmfK-uW7w"/>
                <upperValue                xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5kq0l_KEeeJ4OmfK-uW7w" value="1"/>
              </ownedParameter>
              <ownedParameter     xmi:id="_T5kq01_KEeeJ4OmfK-uW7w"     name="obj"
visibility="public" type="_T5t0y1_KEeeJ4OmfK-uW7w">
                <lowerValue                        xmi:type="uml:LiteralInteger"
xmi:id="_T5kq1F_KEeeJ4OmfK-uW7w"/>
                <upperValue                xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5kq1V_KEeeJ4OmfK-uW7w" value="1"/>
              </ownedParameter>
            </ownedOperation>
            <ownedOperation   xmi:id="_T5kq1l_KEeeJ4OmfK-uW7w"   name="getHandle"
visibility="public">
              <ownedParameter                     xmi:id="_T5kq11_KEeeJ4OmfK-uW7w"
```

```
visibility="public" type="_T5t0y1_KEeeJ4OmfK-uW7w" direction="return">
                <lowerValue                          xmi:type="uml:LiteralInteger"
xmi:id="_T5kq2F_KEeeJ4OmfK-uW7w"/>
                <upperValue                xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5kq2V_KEeeJ4OmfK-uW7w" value="1"/>
              </ownedParameter>
            </ownedOperation>
            <ownedOperation    xmi:id="_T5kq2l_KEeeJ4OmfK-uW7w"    name="getModel"
visibility="public">
              <ownedParameter                    xmi:id="_T5kq21_KEeeJ4OmfK-uW7w"
visibility="public" type="_T5pjaV_KEeeJ4OmfK-uW7w" direction="return">
                <lowerValue                          xmi:type="uml:LiteralInteger"
xmi:id="_T5kq3F_KEeeJ4OmfK-uW7w"/>
                <upperValue                xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5kq3V_KEeeJ4OmfK-uW7w" value="1"/>
              </ownedParameter>
            </ownedOperation>
            <ownedOperation    xmi:id="_T5kq3l_KEeeJ4OmfK-uW7w"    name="setModel"
visibility="public">
              <ownedParameter                    xmi:id="_T5kq31_KEeeJ4OmfK-uW7w"
visibility="public" type="_T5oVTV_KEeeJ4OmfK-uW7w" direction="return">
                <lowerValue                          xmi:type="uml:LiteralInteger"
xmi:id="_T5kq4F_KEeeJ4OmfK-uW7w"/>
                <upperValue                xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5kq4V_KEeeJ4OmfK-uW7w" value="1"/>
              </ownedParameter>
              <ownedParameter    xmi:id="_T5kq4l_KEeeJ4OmfK-uW7w"    name="model"
visibility="public" type="_T5pjaV_KEeeJ4OmfK-uW7w">
                <lowerValue                          xmi:type="uml:LiteralInteger"
xmi:id="_T5kq41_KEeeJ4OmfK-uW7w"/>
                <upperValue                xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5kq5F_KEeeJ4OmfK-uW7w" value="1"/>
              </ownedParameter>
            </ownedOperation>
          </packagedElement>
          <packagedElement  xmi:type="uml:Class"  xmi:id="_T5kq5V_KEeeJ4OmfK-uW7w"
name="FederateAmbassador" visibility="public">
            <generalization                      xmi:id="_T5kq5l_KEeeJ4OmfK-uW7w"
general="_T5t01V_KEeeJ4OmfK-uW7w"/>
            <ownedAttribute    xmi:id="_T5lR0F_KEeeJ4OmfK-uW7w"    name="federate"
visibility="public"                                  type="_T5nHH1_KEeeJ4OmfK-uW7w"
association="_T5oVKV_KEeeJ4OmfK-uW7w">
              <lowerValue                          xmi:type="uml:LiteralInteger"
xmi:id="_T5lR0V_KEeeJ4OmfK-uW7w"/>
              <upperValue                xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5lR0l_KEeeJ4OmfK-uW7w" value="1"/>
            </ownedAttribute>
            <ownedOperation                      xmi:id="_T5lR01_KEeeJ4OmfK-uW7w"
name="FederateAmbassador" visibility="public">
              <ownedParameter   xmi:id="_T5lR1F_KEeeJ4OmfK-uW7w"   name="federate"
visibility="public" type="_T5nHH1_KEeeJ4OmfK-uW7w">
                <lowerValue                          xmi:type="uml:LiteralInteger"
xmi:id="_T5lR1V_KEeeJ4OmfK-uW7w"/>
                <upperValue                xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5lR1l_KEeeJ4OmfK-uW7w" value="1"/>
              </ownedParameter>
            </ownedOperation>
```

```
            <ownedOperation      xmi:id="_T5lR11_KEeeJ4OmfK-uW7w"      name="log"
visibility="private">
              <ownedParameter                    xmi:id="_T5lR2F_KEeeJ4OmfK-uW7w"
visibility="public" type="_T5oVTV_KEeeJ4OmfK-uW7w" direction="return">
                <lowerValue                          xmi:type="uml:LiteralInteger"
xmi:id="_T5lR2V_KEeeJ4OmfK-uW7w"/>
                <upperValue                xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5lR2l_KEeeJ4OmfK-uW7w" value="1"/>
              </ownedParameter>
              <ownedParameter   xmi:id="_T5lR21_KEeeJ4OmfK-uW7w"   name="message"
visibility="public" type="_T5pjaV_KEeeJ4OmfK-uW7w">
                <lowerValue                          xmi:type="uml:LiteralInteger"
xmi:id="_T5lR3F_KEeeJ4OmfK-uW7w"/>
                <upperValue                xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5lR3V_KEeeJ4OmfK-uW7w" value="1"/>
              </ownedParameter>
            </ownedOperation>
            <ownedOperation                      xmi:id="_T5lR3l_KEeeJ4OmfK-uW7w"
name="discoverObjectInstance" visibility="public">
              <ownedParameter                    xmi:id="_T5lR31_KEeeJ4OmfK-uW7w"
visibility="public" type="_T5oVTV_KEeeJ4OmfK-uW7w" direction="return">
                <lowerValue                          xmi:type="uml:LiteralInteger"
xmi:id="_T5lR4F_KEeeJ4OmfK-uW7w"/>
                <upperValue                xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5lR4V_KEeeJ4OmfK-uW7w" value="1"/>
              </ownedParameter>
              <ownedParameter  xmi:id="_T5lR4l_KEeeJ4OmfK-uW7w"  name="theObject"
visibility="public" type="_T5t0y1_KEeeJ4OmfK-uW7w">
                <lowerValue                          xmi:type="uml:LiteralInteger"
xmi:id="_T5lR41_KEeeJ4OmfK-uW7w"/>
                <upperValue                xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5lR5F_KEeeJ4OmfK-uW7w" value="1"/>
              </ownedParameter>
              <ownedParameter                    xmi:id="_T5lR5V_KEeeJ4OmfK-uW7w"
name="theObjectClass" visibility="public" type="_T5t02V_KEeeJ4OmfK-uW7w">
                <lowerValue                          xmi:type="uml:LiteralInteger"
xmi:id="_T5lR5l_KEeeJ4OmfK-uW7w"/>
                <upperValue                xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5lR51_KEeeJ4OmfK-uW7w" value="1"/>
              </ownedParameter>
              <ownedParameter  xmi:id="_T5lR6F_KEeeJ4OmfK-uW7w"  name="objectName"
visibility="public" type="_T5pjaV_KEeeJ4OmfK-uW7w">
                <lowerValue                          xmi:type="uml:LiteralInteger"
xmi:id="_T5lR6V_KEeeJ4OmfK-uW7w"/>
                <upperValue                xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5lR6l_KEeeJ4OmfK-uW7w" value="1"/>
              </ownedParameter>
              <ownedParameter                    xmi:id="_T5lR61_KEeeJ4OmfK-uW7w"
visibility="public" type="_T5smll_KEeeJ4OmfK-uW7w">
                <lowerValue                          xmi:type="uml:LiteralInteger"
xmi:id="_T5lR7F_KEeeJ4OmfK-uW7w"/>
                <upperValue                xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5lR7V_KEeeJ4OmfK-uW7w" value="1"/>
              </ownedParameter>
            </ownedOperation>
            <ownedOperation                      xmi:id="_T5lR7l_KEeeJ4OmfK-uW7w"
name="reflectAttributeValues" visibility="public">
```

```
            <ownedParameter                    xmi:id="_T5lR71_KEeeJ4OmfK-uW7w"
visibility="public" type="_T5oVTV_KEeeJ4OmfK-uW7w" direction="return">
              <lowerValue                          xmi:type="uml:LiteralInteger"
xmi:id="_T5lR8F_KEeeJ4OmfK-uW7w"/>
              <upperValue               xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5lR8V_KEeeJ4OmfK-uW7w" value="1"/>
            </ownedParameter>
            <ownedParameter  xmi:id="_T5lR8l_KEeeJ4OmfK-uW7w"  name="theObject"
visibility="public" type="_T5t0y1_KEeeJ4OmfK-uW7w">
              <lowerValue                          xmi:type="uml:LiteralInteger"
xmi:id="_T5lR81_KEeeJ4OmfK-uW7w"/>
              <upperValue               xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5lR9F_KEeeJ4OmfK-uW7w" value="1"/>
            </ownedParameter>
            <ownedParameter                    xmi:id="_T5lR9V_KEeeJ4OmfK-uW7w"
name="theAttributes" visibility="public" type="_T5t00V_KEeeJ4OmfK-uW7w">
              <lowerValue                          xmi:type="uml:LiteralInteger"
xmi:id="_T5lR9l_KEeeJ4OmfK-uW7w"/>
              <upperValue               xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5lR91_KEeeJ4OmfK-uW7w" value="1"/>
            </ownedParameter>
            <ownedParameter     xmi:id="_T5lR-F_KEeeJ4OmfK-uW7w"     name="tag"
visibility="public" type="_T5oVUF_KEeeJ4OmfK-uW7w">
              <lowerValue      xmi:type="uml:LiteralInteger"      xmi:id="_T5lR-
V_KEeeJ4OmfK-uW7w"/>
              <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_T5lR-
l_KEeeJ4OmfK-uW7w" value="*"/>
            </ownedParameter>
            <ownedParameter  xmi:id="_T5lR-1_KEeeJ4OmfK-uW7w"  name="sentOrder"
visibility="public" type="_T5ubwV_KEeeJ4OmfK-uW7w">
              <lowerValue                          xmi:type="uml:LiteralInteger"
xmi:id="_T5lR_F_KEeeJ4OmfK-uW7w"/>
              <upperValue               xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5lR_V_KEeeJ4OmfK-uW7w" value="1"/>
            </ownedParameter>
            <ownedParameter  xmi:id="_T5lR_l_KEeeJ4OmfK-uW7w"  name="transport"
visibility="public" type="_T5ubwl_KEeeJ4OmfK-uW7w">
              <lowerValue                          xmi:type="uml:LiteralInteger"
xmi:id="_T5lR_1_KEeeJ4OmfK-uW7w"/>
              <upperValue               xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5lSAF_KEeeJ4OmfK-uW7w" value="1"/>
            </ownedParameter>
            <ownedParameter xmi:id="_T5lSAV_KEeeJ4OmfK-uW7w" name="reflectInfo"
visibility="public" type="_T5t02F_KEeeJ4OmfK-uW7w">
              <lowerValue                          xmi:type="uml:LiteralInteger"
xmi:id="_T5lSAl_KEeeJ4OmfK-uW7w"/>
              <upperValue               xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5lSA1_KEeeJ4OmfK-uW7w" value="1"/>
            </ownedParameter>
            <ownedParameter                    xmi:id="_T5lSBF_KEeeJ4OmfK-uW7w"
visibility="public" type="_T5smll_KEeeJ4OmfK-uW7w">
              <lowerValue                          xmi:type="uml:LiteralInteger"
xmi:id="_T5lSBV_KEeeJ4OmfK-uW7w"/>
              <upperValue               xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_T5lSBl_KEeeJ4OmfK-uW7w" value="1"/>
            </ownedParameter>
          </ownedOperation>
```

```
            </packagedElement>
            <packagedElement  xmi:type="uml:Class"  xmi:id="_T5lSB1_KEeeJ4OmfK-uW7w"
    name="EncoderDecoder" visibility="public">
              <ownedAttribute                        xmi:id="_T5lSCF_KEeeJ4OmfK-uW7w"
    name="encoderFactory"      visibility="public"      type="_T5tNo1_KEeeJ4OmfK-uW7w"
    association="_T5oVK1_KEeeJ4OmfK-uW7w">
                <lowerValue                            xmi:type="uml:LiteralInteger"
    xmi:id="_T5lSCV_KEeeJ4OmfK-uW7w"/>
                <upperValue                  xmi:type="uml:LiteralUnlimitedNatural"
    xmi:id="_T5lSCl_KEeeJ4OmfK-uW7w" value="1"/>
              </ownedAttribute>
              <ownedOperation                        xmi:id="_T5lSC1_KEeeJ4OmfK-uW7w"
    name="createHLAunicodeString" visibility="public">
                <ownedParameter                      xmi:id="_T5lSDF_KEeeJ4OmfK-uW7w"
    visibility="public" type="_T5t0yF_KEeeJ4OmfK-uW7w" direction="return">
                  <lowerValue                          xmi:type="uml:LiteralInteger"
    xmi:id="_T5lSDV_KEeeJ4OmfK-uW7w"/>
                  <upperValue                xmi:type="uml:LiteralUnlimitedNatural"
    xmi:id="_T5lSDl_KEeeJ4OmfK-uW7w" value="1"/>
                </ownedParameter>
                <ownedParameter    xmi:id="_T5lSD1_KEeeJ4OmfK-uW7w"    name="value"
    visibility="public" type="_T5pjaV_KEeeJ4OmfK-uW7w">
                  <lowerValue                          xmi:type="uml:LiteralInteger"
    xmi:id="_T5lSEF_KEeeJ4OmfK-uW7w"/>
                  <upperValue                xmi:type="uml:LiteralUnlimitedNatural"
    xmi:id="_T5lSEV_KEeeJ4OmfK-uW7w" value="1"/>
                </ownedParameter>
              </ownedOperation>
              <ownedOperation                        xmi:id="_T5lSEl_KEeeJ4OmfK-uW7w"
    name="createHLAboolean" visibility="public">
```

5> FOM generation

- Below is a list of java code for FOM generation and operation:

```
package objectanalysis;

import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;

/**
 * this class is used for model transformation
 * @author Joseph Youssef
 */
public class WriteToFile {

    public static boolean generateFOM (HashMap<String, ArrayList<String>> fomMap) {
        try {
            //Convenience class for writing character files.
            boolean:true,append
            FileWriter fw = new FileWriter("testFOM.fed",false);
            //Print formatted representations of objects to a text-output stream.
```

```
            PrintWriter out = new PrintWriter(fw);

            //load poRTIco FOM template
            String topMsg = loadTemplate();

            //Create new FOM
            out.write(topMsg);

            Iterator iterator = fomMap.keySet().iterator();
            while(iterator.hasNext()){
                String className = iterator.next().toString();

                //Find UML:class
                ArrayList attrList = fomMap.get(className);
                if (attrList.size() == 0) {
                    //Transform to RTIObjects:class
                    out.println("\n\t\t\t(class " + className + ")");
                }
               else {
                    //Transform to RTIObjects:class
                    out.println("\n\t\t\t(class " + className);
                    for(int i = 0; i < attrList.size(); i++){
                        //Find    UML:Attributes,    and    transform    them    into
                      RTIObjects:Attributes
                   out.println("\n\t\t\t\t(attribute " + attrList.get(i) + " reliable
               timestamp TestSpace)");
                    }
                    out.println("\n\t\t\t)");
                }
            }
            out.println("\n\t\t)");
            out.println("\n\t)");
            out.println("\n)");

            out.close();
            fw.close();
            return true;
        } catch (IOException e) {
            System.out.println("Write file error!");
            e.printStackTrace();
            return false;
        }
    }

    private static String loadTemplate() {
        String   template   =   "(FED\n\t(Federation   Portico-Test)\n\t\t(FEDversion
v1.3)\n\t\t(spaces" +
                "\n\t\t\t(space                          TestSpace\n\t\t\t\t(dimension
TestDimension)\n\t\t\t)\n\t\t(space OtherSpace\n\t\t\t" +
                    "(dimension
OtherDimension)\n\t\t\t)\n\t\t)\n\t(objects\n\t\t(class ObjectRoot\n\t\t\t"+
                    "(attribute  privilegeToDelete  reliable  timestamp)\n\t\t\t(class
RTIprivate)";
                return template;
    }
}
```

- Below is an xml template code for FOM generation and operation (Pokorny 2016):

```
<objectModel>
     <type>FOM</type>
      <securityClassification>Unclassified</securityClassification>
     <applicationDomain>HLA General</applicationDomain>
   </modelIdentification>
   <objects>
      <objectClass>
<name>HLAobjectRoot</name>
         <sharing>Neither</sharing>
         <attribute>
            <name>HLAprivilegeToDeleteObject</name>
            <dataType>HLAtoken</dataType>
            <updateType>Static</updateType>
            <updateCondition>NA</updateCondition>
            <ownership>DivestAcquire</ownership>
            <sharing>PublishSubscribe</sharing>
            <transportation>HLAreliable</transportation>
            <order>TimeStamp</order>
         </attribute>
         <objectClass>
            <name>HLAmanager</name>
            <sharing>Neither</sharing>
            <semantics>This object class is the root class of all MOM object
               classes</semantics>
            <objectClass>
               <name>HLAfederate</name>
               <sharing>Publish</sharing>
               <attribute>
                  <name>HLAfederateHandle</name>
                  <dataType>HLAhandle</dataType>
                  <updateType>Static</updateType>
                  <updateCondition>NA</updateCondition>
                  <ownership>NoTransfer</ownership>
                  <sharing>Publish</sharing>
                  <dimensions>
                     <dimension>HLAfederate</dimension>
                  </dimensions>
                  <transportation>HLAreliable</transportation>
                  <order>Receive</order>
                  <semantics>Handle of the joined federate returned by a Join
                Federation Execution service invocation
                  </semantics>
               </attribute>
               <attribute>
                  <name>HLAfederateName</name>
                  <dataType>HLAunicodeString</dataType>
                  <updateType>Static</updateType>
                  <updateCondition>NA</updateCondition>
                  <ownership>NoTransfer</ownership>
                  <sharing>Publish</sharing>
                  <dimensions>
                     <dimension>HLAfederate</dimension>
                  </dimensions>
```

```
   <transportation>HLAreliable</transportation>
   <order>Receive</order>
   <semantics>
      Name of the joined federate supplied to a successful Join
Federation Execution service invocation
   </semantics>
</attribute>
<attribute>
   <name>HLAfederateType</name>
   <dataType>HLAunicodeString</dataType>
   <updateType>Static</updateType>
   <updateCondition>NA</updateCondition>
   <ownership>NoTransfer</ownership>
   <sharing>Publish</sharing>
   <dimensions>
      <dimension>HLAfederate</dimension>
   </dimensions>
   <transportation>HLAreliable</transportation>
   <order>Receive</order>
   <semantics>Type of the joined federate specified by the joined
federate when it joined the federation
   </semantics>
</attribute>
<attribute>
   <name>HLAfederateHost</name>
   <dataType>HLAunicodeString</dataType>
   <updateType>Static</updateType>
   <updateCondition>NA</updateCondition>
   <ownership>NoTransfer</ownership>
   <sharing>Publish</sharing>
   <dimensions>
      <dimension>HLAfederate</dimension>
   </dimensions>
   <transportation>HLAreliable</transportation>
   <order>Receive</order>
   <semantics>Host name of the computer on which the joined federate
is executing</semantics>
</attribute>
<attribute>
   <name>HLARTIversion</name>
   <dataType>HLAunicodeString</dataType>
   <updateType>Static</updateType>
   <updateCondition>NA</updateCondition>
   <ownership>NoTransfer</ownership>
   <sharing>Publish</sharing>
   <dimensions>
      <dimension>HLAfederate</dimension>
   </dimensions>
   <transportation>HLAreliable</transportation>
   <order>Receive</order>
   <semantics>Version of the RTI software being used</semantics>
</attribute>
<attribute>
   <name>HLAFOMmoduleDesignatorList</name>
   <dataType>HLAmoduleDesignatorList</dataType>
   <updateType>Static</updateType>
   <updateCondition>NA</updateCondition>
```

```
   <ownership>NoTransfer</ownership>
   <sharing>Publish</sharing>
   <dimensions>
      <dimension>HLAfederate</dimension>
   </dimensions>
   <transportation>HLAreliable</transportation>
   <order>Receive</order>
   <semantics>FOM Module designators as specified by the federate when
the Join Federation Execution
      service was invoked. If several identical FOM modules are
provided only the designator of the first
      of these FOM modules shall be added to the list.
   </semantics>
</attribute>
<attribute>
   <name>HLAtimeConstrained</name>
   <dataType>HLAboolean</dataType>
   <updateType>Conditional</updateType>
   <updateCondition>
   Whenever services Time Constrained Enabled or Disable Time
Constrained are successfully invoked (including via the
HLAdisableTimeConstrained interaction).
   </updateCondition>
   <ownership>NoTransfer</ownership>
   <sharing>Publish</sharing>
   <dimensions>
      <dimension>HLAfederate</dimension>
   </dimensions>
   <transportation>HLAreliable</transportation>
   <order>Receive</order>
   <semantics>Whether the time advancement of the joined federate is
constrained by other joined federates
   </semantics>
</attribute>
<attribute>
   <name>HLAtimeRegulating</name>
   <dataType>HLAboolean</dataType>
   <updateType>Conditional</updateType>
   <updateCondition>
      Whenever services Time Regulation Enabled or Disable Time
Regulation are successfully invoked
      (including via the HLAdisableTimeRegulation interaction).
   </updateCondition>
   <ownership>NoTransfer</ownership>
   <sharing>Publish</sharing>
   <dimensions>
      <dimension>HLAfederate</dimension>
   </dimensions>
   <transportation>HLAreliable</transportation>
   <order>Receive</order>
   <semantics>Whether the joined federate influences the time
advancement of other joined federates
   </semantics>
</attribute>
<attribute>
   <name>HLAasynchronousDelivery</name>
   <dataType>HLAboolean</dataType>
```

```
                <updateType>Conditional</updateType>
              <updateCondition>Whenever services Enable Asynchronous Delivery or
              Disable Asynchronous Delivery are successfully invoked (including via
              the                HLAenableAsynchronousDelivery                or
              HLAdisableAsynchronousDelivery interactions).
                </updateCondition>
                <ownership>NoTransfer</ownership>
                <sharing>Publish</sharing>
                <dimensions>
                   <dimension>HLAfederate</dimension>
                </dimensions>
                <transportation>HLAreliable</transportation>
                <order>Receive</order>
                <semantics>Whether the RTI shall deliver RO messages to the joined
              federate while the joined
                     federate's time manager state is not "Time Advancing" (only
              matters if the joined federate is
                     time-constrained).
              </semantics>
</objectModel>
```

6> SOM generation

- Below is an xml template code for SOM generation and operation (Pokorny 2012):

```
<objectModel>
    <modelIdentification>
        <name>Simulations HLA/RTI Tuturial</name>
        <type>SOM</type>
        <version>1.0</version>
        <modificationDate>2016-12-01</modificationDate>
        <securityClassification>Unclassified</securityClassification>
        <purpose>ERP Federation Tutorial</purpose>
        <applicationDomain>Engineering</applicationDomain>
        <description>ERP Federation Tutorial</description>
        <useLimitation>None</useLimitation>
        <poc>
            <pocType>Primary author</pocType>
            <pocName>Joseph Rahme Youssef</pocName>
            <pocOrg>CEFET</pocOrg>
            <pocTelephone>+961 70 109 105</pocTelephone>
            <pocEmail>joseph.rahmeyoussef@gmail.com</pocEmail>
        </poc>
        <reference></reference>
    </modelIdentification>
    <objects>
        <objectClass>
            <name>HLAobjectRoot</name>

            <objectClass>
                <name>Conversion</name>
                <sharing>PublishSubscribe</sharing>
                <semantics>A Conversion</semantics>
                <attribute>
                    <name>Model</name>
```

```
                    <dataType>HLAunicodeString</dataType>
                    <updateType>Static</updateType>
                    <updateCondition>NA</updateCondition>
                    <ownership>NoTransfer</ownership>
                    <sharing>PublishSubscribe</sharing>
                    <dimensions/>
                    <transportation>HLAreliable</transportation>
                    <order>Receive</order>
                    <semantics>The Conversion Model</semantics>
                </attribute>
            </objectClass>

        </objectClass>
    </objects>
    <interactions></interactions>
    <dataTypes></dataTypes>

    <switches>
        <autoProvide isEnabled="true"/>
        <conveyRegionDesignatorSets isEnabled="false"/>
        <conveyProducingFederate isEnabled="false"/>
        <attributeScopeAdvisory isEnabled="false"/>
        <attributeRelevanceAdvisory isEnabled="false"/>
        <objectClassRelevanceAdvisory isEnabled="false"/>
        <interactionRelevanceAdvisory isEnabled="false"/>
        <serviceReporting isEnabled="false"/>
        <exceptionReporting isEnabled="false"/>
        <delaySubscriptionEvaluation isEnabled="false"/>
        <automaticResignAction resignAction="CancelThenDeleteThenDivest"/>
    </switches>
</objectModel>
```

7> RTI specific code

- Below is a list of java code for RTI execution:

```
package edu.bordeaux.josephyoussef;

import hla.rti1516e.CallbackModel;
import hla.rti1516e.RTIambassador;
import hla.rti1516e.ResignAction;
import hla.rti1516e.RtiFactory;
import hla.rti1516e.RtiFactoryFactory;
import hla.rti1516e.exceptions.FederatesCurrentlyJoined;
import hla.rti1516e.exceptions.FederationExecutionAlreadyExists;
import hla.rti1516e.exceptions.FederationExecutionDoesNotExist;
import java.io.File;
import java.io.InputStream;
import java.io.PrintStream;
import java.net.MalformedURLException;
import java.net.URI;
import java.net.URL;
import java.util.UUID;

public class Main {
        private RTIambassador rtiamb;
        private FederateAmbassador fedamb;
        private ConversionClass conversionClass;
```

```
private void log(String message) {
	System.out.println("> " + message);
}

public ConversionClass getConversionClass() {
	return this.conversionClass;
}

private void createRtiAmbassador() throws Exception {
	log("Creating RTIambassador.");
	this.rtiamb = RtiFactoryFactory.getRtiFactory().getRtiAmbassador();
}

private void connect() throws Exception {
	log("Connecting...");
	this.fedamb = new FederateAmbassador(this);
	this.rtiamb.connect(this.fedamb, CallbackModel.HLA_IMMEDIATE);
}

private void createFederation(String federationName) throws Exception {
	log("Creating Federation " + federationName);
	try {
		URL[] modules = { new File("foms/HLAstandardMIM.xml").toURI()
				.toURL() };
		this.rtiamb.createFederationExecution(federationName, modules);
		log("Created Federation.");
	} catch (FederationExecutionAlreadyExists exists) {
		log("Didn't create federation, it already existed.");
	} catch (MalformedURLException urle) {
		log("Exception loading one of the FOM modules from disk: "
				+ urle.getMessage());
		urle.printStackTrace();
		return;
	}
}

private void joinFederation(String federationName, String federateName)
		throws Exception {
	URL[] joinModules = { new File("foms/conversion_som.xml").toURI()
			.toURL() };
	this.rtiamb.joinFederationExecution(federateName,
			"ConversionFederateType", federationName, joinModules);
	log("Joined Federation as " + federateName);
}

public void runFederate() throws Exception {
	String serial = UUID.randomUUID().toString().substring(1, 5)
			.toUpperCase();

	createRtiAmbassador();
	connect();
	createFederation("ERPFederation");
	joinFederation("ERPFederation", "ConversionClass" + serial);

	this.conversionClass = new ConversionClass(this.rtiamb);

	publishAndSubscribe();

	this.conversionClass.createNew();

	this.conversionClass.updateAttributeValues();

	System.out.println("Press a key to exit.");
	while (System.in.available() == 0) {
		try {
			Thread.sleep(2000L);
```

```
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }
        exitFromFederation();
    }

    private void exitFromFederation() throws Exception {
        this.rtiamb.resignFederationExecution(ResignAction.DELETE_OBJECTS);
        log("Resigned from Federation");
        try {
                this.rtiamb.destroyFederationExecution("ERPFederation");
                log("Destroyed Federation");
        } catch (FederationExecutionDoesNotExist dne) {
                log("No need to destroy federation, it doesn't exist");
        } catch (FederatesCurrentlyJoined fcj) {
                log("Didn't destroy federation, federates still joined");
        }
    }

    private void publishAndSubscribe() throws Exception {
        this.conversionClass.publish();
        log("Published");
    }

    public static void main(String[] args) {
        try {
                new Main().runFederate();
        } catch (Exception rtie) {
                rtie.printStackTrace();
        }
    }
}

public class FederateAmbassador extends NullFederateAmbassador {
    private Main federate;

    public FederateAmbassador(Main federate) {
        this.federate = federate;
    }

    private void log(String message) {
        System.out.println("> " + message);
    }

    public void discoverObjectInstance(ObjectInstanceHandle theObject,
                ObjectClassHandle theObjectClass, String objectName)
                throws FederateInternalError {
        log("New object found");

        if (!(this.federate.getConversionClass().isClassOf(theObjectClass)))
                return;
        try {
                this.federate.getConversionClass().createNew(theObject);
                log("New Conversion discovered");
        } catch (Exception e) {
                e.printStackTrace();
        }
    }

    public void reflectAttributeValues(ObjectInstanceHandle theObject,
                AttributeHandleValueMap theAttributes, byte[] tag,
                OrderType sentOrder, TransportationTypeHandle transport,
                FederateAmbassador.SupplementalReflectInfo reflectInfo)
                throws FederateInternalError {
        log("Attribute reflection");
```

```
                if (!(this.federate.getConversionClass().isAConversion(theObject)))
                        return;
                this.federate.getConversionClass().update(theAttributes, theObject);
        }
}

public class EncoderDecoder {
        private EncoderFactory encoderFactory;

        public HLAunicodeString createHLAunicodeString(String value) {
                return this.encoderFactory.createHLAunicodeString(value);
        }

        public HLAboolean createHLAboolean(boolean value) {
                return this.encoderFactory.createHLAboolean(value);
        }

        public HLAfloat32BE createHLAfloat32BE(float value) {
                return this.encoderFactory.createHLAfloat32BE(value);
        }

        public HLAfloat64BE createHLAfloat64BE(double value) {
                return this.encoderFactory.createHLAfloat64BE(value);
        }

        public HLAinteger32BE createHLAinteger32BE(int value) {
                return this.encoderFactory.createHLAinteger32BE(value);
        }

        public HLAinteger64BE createHLAinteger64BE(long value) {
                return this.encoderFactory.createHLAinteger64BE(value);
        }

        public String toString(byte[] bytes) {
                HLAunicodeString value = this.encoderFactory.createHLAunicodeString();
                try {
                        value.decode(bytes);
                        return value.getValue();
                } catch (DecoderException de) {
                        de.printStackTrace();
                }
                return "";
        }

        public double toFloat64(byte[] bytes) {
                HLAfloat64BE value = this.encoderFactory.createHLAfloat64BE();
                try {
                        value.decode(bytes);
                        return value.getValue();
                } catch (DecoderException de) {
                        de.printStackTrace();
                }
                return -1.0D;
        }

        public int toInteger32(byte[] bytes) {
                HLAinteger32BE value = this.encoderFactory.createHLAinteger32BE();
                try {
                        value.decode(bytes);
                        return value.getValue();
                } catch (DecoderException de) {
                        de.printStackTrace();
                }
                return -1;
        }
```

```
	public long toInteger64(byte[] bytes) {
		HLAinteger64BE value = this.encoderFactory.createHLAinteger64BE();
		try {
			value.decode(bytes);
			return value.getValue();
		} catch (DecoderException de) {
			de.printStackTrace();
		}
		return -1L;
	}

	public boolean toBoolean(byte[] bytes) {
		HLAboolean value = this.encoderFactory.createHLAboolean();
		try {
			value.decode(bytes);
		} catch (DecoderException de) {
			return false;
		}
		return value.getValue();
	}

	public EncoderDecoder() throws Exception {
		this.encoderFactory = RtiFactoryFactory.getRtiFactory()
				.getEncoderFactory();
	}
}
```