



José Alexandre Pires Ferreira

Licenciado em Ciências de Engenharia

**Monitoring morphisms to support
sustainable interoperability of enterprise
systems**

Dissertation to obtain the Master degree in Electrical Engineering
and Computer Science

Orientador: Ricardo Luís Rosa Jardim Gonçalves

Professor Auxiliar, Departamento de Engenharia Electrotécnica
Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Orientador: Carlos Manuel Melo Agostinho, Investigador, UNINOVA

Júri:

Presidente: Doutor José António Barada de Oliveira
Arguente: Doutor João Pedro Mendonça de Assunção da Silva
Vogais: Doutor Ricardo Luís Rosa Jardim Gonçalves
Mestre Carlos Manuel de Melo Agostinho



FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Fevereiro 2012

Copyright

Monitoring morphisms to support sustainable interoperability of enterprise systems © José Alexandre Pires Ferreira

A Faculdade de Ciências e Tecnologia e a Universidade Nova de Lisboa têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objectivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

To my family

ACKNOWLEDGEMENTS

I would like to thank all the people who, in some way, supported me during the realisation of my course and this dissertation.

First my family, that never gave up on me during the end of this phase of my life, which supported me all these years. To my brother that helped me during the time I was writing this dissertation.

To Sónia that was my last motivation and help to finally finish the course and this dissertation, that gave me hope and believed in me, thus made me believe in myself. And kept me motivated and pushed me all the time in order to finish the dissertation.

To all my colleagues at GRIS, especially to Carlos Agostinho that helped and believed in me all this time, pushing me to do a better work and to successfully complete this dissertation. João Sarraipa to the help he gave me during the dissertation.

To my advisor Dr. Ricardo Gonçalves for believing in my capabilities and giving me his advice towards the successful completion of this work, and the great opportunity to work in his group.

Finally, to my special friends, who shared all the hard work during the course, that took too many years, but finally is over. Thank you Paulo Polaina, Francisco Cavaco and António Paulo, for all these years together and the great nights we passed together.

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 234344 (www.crescendo-fp7.eu/).

ABSTRACT

Nowadays, organizations are required to be part of a global collaborative world. Sometimes this is the only way they can access new and wider markets, reaching new opportunities, skills and sharing assets, e.g. tools, lessons learnt. However, due to the different sources of enterprise models and semantics, organizations are experiencing difficulties in exchanging vital information via electronic and in a seamlessly way. To solve this issue, most of them try to attain interoperability by establishing peer-to-peer mappings with different business partners, or in optimized networks using neutral data standards to regulate communications. Moreover, the systems are more and more dynamic, changing frequently to answer new customer's requirements, causing new interoperability problems and a reduction of efficiency. This dissertation proposes a multi-agent system to monitor existing enterprise systems, by being capable of detecting morphism changes. With this, network harmonization breakings are timely detected, and possible solutions are suggested to regain the interoperable status, thus enhancing robustness for reaching sustainability of business networks.

KEYWORDS

The keywords of this dissertation are: Interoperability, Model Morphisms, Sustainable Interoperability, Multi-Agent Systems and Model-Based Systems Engineering.

Hoje em dia, as organizações têm a necessidade de se integrarem no mundo global e colaborativo. Por vezes esta é a única forma que estas têm para alcançar novos e maiores mercados, competências e partilha de recursos. No entanto, devido às diversas fontes de modelos empresariais e semântica, as organizações estão a enfrentar dificuldades na troca de informações em formato electrónico. Para resolver este problema, é prática tentar alcançar-se a interoperabilidade através mapeamentos ponto a ponto com os diversos parceiros de negócios, ou através de redes optimizadas, que usam normas de modelos de dados para regular as comunicações dentro da rede colaborativa. Além disso, os sistemas são cada vez mais dinâmicos, sofrendo assim, frequentes alterações como resposta à exigência apresentada por novos clientes, causando problemas de interoperabilidade e uma redução de eficiência. Este trabalho, propõe um sistema de motivação baseada em multi-agentes, capaz de detectar alterações nos morfismos do “Communicator Mediator”. Assim, as quebras de harmonização numa rede colaborativa são detectadas a tempo, e o sistema é capaz de sugerir possíveis soluções para recuperar o estado interoperável, aumentando assim a robustez e por consequência alcançar a sustentabilidade das redes de negócios.

PALAVRAS-CHAVE

As palavras-chave desta dissertação são: Interoperabilidade, Modelos de Morfismos, Sustentabilidade Interoperável, Sistema de Multi Agentes e Modelos Baseados em Engenharia de Sistemas.

TABLE OF ACRONYMS

ACC	Agent Communication Channel
ACL	Agent Communication Language
AMS	Agent Management System
AP	Application Protocol
ATHENA	Advanced Technologies for interoperability of Heterogeneous Enterprise Networks
ATL	ATLAS Transformation Language
BDA	Behavioural Digital Aircraft
C ⁴ IF	Connection, Communication, Consolidation, Collaboration Interoperability Framework
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CAS	Complex Adaptive Systems
CAS-SIF	CAS to support Sustainable Interoperability Framework
CIM	Computer Independent Model
CM	Communicator Mediator
CN	Collaboration Network
CRESCENDO	Collaborative and Robust Engineering using Simulation Capability Enabling Next Design Optimisation
CWM	Common Warehouse Metal Model
DF	Director Facilitator
EI	Enterprise Interoperability
EIF	European Interoperability Framework
ENS	Event Notification Service
ERP	Enterprise Resource Planning
ES	Enterprise Systems
EPS	European Public Services
EU	Europe Union
FIPA	Foundation for Intelligent Physical Agents
GRIS	Group for Research in Interoperability of Systems
GUI	Graphical User Interface

ICT	Information and Communication Technology
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IL & ML	Interoperability Layers and Maturity Levels
IM	Information Model
INTEROP	Interoperability Research for Networked Enterprises Applications and Software
IS	Information Systems
ISA	Industry Standard Architecture
ISO	International Organisation for Standardization (http://www.iso.org)
iSURF	An Interoperability Service Utility for Collaborative Supply Chain Planning across Multiple Domains Supported by RFID Devices
IT	Information Technology
JADE	Java Agent DEvelopment framework
KB	Knowledge Base
LC	Life Cycle
LISI	Levels of Information System Interoperability
MAS	Multi-Agent System
MBSE	Model Driven-Based Engineering
MDA	Model Driven Architecture
MDD	Model Driven Development
MDE	Model Driven Engineering
MDI	Model Driven Interoperability
MIRAI	Monitoring morphisms to support sustainable Interoperability of enterprise systems
MOF	Meta Object Facility
MoMo	Model Morphism
MSI	Modelling and Simulation Interoperability
OMG	Object Management Group (http://www.omg.org)
OWL	Web Ontology Language
P2P	Peer to Peer
PIM	Platform Independent Model
PLC	Product Life Cycle
PLM	Product Lifecycle Management
PSM	Platform Specific Model

SE	Systems Engineering
STEP	Standard for the Exchange of Product Data
SysML	System Modelling Language
TTCN	Tree and Tabular Combined Notation
UDDI	Universal Description, Discovery and Integration
UML	Unified Modelling Language
UNINOVA	Institute for the Development of New Technology
VHDL	VHSIC Hardware Description Language
WAN	Wide Area Networks
WSIG	Web Service Integration Gateway
XMI	XML Metadata Interchange
XML	Extensible Markup Language

TABLE OF CONTENTS

Acknowledgements	vi
Abstract	viii
Resumo	x
Table of Acronyms	xii
Table of Contents	xvi
Table of Figures	xviii
List of Tables	xx
1. Introduction	1
1.1. Research Framework and Motivation	2
1.2. Research Method	3
1.3. Research Problem and Questions	4
1.4. Hypothesis	4
1.5. Dissertation Outline	4
2. Model-Based Systems Engineering	7
2.1. Requirements and Benefits of Modelling	8
2.2. Product Life Cycle	9
2.3. MBSE Information Models and Data Standards for Interoperability	12
2.4. Systems Engineering Ontology	16
2.5. Open Research Issues	16
3. Enterprise Systems Interoperability	19
3.1. Interoperability Layers and Maturity Levels (IL & ML)	19
3.2. Model-Driven Interoperability	26
3.3. Semantic Interoperability	29
3.4. Model Morphisms	29
3.5. CAS-Based Framework to Support Sustainable Interoperability (CAS-SIF)	34
3.6. Open Research Issues	35
4. Multi-Agent Systems (MAS) to support sustainable interoperability	37
4.1. MAS Overview	38
4.2. MIRAI Framework and Architecture	39
4.3. Communication Mediator (CM)	41
4.4. MIRAI Intelligent Supervisor Block	42
4.5. MIRAI External Communicator Block	49
4.6. MIRAI Administration Block	51
4.7. MIRAI Life Cycle Monitor Block	52
4.8. MIRAI Usability Cases	54
5. Proof-of-concept Implementation	57
5.1. Application Scenarios	57

5.2.	Implementation Steps	64
5.3.	Implementation Overview and Technology Used	68
6.	Testing and Hypothesis Validation	73
6.1.	Testing Methodologies	73
6.2.	Requirements and Functionalities	77
6.3.	Testing	79
6.4.	Hypothesis Validation	86
6.5.	Scientific Validation	86
6.6.	Industrial Validation	87
7.	Final Considerations and Future Work	89
7.1.	Future Work	90
8.	References	93
9.	Annex	97
9.1.	List of MisMatches	99

TABLE OF FIGURES

Figure 1-1 - Phases of the Classical Research Method (based on (Camarinha-Matos 2010))	3
Figure 2-1 - MBSE scope (adapted from (Nallon 2003), (Friedenthal et al. 2008)).....	7
Figure 2-2 - Life Cycle based on ISO/IEC 15288 (Nallon 2003).....	10
Figure 2-3 – Life Cycle development models: (a) Waterfall, (b) Spiral, (c) "Vee" (Estefan 2007)	12
Figure 2-4 - SysML diagram types (OMG 2011a).....	13
Figure 3-1 - C ⁴ IF Framework (Peristeras & Tarabanis 2006)	20
Figure 3-2 - LSI interoperability maturity model (C4ISR 1998)	21
Figure 3-3 - ATHENA MDI Framework (ATHENA 2010)	22
Figure 3-4 - EIF Framework (ISA 2010)	23
Figure 3-5 - Interoperability Classification Framework (H. Panetto 2007)	24
Figure 3-6 - Interoperability Practices Layers (Carlos Agostinho & Ricardo Jardim-Goncalves 2009)..	25
Figure 3-7 - The levels of MDA approach (Petzmann et al. 2007).....	28
Figure 3-8 - The relationship between model and meta-model (B Selic 2003)	30
Figure 3-9 - Example of mappings between two ontologies.....	31
Figure 3-10 - CAS-SIF Framework (Carlos Agostinho & Ricardo Jardim-Goncalves 2009)	34
Figure 4-1 - MIRAI Network.....	39
Figure 4-2 - MIRAI Architecture.....	40
Figure 4-3 – Structure of Communication Mediator (J. Sarraipa et al. 2010).....	42
Figure 4-4 - System to re-adapt the mappings.....	42
Figure 4-5 - Use Case of Agent Monitor Mediator.....	43
Figure 4-6 - Use Case of Agent MoMo	44
Figure 4-7 - MIRAI High Level Interaction	44
Figure 4-8 - State Machine of Intelligent Supervisor.....	45
Figure 4-9 - Diagram sequence of Intelligent Supervisor and External Communicator with internal problem	46
Figure 4-10 - Diagram sequence of Intelligent Supervisor and External Communicator with external problem	47
Figure 4-11 - Detection and propose of morphisms	48
Figure 4-12 - Mappings between the models	48
Figure 4-13 - Use Case of Agent Communicator	50
4-14 - Actions of the Agent Communicator	50
Figure 4-15 - Use Case of Agent User	51
4-16 - Interaction between the Agent User and the rest of the agents.....	52
Figure 4-17 - Use Case of Agent Persistor	52
Figure 4-18 - Use Case of Agent Persistor Police	53
Figure 4-19 - State Machine of Life Cycle Monitor.....	53
Figure 4-20 - Diagram Sequence of Life Cycle Monitor.....	54
Figure 4-21 – Morphism Evolution Scenario	54
Figure 4-22 – New Enterprise Scenario	55
Figure 4-23 – New Connection Scenario	55
Figure 4-24 – Removal of an Enterprise Scenario	56
Figure 5-1 - MIRAI Network for the Crescendo Scenario	58

Figure 5-2 – Representation of the mappings between the models without the evolution in the system	59
Figure 5-3 - Representation of the mappings between the models with the evolution in the system	60
Figure 5-4 - Mappings of the scenario.....	61
Figure 5-5 - Reacting of the network caused by a change of software.....	62
Figure 5-6 – Scenario of concepts mappings.....	63
Figure 5-7 – Mappings of the concepts	63
Figure 5-8 - MIRAI Architecture.....	64
Figure 5-9 - Activity Diagram of Agent Monitor Mediator and Agent MoMo	65
Figure 5-10 - Example of new MatchClass creation	66
Figure 5-11 - Example of a mapping.....	67
Figure 5-12 - Web Service Scenario.....	68
Figure 6-1 - Family example	79
Figure 6-2 - Mapping relating Name in EXPRESS with Family Name in EXPRESS 2	80
Figure 6-3 - MIRAI GUI and creating Agents.....	80
Figure 6-4 - Agent Persistor resurrecting a Dead Agent.....	81
Figure 6-5 - Agent Monitor Mediator working.....	83
Figure 6-6 - Agente MoMo proposing a possible solution	84
Figure 6-7 - Proposal of a new morphism	84
Figure 6-8 - Asking to the user to choose solution.....	85
Figure 6-9 - Current barriers and envisaged CRESCENDO solution.....	87

LIST OF TABLES

Table 3-1 - Semantic Mismatches (based on (Carlos Agostinho et al. 2011)).....	32
Table 5-1 – Purpose of the used technology by the Proof-Of-Concept Implementation	69
Table 6-1 - Simplified example of a TTCN table test	77
Table 6-2 - Agent Persistor functional test.....	81
Table 6-3 - Agent Persistor Police functional tests.....	82
Table 6-4 - Agents Persistor's non-functional tests.....	82
Table 6-5 - Agent Monitor Mediator functional test.....	83
Table 6-6 - Agent Monitor Mediator non-functional test	84
Table 6-7 - Agent MoMo functional test	85
Table 6-8 - Agent MoMo non-functional test	86

1. INTRODUCTION

In the last few years, collaboration between different enterprises have increased significantly with the possibility of combining resources towards achieving a common goal, as well as having to boost productivity and reduce expenses. Thus the chances of survival of smaller enterprises in the current turbulent market are increasing (Hassell et al. 2006). The enterprises are then forced to be adaptable with other enterprises network environments in order to have a collaborative business, while following all the Product Life Cycle (PLC) phases (C. Agostinho, Malo, et al. 2007).

Schrage (Schrage 1990), emphasizes that the issue in collaboration "*is the creation of value*". In this context, not only data exchange in industrial process (e.g. supply chain, e-procurement, etc), but also in Systems Engineering (SE) has been the target of multiple collaborative networks. It demands the combination of multiple complex processes, such as requirements elicitation, product design and development, teams' management, logistics, etc, that normally are performed in conjunction by different enterprises. The systems engineering method recognizes each system as an integrated whole though composed of diverse, specialized structures and sub-functions that aid in the development and Life Cycle (LC) of a product, using information and product models to describe and integrate the whole process (Chestnut 1965).

In this sense, being defined as the ability that two or more systems have to exchange information and use it accurately (Geraci et al. 1991), interoperability, namely the lack of it, could disturb the functionalities of the enterprise systems networks and decrease their competitiveness and innovation. Thus, it is important to maintain the interoperability, since it is a key concern for network enabled complex systems, which in turn have eased the collaborative enterprise and implicitly globalisation (Kotze & Neaga 2010).

When applications are used to manage the same or different models within a network, several types of costs could be incurred regarding interoperability, like translation or data re-entry for seamless information flows (Carlos Agostinho et al. 2011). Moreover, if systems are only partially interoperable, translation or data re-entry is required for information flows (Carlos Agostinho et al. 2011). This situation has a direct impact in the business processes and SE practices that cross-cut these organizations (e.g. collaborative product development), since it is essential to have interoperable models and efficient data exchange for the integrated design, product breakdown (decomposition into parts) and manufacturing (Haskins 2006). These techniques enable the development of complex and costly enabling systems, such as a flight simulator or a high-volume production line, which allow validation of the system's concepts, or supports training of personnel in ways that would otherwise be cost prohibitive (Haskins 2006). The main result of modelling is to predict characteristics (performance, reliability, operations, cost, etc.) across the spectrum of system attributes throughout its LC. This is what Model-Based System Engineering (MBSE) specifies and has also the aims to facilitate systems engineering activities that have traditionally been performed using the document-based approach.

However, in our days, when achieved, this stability is hard to maintain since due to market requirements and the increasingly dynamicity of customer needs, business requirements are constantly being adapted, causing systems, models and semantics to change, thus leading to harmonization breaking and disruption of the interoperability in the enterprise systems networks (Carlos Agostinho & Ricardo Jardim-Goncalves 2009).

These problems are the cause of the breaking of the harmonization in the interoperability. These occur because of different issues, and are described by Kotzé and Neaga in (Kotze & Neaga 2010):

- Using standards do not always guarantee achieving interoperability;
- There is a need for analysis and evaluation of interoperability itself;
- Improvements of existing system interoperability should be beneficial;
- Rapid development of technology may decrease the degree of systems interoperability if the interoperability requirements are not considered at design level.

There are several approaches, from different perspectives, to achieving interoperability (Kotze & Neaga 2010). Since, almost all the solutions try to solve the problems of interoperability, and that is part of the possible solution. Because, as already mentioned, for different reasons, sometimes it is not possible to maintain that, it is also needed to monitor and support the system, and this is another part of the problem, maintain the interoperability status.

This is the motivation behind of this thesis, to advance the research on organizations interoperability using a multi-agents framework to monitor the mappings between the different organizations and maintain the interoperability state. Using tuples suggested by Agostinho et al. (Carlos Agostinho et al. 2011), as a possible solution to the problem on the modelling of data, semantic and structural mappings as traceable tuples that when integrated in knowledge bases that are dedicated to managing mismatches during communications, so that network evolutions can be closely monitored and reactions can be triggered automatically.

1.1. Research Framework and Motivation

With each passing day the need to share information is becoming more and more important for the organisations, even more important is that failures never occur in that information. It is important to help the integration of the PLC phases, since manufacturers, distributors, designers, retailers, warehouses, all have their proprietary solutions, so maintaining interoperability between them is crucial (Ricardo Jardim-Goncalves, Carlos Agostinho, et al. 2011).

So, to maintain good collaboration functionality and good interoperability status between the organisations is not an easy task, but whenever it is achieved, it will bring some benefits to the organisations, some of those benefits are: shorter response times, reduced cycle times from order to cash, reduced inventories and significant reduction in the number of costly errors, keeping companies ahead of competition and positioning them for world-class performance (Ricardo Jardim-Goncalves,

Carlos Agostinho, et al. 2011), thus using the MBSE paradigm to describe the PLC.

So, this dissertation aims to contribute to maintain the interoperability status of a collaborative network, by doing an inspection in the information and warning whenever there is a lack of interoperability. By doing this, there will be less waste of time, either at production line or by searching for a possible solution.

1.2. Research Method

The research method used in this dissertation is based on the classical research method (Camarinha-Matos 2010) and consists on seven steps. This method begins with the search of the problem, and ends with the interpretation of the results. If for some reason the results are unsatisfactory this method allows to return to the first step and experiment a new approach. This method is represented in Figure 1-1.



Figure 1-1 - Phases of the Classical Research Method (based on (Camarinha-Matos 2010))

Each step of this method is described below:

1. Research Question / Problem: It is the most important step in a research, because it is here that the area of interest is defined. It is possible that several secondary questions appear in order to help define the main idea. These questions are located in the section 1.3.
2. Background / Observation: Here is the section to do the research for the state of the art, by studying previously similar works, presenting literature review and previously projects with the objective to help start the dissertation. This is important because previous works will have an impact on the new one, and help in a new approach. So, showing existing ideas of other authors will create and open new solutions, to be used in the development for this dissertation. This background is located in the section 2 and 3.
3. Formulate Hypothesis: Here is where the predicted results of the research work are managed. It is important that an hypothesis is simple to understand, specific, conceptually clear and measured. In section 1.4 is the hypothesis of this dissertation.

4. Design Experiment: In this step the detailed plan of the experimental phase steps that is often composed by the design of a prototype or even system architecture is presented. The section 4 and 5 are the design of a prototype and the proof-of-concept, respectively.
5. Test Hypothesis / Collect Data: In this phase the evaluation of the system architecture is made. This is made by testing and simulating different scenarios. In each test data for further analysis and hypothesis validation is collected. These tests are done in section 6.2.
6. Interpret / Analyze Results: Here is where the results are analysed and the veracity of the hypothesis is found out. If for some reason the results are not satisfactory, it is possible to try a different approach, and return to step 1. Moreover, when positive results are achieved, it is possible to look to next steps giving some ideas for further research. This is made in the sections 6.2 and 6.3.
7. Publish Findings & Transfer to Industry: When good results are achieved and a good contribution to the scientific community is made, it is important to share these results with the community and in some cases transfer the technology to industry. To present the results to the scientific community is made using different means, like scientific paper, in conferences, Journals, and so on. In the case of the industry it is very important to accept the results, so it can be used. The scientific validation is presented in section 6.5 and the industrial validation is presented in section 6.6.

1.3. Research Problem and Questions

- Is it possible to maintain the interoperability status in a collaborative network, responding to system's changes and models evolution?
 - In a network of enterprises, how is it possible to detect information model changes and system's evolutions?
 - Can Multi-Agent Systems (MAS) propose possible mapping morphisms readjustments to system administrators?
 - How will the Network receive the knowledge about the changes at the enterprise level?

1.4. Hypothesis

- If the MIRAI framework is able to monitor the system and detect evolutions in the model morphisms at the enterprise level, then it is possible to detect changes in the systems information models, and create new mappings, thus re-enabling network interoperability.

1.5. Dissertation Outline

In this section the context of this dissertation is described and is explained the content of each section. Sections 2 and 3 present the topics that are the background of this thesis. Section 2 covers Model-Based Systems Engineering area by doing a perspective about MBSE, the life cycle of the

product, information models and MBSE ontology. Section 3 does a brief description of some approaches to classify the interoperability layers, and then covers the Model-Driven Interoperability (MDI) method for systems development and implementation, where a brief description of the Model-Driven Architecture (MDA) and Model-Driven Engineering (MDE) is done. Later it will be described what semantic interoperability is and finally it will be introduced the model morphisms (MoMo).

Finally in section 4, a description about Multi-Agent Systems is done, and an explanation about the advantages of using MAS in the interoperability is presented. This section defines a framework based in MAS that has the objective of monitoring the existing mappings stored in each enterprise's Communicator Mediator (CM), with the aim to help the organisations to achieve the interoperability status. The proof-of-concept implementation steps are then presented in section 5, describing the architecture of the agents and how the monitor of the morphisms works. Section 6 is about the validation of the proof-of-concept implementation and the proposed framework, where the functional and non-functional tests of the framework are presented.

Finally, section 7 draws the conclusions and future work topics.

2. MODEL-BASED SYSTEMS ENGINEERING

Usually in large projects the organizations employ a document-based systems engineering approach. This approach is characterized by the generation of textual specifications and design documents, in hard-copy or electronic file format, that are then exchanged between customers, users, developers, and testers. These documents and drawings represent the systems requirements and the design information, then the systems engineering are responsible on controlling the documentation and ensuring the documents and drawings are valid, complete, and consistent (Ogren 2000).

This method is rigorous but has some limitations, like the completeness, consistency, and relationships between requirements, design, engineering analysis, and test information. These are difficult to assess since the information is spread across several documents (Ogren 2000). Because of these limitations in SE and the need to improve the quality and to speed the access to information in 1993 the MBSE was introduced, with the intention to facilitate SE activities.

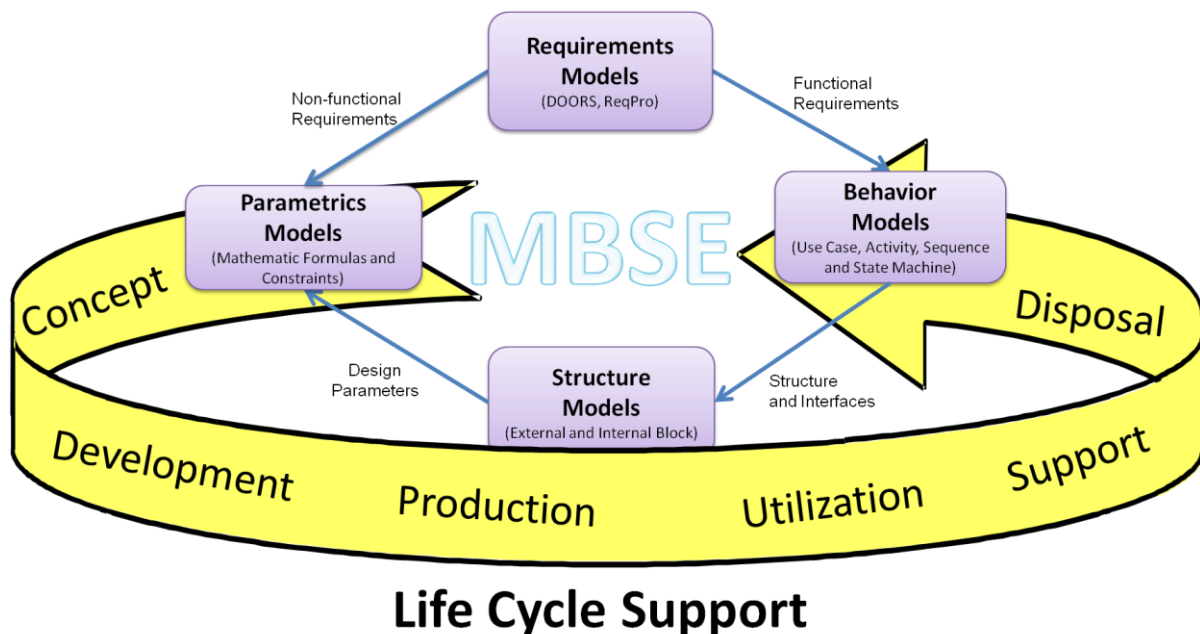


Figure 2-1 - MBSE scope (adapted from (Nallon 2003), (Friedenthal et al. 2008))

MBSE is the formalized application of modelling to support the systems engineering processes, namely requirements, design, analysis, verification and validation activities beginning at the conceptual design phase and continuing throughout development and later LC stages (Operations & Crisp II 2007), (INCOSE 2011).

Among the many ways to describe the LC of a product or a system, some are more focused on design and production, whilst others are centred on support. Recently, the disposal stages have also been carefully studied due to the impact in the environment (Carlos Agostinho et al. 2011), (Frisch et al. 2007). Figure 2-1 illustrates a possible view, since it takes many models to describe a system. This model presents the PLC, starting from the concept where the stakeholder's needs are identified until

the disposal of the product, then merged with the development process used in the MBSE, namely (Nallon 2003):

- The *requirements models* that represent the relationships between user requirements and/or model objects. A primary benefit of modelling requirements is the opportunity this provides for analyzing them with techniques such as requirements animation, reasoning, etc (Nuseibeh & Easterbrook 2000), the examples of tools used to construct this models are the DOORS and ReqPro;
- The *behaviour models* to represent the intended and unintended behaviours for a system of interest (e.g. a product), thus responding to functional requirements, the tools used to represent the behaviour models are the Use Case, Activity, Sequence and State Machine Diagram;
- The *parametric models* to reply to the non-functional requirements representing the formal relationships and constraints of the system and its components. The tools used in the parametrics models are the Mathematic formulas and constraints;
- And finally the *structure models* which describe the enterprise and system level contexts from both the logical and physical viewpoints. These are represented with External and Internal Block.

2.1. Requirements and Benefits of Modelling

Among the multiple ways to model a system or a product, there is no absolute recipe. However, modelling using MBSE need to have the requirements properly formalized so that the remaining models can be specified with a good level of detail and the final product can achieve the expected quality. Thus, Ogren (Ogren 2000) defines 5 key principles on modelling, namely:

- *Determinism with formality*, so that everything expressed in the model has a single, defined and obvious meaning;
- *Understandability*, since systems engineering should be done in close cooperation with end users, the models are only useful is they are readily understood, without extensive education or experience in software or mathematics;
- *Inclusion of system missions* to be able to extract the systems missions out the models and express how different parts of the system contribute together;
- *Modelling of structure and behaviour*, to support splitting a system into subsystems, with clarification of interfaces between these systems, and the modelling technique shall also allow a definition of behaviour within the subsystems defined;
- *Possibility of verification support*, it should be possible to verify a completed model. This verification can be against defined requirements, but it can also concern verification of completeness, etc. For complex systems, verification will often require computer support, depending on the large amounts of information to be managed.

These are the most important points to achieve an acceptable quality to modelling a complex system, with these requirements the use of the MBSE in the design of the PLC brings some benefits to the organizations, one of the most important benefits is to provide an opportunity to address many of the limitations of the document-based approach by providing a more rigorous means for capturing and integrating system requirements, design, analysis, and verification information, and facilitating the maintenance, assessment, and communication of this information across the system's life cycle. Other MBSE potential benefits are described on (Ogren 2000) and also described below:

- Enhanced communications;
 - ✓ Shared understanding of the system across the development team and other stakeholders;
 - ✓ Ability to integrate views of the system from multiple perspectives;
- Reduced development risk
 - ✓ Ongoing requirements validation and design verification;
 - ✓ More accurate cost estimates to develop the system;
- Improved quality
 - ✓ More complete, unambiguous, and verifiable requirements;
 - ✓ More rigorous traceability between requirements, design, analysis, and testing;
 - ✓ Enhanced design integrity;
- Increased productivity
 - ✓ Faster impact analysis of requirements and design changes;
 - ✓ Reuse of existing models to support design evolution;
 - ✓ Reduced errors and time during integration and testing;
 - ✓ Automated document generation;
- Enhanced knowledge transfer
 - ✓ Specification and design information captured in a standard format that can be accessed via query and retrieval.

Therefore, the MBSE framework can bring an added value to enterprise networks, maximizing the efficiency of collaborations and stimulating interoperability through the different models used along the system LC of the products.

2.2. Product Life Cycle

Every system has a life cycle, even if it is not formally defined. The system of interest, follows a life cycle from its initial conceptualization through its eventual disposal (Haskins 2006; ISO 19760 2002).

The purpose in defining life cycle of a system is to meet its required functionality throughout its life in an efficient manner. This is usually done by defining life cycle stages, and using decision gates to determine readiness to move from one stage to the next (Haskins 2006), (ISO 19760 2002).

2.2.1. Life Cycle Features

According to Haskins (Haskins 2006) the life cycle features are divided in two topics, the three aspects of the life cycle and the decision gates, the first describes that every system or product life cycle consists of three aspects, business aspect (business case), budget aspect (funding), and the technical aspect (product). It is important that these three aspects are in balance and given equal emphasis at all decision gate reviews.

The Decision gates ensure that new activities are not pursued until the previously scheduled activities (on which new ones depend) are satisfactorily completed and placed under configuration control (Haskins 2006).

The INCOSE in (Haskins 2006) proposed that the primary objectives of decision gates are to:

- Ensure that the elaboration of the business and technical baselines are acceptable and will lead to satisfactory verification and validation;
- Ensure that the risk of proceeding to the next step is acceptable;
- Continue to foster buyer and seller teamwork.

In the beginning of each project at least two decision gates exist: authority to proceed and final acceptance of the project deliverable (Haskins 2006).

2.2.2. Life Cycle Stages

In the Figure 2-2 the six life cycle stages based on ISO/IEC 15288 (Nallon 2003) are represented, every stage has a purpose and a decision gate, the decision gate is equal for everyone, and can be one or more of the different gates that are represented here: Execute next stage; Continue this stage; Go to a preceding stage; Hold project activity and Terminate project. The purpose of each life cycle stage is described above and they are based on (Estefan 2007), (Faulconbridge & Ryan 2005):

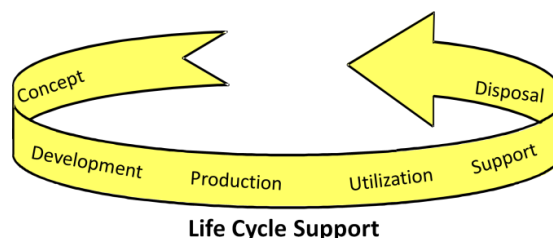


Figure 2-2 - Life Cycle based on ISO/IEC 15288 (Nallon 2003)

Concept

The purpose of the Concept Stage is to assess new business opportunities and to develop preliminary system requirements and a feasible design solution.

In this stage, the studies to evaluate the concepts are done and a justification of the reason of the chosen concept is required. This evaluation differ from being hardware or software, in hardware mock-ups are made and in software created code, after that some engineering models and simulations are executed, since in some cases it is necessary to do more tests.

Some prototypes to help in the verification of the feasibility of the concepts and to explore risks and opportunities are then created. These studies are made to control the risk and opportunity evaluation to include affordability assessment, environmental impact, failure modes, and hazard analysis.

The objective of this study is to provide confidence that the business case is sound and the proposed solutions are achievable (Haskins 2006).

Development

The purpose of the Development Stage is to execute the development of the system-of-interest that meets acquirer requirements and can be produced, tested, evaluated, operated, supported, and retired.

It is decided at this stage the baseline of the system components, where they are integrated and verified, it is here that the development of the individual subsystems and components in the system are then initiated. Some models that help in the evaluation of the system, within several existing models are used in the reinforcement of the subject. There are three models that are the most used (Figure 2-3): the *Royce Waterfall Mode*, *Boehm's Spiral Model*, and *Forsberg and Moog's "Vee" Model*. It is possible to pass to the next stage when the definition of the system is sufficiently detailed to commence the production.

Production

The purpose of the Production Stage is to produce or manufacture the product, to test the product, and to produce related supporting and enabling systems as needed.

The products will be produced following the design specifications defined in the concept stage. With the finished product in hand some tests are made and the product is evaluated to ensure that the final system configuration meets its intended purpose. This is needed because if for some reason the product has defects, thus detected in time, and causing the production to costs less. The production has to follow the documentation comprising the Product Baseline, so some auditions to control are made, if the auditions run successfully, the Product Baseline is approved.

Utilization

The purpose of Utilization Stage is to provide logistics, maintenance, and support services that enable continued system-of-interest operation and a sustainable service.

Product modifications are often planned during the utilization stage, because occasionally it is necessary to rectify some performance shortfalls, to meet changing operational requirements or external environments, or to enhance current performance or reliability.

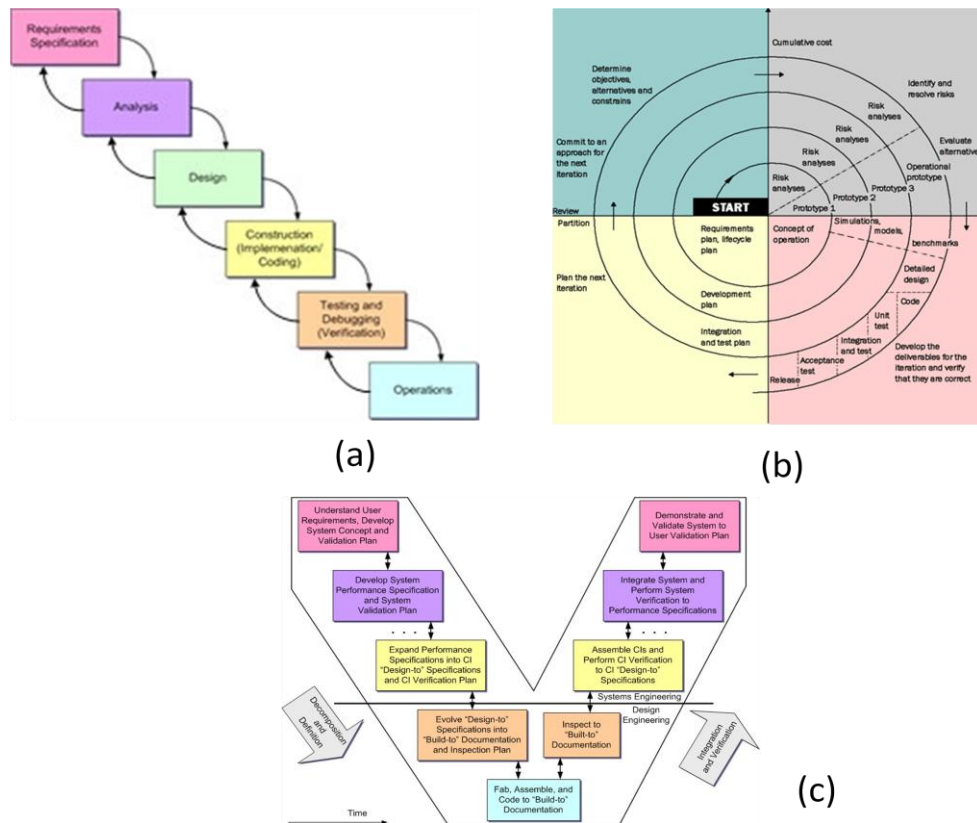


Figure 2-3 – Life Cycle development models: (a) Waterfall, (b) Spiral, (c) "Vee" (Estefan 2007)

Support

The purpose of the Support Stage is to provide logistics, maintenance, and support services that enable continued system-of-interest operation and a sustainable service.

In the LC of a product some supportability problems usually happens, so some modifications in the product to reduce operational costs or extend their life are proposed. These changes require systems engineering assessment to avoid loss of system capabilities while under operation.

Disposal

The Disposal Stage is to prepare the product for the removal of a system-of-interest and related operational and support services, and to operate and support the retirement system itself.

Since every product has one day to be retired from the market and with that it completes the entire life cycle of the product. At this point, the priority of the systems engineering is to ensure that the disposal requirements are satisfied. This phase was already planned for disposal as it is a part of the system definition during the concept stage.

2.3. MBSE Information Models and Data Standards for Interoperability

Early it was said that the MBSE is the formalized application of modeling to support the systems engineering processes, this is made by describing all the process using information models, since information models is a representation of concepts, relationships, constraints, rules, and operations to

specify data semantics for a chosen domain of discourse (Lee 1999).

For example in the Figure 2-1 is a representation of the MBSE framework, and in the middle of the figure is an information model used to model the systems. When working with complex systems, becomes difficult to create this models, so to help in this, the MBSE suggest the use one of the two standard languages, i.e. SysML¹ and AP233², to help in the creation of system engineering models.

2.3.1. SysML

The System Modelling Language (SysML) is a graphical modelling language from the Object Management Group (OMG) that supports the analysis, specification, design, verification and validation of complex systems (Ogren 2000). The language was created to specify and architect systems and its components. With it, it is possible to integrate with UML for software design and VHDL³ for hardware design.

SysML was design to provide a simple but powerful construction for modelling a wide range of systems engineering problems. It is particularly effective in specifying requirements, structure, behaviour, allocations, and constraints on system properties to support engineering analysis. The language is intended to support multiple processes and methods such as structured, object-oriented, and others, but each methodology may impose additional constraints on how a construct or diagram kind may be used (OMG 2010).

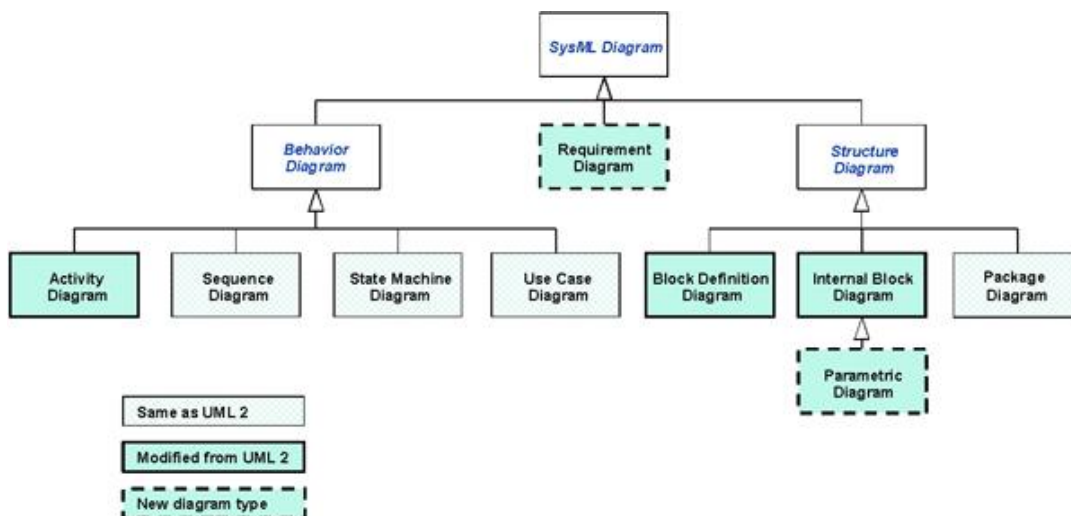


Figure 2-4 - SysML diagram types (OMG 2011a)

The SysML includes nine diagrams as shown in the Figure 2-4. These diagrams are described above, along with its relationship to UML diagrams, and they are based on (Ogren 2000):

- *Requirement diagram* represents text-based requirements and their relationship with other

¹ Systems Modelling Language (SysML) - <http://www.omgsysml.org/>

² STEP AP 233: Systems engineering data representation - <http://www.ap233.org/>

³ VHSIC hardware description language (VHDL) - <http://www.eda.org/vasg/>

requirements, design elements, and test cases to support requirements traceability, they don't exist in UML;

- *Activity diagram* represents behaviour in terms of the ordering of actions based on the availability of inputs, outputs, and control, and how the actions transform the inputs to outputs, is a modification of UML activity diagram;
- *Sequence diagram* represents behaviour in terms of a sequence of messages exchanged between parts, is the same as UML sequence diagram;
- *State machine diagram* represents behaviour of an entity in terms of its transitions between states triggered by events, is the same as UML state machine diagram;
- *Use case diagram* represents functionality in terms of how a system or other entity is used by external entities (i.e., actors) to accomplish a set of goals, is the same as UML use case diagram;
- *Block definition diagram* represents structural elements called blocks, and their composition and classification, is a modification of UML class diagram;
- *Internal block diagram* represents interconnection and interfaces between the parts of a block, is a modification of UML composite structure diagram;
- *Parametric diagram* represents constraints on property values, such as $F=m*a$, used to support engineering analysis, not exist in UML;
- *Package diagram* represents the organization of a model in terms of packages that contain model elements, is the same as UML package diagram.

With these diagrams the SysML provides a means to capture the system modelling information as part of an MBSE approach without imposing a specific method on how this is performed (Ogren 2000).

2.3.2. AP233

The AP233 is a standard Application Protocol (AP) under STEP initiative (ISO/DIS 10303-233 2011), (ISO TC184/SC4 2004). ISO 10303 (STEP) is an international standard for the computer-interpretable representation of product information and for the exchange of product data. Its objective is to provide a means of describing product data throughout its LC, independent from any particular system (Frisch et al. 2007).

The product data are described using EXPRESS (OMG 2011b), the STEP modelling language, which combines methods from the entity-attribute-relationship languages with object modelling concepts. EXPRESS provides general and powerful mechanisms for representation of inheritance among the entities constituting the AP standard model, and it also encloses a full procedural programming language used to specify constraints on populations of instances.

Like in SysML the AP233 also has different models used to describe a system. These models

are used to help in the MBSE LC of the product. Above are the different models based on (OSJTF 2006):

- *Behavioral Models* is a data model that capture semantics associated with how a system acts or performs (responds to excitation); include functions, inputs, outputs and control operators which define the ordering of functions; includes both Function-based and State-based behaviors; enables generation of Functional Flow Block Diagrams, Finite State Machines, Causal Chain, Data Flows and Sequence Diagrams, etc;
- *Function-based Behavior Model* is a data model that represents response to excitations based on the transformation of inputs into outputs by functions (activities) including the ordering and triggering of functions;
- *State-based Behavior Model* is a data model that represents response to excitations in the digital approximation based on state of an object, transitions between states and actions or activities launched by the states or transitions;
- *Requirements* is a data model that captures requirements as text strings with traceability, allocation, weighting and risk identified with each requirement [Text-based Requirements (TBR)] and that describes requirements as structured and quantified formalisms that may be decomposed from text-based requirements; can include tables, spreadsheets, graphs, charts, pictures and equations [Property-based Requirements (PBR)];
- *Allocation* is a data model that captures allocation relationships between requirements and functions;
- *Structural Models* is a data model that captures the organization of a system (e.g. how a system is built, static relationships between subsystems, components, or parts that constitute the system); describes what is designed, characterized built and maintained;
- *Risk Analysis* is a data model that identifies risk status, relationships, likelihood, consequence, impact, approach strategy, and contingencies;
- *Rules* are a data model that describes the information associated with constraints imposed on a product or process by system requirements, physical limitations and/or environmental restrictions. The model includes sufficient information to represent and/or support the constraint e.g. predicate calculus, the constraint life-cycle, constraint execution and associated meta-data, such as source, date and time, authorization, justification, description and notes;
- *Validation Model* is a data model that captures the information used to demonstrate that the emerging product is consistent with the stakeholder needs;
- *Verification Model* is a data model that captures the information used to demonstrate that the emerging product is consistent with the system requirements;
- *AP Interface Models* is a data model that provides interfaces between domain specific data

models such as mechanical, electronic, structural analysis, thermal analysis, manufacturing, etc. e.g. the transform between engineering analysis (AP209, STEP-TAS, STEP-NRF, AP235, etc.) and any AP233 module set;

- *Data Presentation* is a data model that provides a consistent set of presentation mechanisms and advanced schematics product model definitions that present the computer sensible model data (defined in representation model space) onto a human understandable schematic diagram (presentation space), conforming to conventional and/or future draughting standards;
- *Measurement* is a data model that includes information associated with the product development process quantification and its control and optimization.

2.4. Systems Engineering Ontology

Collaboration and sharing of models requires the use of common terminology with well-defined meaning. For collaboration, the meaning of the models needs to be expressed without the models having to be accompanied by subject domain experts. Thus, for concepts like part of and product version, the informal meaning and even natural language definitions within standards are not sufficiently precise to rule out different interpretations for the same term (Beca et al. 2011).

Ontologies, which make these concepts precise, address this problem. Following very simple modelling principles, it uses classes, properties and relationships to describe and represent an area of knowledge defining the concepts and their relationships (Ehrig & Staab 2004). This way it is acceptable to conclude that it enables the creation of a kind of *Lingua Franca* for common understanding and exchanging (R. Jardim-Goncalves, C. Agostinho, et al. 2011).

Because of this matter, it was normal to implement ontologies in the system engineering with the intention to make implicit design artefacts explicit, such as ontologies representing process or service vocabularies relevant to some set of components (OMG 2009) and this will facilitate the ability to quickly develop models and to make their sharing easier.

The ontologies are used throughout the enterprise system development life cycle process to augment and enhance the target system as well as to support validation and maintenance. This is important because automated reasoning can mitigate engineering tasks that are currently manual, error prone, and time consuming.

2.5. Open Research Issues

Despite having two standard models languages associated to MBSE, frequently enterprises remain using traditional models for engineering analysis and simulation models such as CAD, CAE, reliability, costing, PLM, etc. This raises an interoperability problem to MBSE collaborative networks, since without using SysML or AP233 to integrate the four models needed in SE, the study of how models interact with each other throughout the system LC to benefit stakeholders remains an open issue. In fact, the MBSE Modelling and Simulation Interoperability (MSI) Challenge Team is active

looking for a solution to incorporate the diversity of models currently in use in SE (OMG 2011b). So far, the most common way to integrate models between different partners is to establish P2P mappings between the different models (Carlos Agostinho & Ricardo Jardim-Goncalves 2009).

Other issue is when the requisites of a product is changed to satisfy the customer needs, every time this happened the other models will suffer modifications causing problems on the level of the interoperability. To an enterprise this is an unbearable problem that they want to avoid, since this cause a delay in the LCP. This is a sustainable issue that needs to be solved, so one way is to monitor the models to detect evolutions and with time propose solutions.

3. ENTERPRISE SYSTEMS INTEROPERABILITY

Enterprise Systems (ES) are large-scale, integrated application-software packages that use the computational, data storage, and data transmission power of modern Information Technology (IT) to support processes, information flows, reporting, and data analytics within and between complex organizations. The integrated content may then be used to provide a configuration management solution throughout the life cycle in relation to the products, assets, processes and requirements of the entity (laboratory, facility, SDD, etc.) (US Department of Energy 2011). An example of this integration is the MBSE that formalize applications to support the systems along the PCL.

Looking to this description, ES appear to be a dream come true, since commercial software packages promise the seamless integration of all the information flowing throughout the company. But if an organization has two systems and they cannot communicate with each other, then this will bring some problems in the manufacturing productivity and customer responsiveness suffers. This makes an organization's systems to become fragmented, meaning the business is also fragmented (Davenport 1998).

This is a problem for the organizations, a break in the systems that will difficult the efficiency and cooperation. Rather, the organizations need the ability for a system or a product to work with other systems or products without special effort on the part of the customer, this is what IEEE (Geraci et al. 1991) defines as interoperability. The IEEE also defines interoperability as the “ability of a system or a product that works with other systems or products without special effort on the part of the customer” (Geraci et al. 1991).

The lack of interoperability can be a great issue to the enterprises, so the modern enterprises have to be interoperable in terms of not only their IT systems, but also their business processes, their applications and even their human resources. Enterprises are concerned with interoperability between organisational units or business processes either within a large enterprise or within an enterprise network. The challenge lies in facilitating communication, cooperation, and coordination among these units and processes (Kotze & Neaga 2010), (Ray & Jones 2006). Because of these matters, this ability is gaining an important position in our days, since the organisations applications and software systems need to be interoperable in order to achieve seamless business across organisational boundaries.

In this section different maturity levels that categorize interoperability in different layers are presented, helping to distinguish the different problems and position the several interoperability frameworks available.

3.1. Interoperability Layers and Maturity Levels (IL & ML)

In the past few years, several authors have presented different solutions to help achieve interoperability. Normally the interoperability types follow a scale of advancement, where the higher a type is placed in the scale, the more advanced is the interoperability maturity achieved. For this reason the interoperability types are sometimes called levels, where to reach an upper level of

interoperability, all the previous levels have to be successfully addressed (Peristeras & Tarabanis 2006).

Maturity levels describe the stages through which they are defined, implemented, and improved. They have the objective of providing a guide to select improvement strategies by determining the current capabilities of specific processes and identifying the issues most critical to quality and process (C4ISR 1998).

A presentation of six well known interoperability typologies, are presented in the next sub-sections.

3.1.1. Connection, Communication, Consolidation, Collaboration Interoperability Framework (C⁴IF)

The C⁴IF was presented in (Peristeras & Tarabanis 2006) is a framework that use the basic linguistics concepts to the domain of Information System (IS) communication. With this concepts it was define four interoperability types in the framework, which are represented in the Figure 3-1 and described above:

- *Connection* refers to the ability of information systems that exchange signals. To succeed in this, a physical contact/connection should be established between two (or more) systems;
- *Communications* refers to the ability to exchange data in IS's. To succeed in this, a predefined data format and/or schema need to be accepted by the interlocutors;

In this type at least two levels of communication exist, the first level is the exchange based on a commonly accepted data format, the second level is more advanced, and the exchange includes data;

- *Consolidation* refers to the ability of IS's to understand data. To succeed in this, a commonly accepted meaning for the data needs to be established between the interlocutors;
- *Collaboration* refers to the ability of systems to act together. Action results in changes in the real world. To succeed in this, a commonly accepted understanding for performing functions / services / processes / actions needs to be established between the interlocutors or IS's. One of the greatest advantages is that the three areas are considered disjoint;

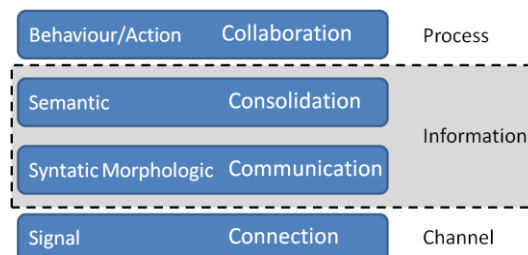


Figure 3-1 - C⁴IF Framework (Peristeras & Tarabanis 2006)

Another concept presented in this work, is the three demarcated areas where the four interoperability types are organized. The three areas are:

- *Channel* refers to the connection layer and the ability of IS's to exchange signals;
- *Information* refers to the communication and the consolidation layers, and the ability of IS's to exchange data and information;
- *Process* refers to the collaboration layer and the ability of IS's to act together.

3.1.2. Levels of Information System Interoperability (LISI)

An early classification was defined in the Levels of Information System Interoperability (LISI) (C4ISR 1998), which focuses on assessing systems against increasing levels of sophistications that focus in exchanging and sharing information and services through the system's life cycle. This occurs through five layer represented in Figure 3-2 and that are described below:

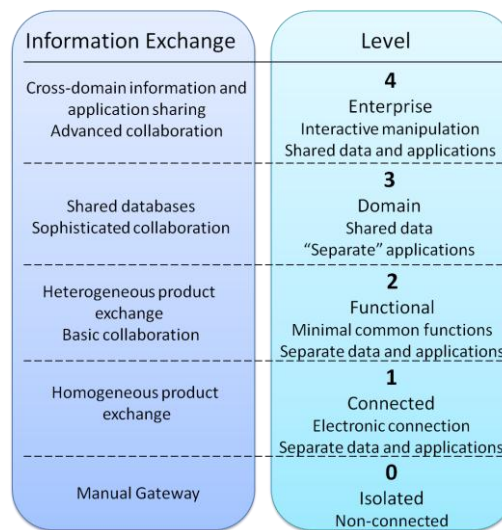


Figure 3-2 - LISI interoperability maturity model (C4ISR 1998)

- *Level 0 - Isolated Interoperability in a Manual Environment.* These systems cover the wide range of isolated, or stand-alone, systems. Direct electronic connection is not allowed or is available, so the only interface between these systems is by manually re-keying or via extractable, common media;
- *Level 1 - Connected Interoperability in a Peer-to-Peer Environment.* These are capable of being linked electronically and providing some form of simple electronic exchanges, with a certain limitation. They are capable of passing homogeneous data types, such as voice, simple e-mail, or fixed graphic files such as GIF or TIFF images between workstations;
- *Level 2 - Functional Interoperability in a Distributed Environment.* These systems reside on local networks that allow data sets to be passed from system to system. There is an increasing of complexity in media exchanges, with the use of the formal data models;
- *Level 3 - Domain-Based Interoperability in an Integrated Environment.* These systems are capable of being connected via wide area networks (WANs) that allow multiple users to access data. A domain-based data model is present (logical and physical) that is understood, accepted, and implemented across a functional area or group of organizations

that comprises a domain. To express the increase of capabilities between the levels, LISI presented the terms PAID - the Procedures imposed by information management, the capabilities of Applications that act on that data, the type of Infrastructure required, and the nature of data transferred;

- *Level 4 - Enterprise-Based Interoperability in a Universal Environment.* These systems are capable of operating using a distributed global information space across multiple domains. Multiple users can access and interact with complex data simultaneously, and this data is shared and distributed throughout the system.

3.1.3. ATHENA Interoperability Framework

From 2004 to 2007, the ATHENA Interoperability Framework was developed with the aim to adopt an holistic perspective on interoperability. This framework (Figure 3-3) is prepared to analyse and understand the business needs and the technical requirements addressing interoperability across business, knowledge, application, and data layers, and envisages integrating in three ways: Conceptual, Technical and Applicative (Athena IP 2004), (ATHENA 2010), that are described below:

- *Conceptual Integration* which focuses on concepts, metamodels, languages and model relationships. It provides us with a foundation for systemising various aspects of software model interoperability;
- *Technical Integration* which focuses on the software development and execution environments. It provides us with development tools for developing software models and execution platforms for executing software models;
- *Applicative Integration* which focuses on methodologies, standards and domain models. It provides us with guidelines, principles and patterns that can be used to solve software interoperability issues.

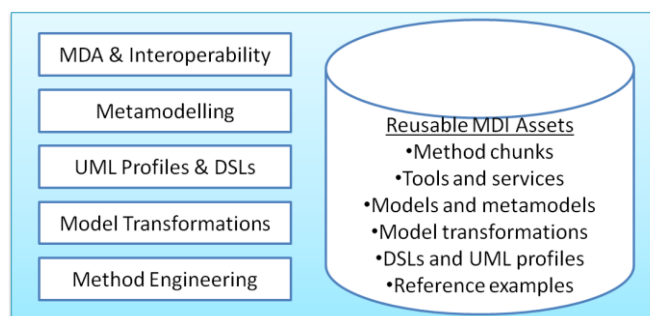


Figure 3-3 - ATHENA MDI Framework (ATHENA 2010)

3.1.4. European Interoperability Framework

The European Interoperability Framework (EIF) is developed and maintained by the IDABC⁴ and ISA⁵ programmes (ISA 2010). The EIF is concerned with interoperability in the very specific context of the provision of European Public Services (EPS), in other words, for EIF the EPS means a "cross-border public sector service supplied by public administrations, either to one another or to European businesses and citizens by means of cooperation between those administrations".

The EIF is part of a set of interoperability initiatives aiming at providing support to the establishment of EPS, with this in mind, the EIF presents four interoperability levels that are represent in Figure 3-4 and described below:

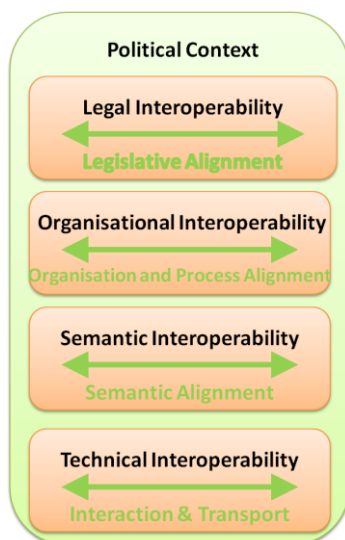


Figure 3-4 - EIF Framework (ISA 2010)

- *Political Context* is the establishment of a new EPS in the result of a direct or indirect action at political level. The objective is to facilitate cooperation between public administrations, in order to be effective, this is achieved by sharing visions, agree on objectives and align priorities;
- *Legal Interoperability*, one of the biggest problems in the EPS is that each one works within its own national legal framework. Causing problems of interoperability, and causing difficulties to working together or sometimes impossible. So, to avoid this problem, when exchanging information between Member States, the information must be maintained across borders and the data protect legislation in both originating and receiving countries must be respected;
- *Organisational Interoperability* is how the organisations collaborate to achieve their mutual agreed goals. This is divided in two main points, the Business process and the related exchange of information;

⁴ IDABC: Interoperable delivery of pan-European eGovernment services to public administrations, businesses and citizens

⁵ ISA: Interoperability solutions for European public administrations

- *Semantic Interoperability* enables organisations to process information from external sources in a meaningful manner. It is this level that controls the exchange of information, and ensures that it is understood and preserved through the different exchanges. To aim the interoperability the semantic interoperability proposes two points that are the minimum requisite, they are: Agreed processes and methodologies for developing semantic interoperability assets; Sector-specific and cross-sectoral communities to agree on the use of semantic interoperability assets at EU level;
- Technical Interoperability is the last level, and covers the technical aspects of the information systems. Aspects like the interface specifications, interconnection service, data integration services, data presentation and exchange, etc.

3.1.5. Interoperability Classification Framework

In (Panetto 2007) Panetto propose a maturity level, with six kinds interoperability solutions, represented in Figure 3-5:

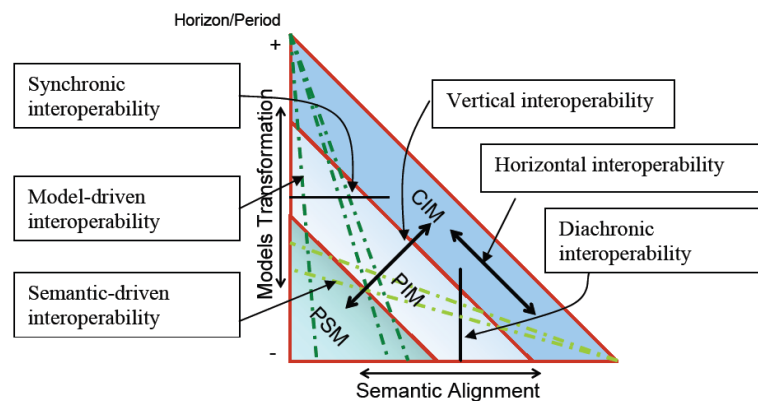


Figure 3-5 - Interoperability Classification Framework (Panetto 2007)

- Synchronic - Are issues where applications exchange models defined by compatible languages (the same syntax) but with different semantics, in a synchronous way;
- Model-driven – This level is about technologies (or standards) to solve model syntactic transformations;
- Semantic-driven – The developments where the semantic alignment is the main issue;
- Vertical - when exchanging models from different abstraction levels, this exchange process from one application to another involves models transformations (syntactic) and semantic alignment (also called concept mapping);
- Horizontal – when an applications' interoperability problem may occur when exchanging models at the same abstraction level (CIM - Computer Independent Model, PIM - Platform Independent Model or PSM - Platform Specific Model) , this exchange process from one application to another involves models transformations (syntactic) and semantic alignment (also called concept mapping);

- Diachronic - Are issues when applications interoperate over time by exchanging models referring to different views of the same product. In this case, models have compatible semantics but need to be syntactically transformed before being exchanged. This allows streamlining model management and creating a true information management system.

3.1.6. Interoperability Practices Pyramid

Finally, in 2009, Agostinho and Jardim-Goncalves (Carlos Agostinho & Ricardo Jardim-Goncalves 2009) proposed another approach towards interoperability levels classification that divides the interoperability types into five layers, as presented in Figure 3-6:

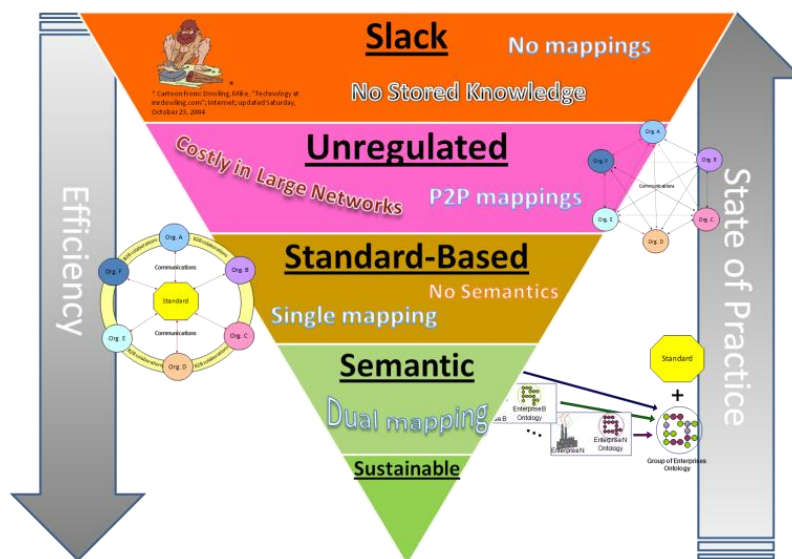


Figure 3-6 - Interoperability Practices Layers (Carlos Agostinho & Ricardo Jardim-Goncalves 2009)

- Slack interoperability* – When there is no previous understanding between the sender and receiver. They have rudimentary communication methodologies with little support of advanced IS, e.g. ERP - Enterprise Resource Planning;
- Unregulated interoperability* - Organisations work with peer-to-peer relationships, and maintain their own data format and business rules mediated by, as many mappings as the number of business partners;
- Standard-based interoperability* - Several dedicated reference models covering many industrial areas are based on common collaboration models;
- Semantic interoperability* – Athena (Athena IP 2004) identified semantics as a cross-cutting issue along the four levels of interoperability within an enterprise since a system might be able to exchange data with another but still not be fully interoperable. This layer is therefore based on the previous and complements the reference model with reference knowledge (e.g. ontology) so that the content of the information exchanged is annotated (Missikoff et al. 2003), (J. Sarraipa et al. 2007) and unambiguously defined: what is sent is the same of what is understood;

- v. *Sustainability Interoperability* - The simplest way to describe the term "sustainability" in this context is that it is related to the aim of improving the quality of service by contributing to a more robust interoperability, avoiding excessive consumption of resources (e.g. manpower and time), when the dynamicity of system and networks causes harmonization breaking. It is a novel concept whose objectives are to reconcile the economics interests ensuring that network maximizes its efficiency by remaining interoperable at most times in any of the 3 previous levels (exception for slack interoperability).

3.1.7. Outlook in IL & ML

All the presented interoperability layers and maturity levels have the purpose to evaluate the interoperability status inside an organisation or specific system or network. It is relevant to know, in terms of interoperability, the position one is in, so that it can improve and become more efficient when exchanging information with its business partners. This thesis can be a facilitator for companies to improve their systems.

To complement that, some of the technologies mentioned in the many maturity topics have been further evaluated (e.g. MDI) and presented in the following sections, since they could be used for implementation purposes. Panetto (Panetto 2007) described a maturity level that recognizes the Model-Driven Interoperability (MDI) method as a major enabler for enterprise interoperability. Due to its great potentialities, it was decided to it would be a subject for study, to verify whether it is relevant to this work. Moreover, the EIF suggests a level that uses a semantic interoperability, which also has many implications in today's systems, thus it was decided to contemplate that technology in section 3.3. Finally it was also considered important to look upon the model morphisms (MoMo) used in the work of Agostinho and Jardim-Goncalves (Carlos Agostinho & Ricardo Jardim-Goncalves 2009). In section 3.4, the MoMo concept is explained and in section 3.5 the CAS-SIF framework is described since it is a clear advancement of the state of the art, and the maturity level where this thesis can be of more assistance.

3.2. Model-Driven Interoperability

The Interoperability or the ability of a system or a product to work with others systems is gaining an important position in our days, since the organizations applications and software systems need to be interoperable in order to achieve seamless business across organisational boundaries. In the last decades appear a lot of solutions to help to improve interoperability, however full interoperability is not still achieved (Bourey et al. 2007), (Elvesæter et al. 2006).

To achieve interoperability a need to considerate the interaction in all the layers of the organisations is required, that is the business, knowledge, and ICT (Information and Communication Technology) levels and semantics can also be used. To help in this matter it was introduce a model-driven that approaches to generate software, this provides many advantages by improving portability, interoperability and reusability through the architectural separation of concerns. With this, it was born the Model-Driven Development (MDD), in particular the Model-Driven Architecture (MDA) (Bourey et al. 2006).

Several projects to provide a better way of addressing and solving interoperability issues using semantic annotation and ontology were created. ATHENA and INTEROP define an interoperability framework for MDD of enterprise applications and software systems to address interoperability (Xu et al. 2009).

3.2.1. Model-Driven Engineering

Model-Driven Engineering (MDE) also known as Model Driven Development (MDD) is described in (Hailpern & Tarr 2006) as a software-engineering approach consisting of the application of models and models technologies to raise the level of abstraction at which developers create and evolve software.

With the advantage of expressing the models using concepts that are much less bound to the underlying implementation technology and are closer to the problem domain relative to most popular programming languages. This makes the models easier to specify, understand, and maintain. Other advantages of MDE is the standardization, that provides better practices, enables and encourages reuse, and facilitates interworking between complementary tools, and the programs are automatically generated from their corresponding models. And the most important issue that the MDE help is that it tools can perform a model checking that can detect and prevent many errors early in the life cycle (Bran Selic 2003), (Schmidt 2006).

Since the 1980s there is the attempt to create technologies that further elevates the level of abstraction used to develop software. Several attempts were made to create and implement a software engineering to help in this matter, but for different reasons none of them were widely adopted in practice. Using that past experience, the existing and emerging MDE technologies learned the lesson and developed higher-level platform and language abstractions (Schmidt 2006).

3.2.2. Model-Driven Architecture

The OMG introduced the Model-Driven Architecture (MDA) that was created with the intention to support the MDE in the specification of software systems based on a model transformation concept, and with the ability to address the complete development lifecycle, covering analysis and design, programming, testing, component assembly as well as deployment and maintenance. It was created with the objective to accomplish three main goals; portability, interoperability and reusability (ATHENA 2010), (Truyen 2006), (Petzmann et al. 2007).

With these in mind, we can say that one of the great advantages of the MDA is the standardized techniques that are used, like the Unified Modelling Language (UML), the Meta Object Facility (MOF), the XML Metadata Interchange (XMI) and the Common Warehouse Meta Model (CWM) (Truyen 2006), (Petzmann et al. 2007).

Since the technology that is in focus today, tomorrow may be obsolete, this provokes a constant change and the emergence of new platforms and technologies. Because of this problem, the MDA enables the rapid development of new specifications that leverage them, and reduce the process of their integration. This makes the MDA a comprehensive, structured solution for application

interoperability and portability into the future (Truyen 2006).

The manner provided by MDA to solve this situation, is to define an approach and the standards to support it, allowing the same model specifying system to be realized on multiple platforms through auxiliary mapping standards, and it also allows different applications to be integrated by explicitly relating their models, so the MDA describes how to transform these models step by step from an independent system level to platform models. These models and the transformation of these models are focused by the MDA approach, and they can be more accurately be described as layers of abstraction, since with the three layers a set of models can be constructed, each one corresponding to a more focused viewpoint of the system, next is a description of three presented models (Figure 3-7) (Truyen 2006), (Petzmann et al. 2007), (Miller & Mukerji 2003), (Miller & Mukerji 2001):

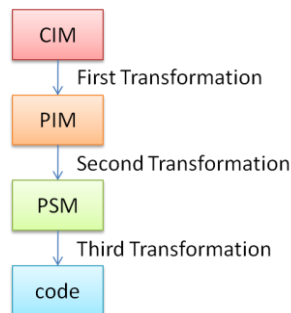


Figure 3-7 - The levels of MDA approach (Petzmann et al. 2007)

- Computation Independent Model (CIM) is where the requirements for a system are designed / modelled, and it helps in presenting exactly what the system is expected to do. It is useful, not only as an aid to understanding a problem, but also as a source of a shared vocabulary for use in other models;
- Platform Independent Model (PIM) is where the view of a system from the platform independent viewpoint is designed / modelled. The goal is producing models, which can be transformed in an arbitrary system platform;
- Platform Specific Model (PSM) is a view of specific platform. The PSM merges the specification of the PSM with the specific details of a particular platform.

In the end the MDA software engineering approach has to produce a code which can be run on specific platforms. This code can do the transformation from one level to another, and they have to be the more automatic as possible. Another feature that is presented in MDA and is important in the transformations of the models is the notion of mapping that is used to modify one model in order to get another model, later in this chapter it will be discussed.

3.2.3. Model-Based Systems Engineering

Since the MBSE have the aim to formalize application of modelling to support the systems engineering processes and these models go from requirements to validation of activities, reaching all the LC of the product. From times to times, the processes suffer some modifications, needing to adapt part of the models. Using the MDE to describe these models will help to create an interoperable

system, easier to adapt and fast.

3.3. Semantic Interoperability

The IEEE defines semantic as relationships of symbols or groups of symbols to their meanings in a given language (Geraci et al. 1991). Each year semantic problems are becoming increasingly important, as a result of a world more and more global, with enterprises being geographically distributed all over the world and needing to cooperate with each other (UKOLN 2006). This cooperation requires an exchange of data, so different information systems need to communicate and seamless interoperate.

In answer to this requirement the semantic interoperability appears, that the Ojo et al. described in (Ojo et al. 2009) as the capability of organisations in public, private voluntary and other sectors, and their information system to:

- 1) Discover required information;
- 2) Explicitly describe the meanings of the data they wish to share with other organisations;
- 3) Process received information in a manner consistent with intended purpose of such information.

From this definition, several semantic interoperability challenges arise, in particular: the need to integrate information across agencies to support strategic decision making, delivery of seamless services (Ojo et al. 2009). The provision of this semantic information and the mapping process determines the degree of semantic coherence in a given service. Consequently, there are different levels of semantic coherence or interoperability (UKOLN 2006).

3.4. Model Morphisms

The concept of morphism is described in mathematics as an abstraction of a structure-preserving map between two mathematical structures (Ogren 2000). Recently, this concept is gaining some meaning in computer science, more exactly in systems interoperability. This new meaning of Morphism describes the relations (e.g. mapping, merging, transformation, etc.) between two or more IS specifications.

In pursue of achieving a sustainable interoperability solution, authors address morphisms, namely mappings representation to reach an interoperable state between enterprise systems. They specify the relation between two information models, either structural or conceptual as is the case of ontologies. According to the pyramid-like interoperability levels explained before (Figure 3-6), the concept of mapping is used in the four more efficient layers:

- Unregulated interoperability uses P2P mappings and can evidence as many as the number of partners, the number of mappings required are represented by the Equation 1, where x is the number of partners;

$$\forall x \in \mathbb{N} : \text{map}(x) = \begin{cases} 0, & x = 1 \\ \text{map}(x-1) + (x-1), & x > 1 \end{cases} \quad \text{Equation 1}$$

- Standard-based interoperability uses a single mapping between the internal system model of one of the partners and the standardised model being used as the reference for communications within the network, (Equation 2, where x represents the quantity of partners);

$$\forall x \in \mathbb{N} : map(x) = x \quad \text{Equation 2}$$

- Semantic interoperability uses a dual mapping, this happened because in addition to the morphism of the standard-based interoperability, a map with the organisation's semantics to domain ontology is also needed (Equation 3, where x represents the quantity of partners);

$$\forall x \in \mathbb{N} : map(x) = 2x \quad \text{Equation 3}$$

Sustainability Interoperability is implemented on top of any of the other layers, these results in the sustainability to maintain all the mappings described in each of the layers.

3.4.1. Models, Languages and Meta-Models

In nowadays in the engineering disciplines it is usual to use models to design complex systems. Since the constructing of a structure as complex as a bridge or an automobile, it is almost impossible to be made without first constructing a variety of specialized system models. The use of models are important because they aim to reduce the risk by helping to understand a complex problem and its potential solutions before undertaking the expense and effort of a full implementation. Thus, it is obvious that the software systems are the most benefited with the using of models and modelling techniques (B Selic 2003).

So, the modelling is the process of converting our perceived views of reality into a representation of it and the meta-modelling is the process of specifying the requirements to be met by the modelling or establishing the specification which the modelling process must fulfil. As shown in the Figure 3-8, the model implies that the modeller abstracts properties from things in order to obtain a representation of the physical world. This process of abstraction can be applied to modelling itself, to obtain a model of the modelling process which it is called a meta-model. The meta-model defines concepts of a given domain, their relations, and forms a modelling language used to create executable domain models (Smolik 2000), (Van Gigch 1991).

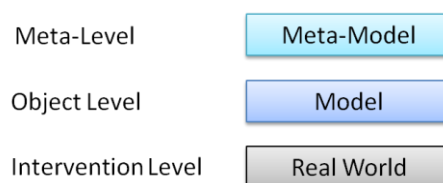


Figure 3-8 - The relationship between model and meta-model (B Selic 2003)

The main goal of the meta-model is to specify what is important to define in a model about the system. Since it is the meta-model that specify what elements may be contained in the model and how they relate among each other. When used in real world, meta-model is a specification of a domain

specific modelling language, a language used to express the requirements on the system and define how exactly it should realize them (Smolik 2000).

To help with the models, it was decided the standards for the MDE, this is important because the standardization provides developers with uniform modelling notations for a wide range of modelling activities. So, the UML becomes the standard of MDE, along with several technologies related to modelling. The choice of UML to be the key of MDD is because the UML is scalable to large software sublanguages, modular structure for easier adoption, increased semantic precision and conceptual clarity and represents a suitable foundation for MDA (B Selic 2003) (Sacevski & Veseli 2007).

3.4.2. Morphism Types

Two types of morphisms exist, the ones that modify the operand, and the ones that do not. The first one receives the classification as model-altering, and the second one the non-altering, below is a description of this two classification based on (Carlos Agostinho, Joao Sarraipa, et al. 2007), (Carlos Agostinho et al. 2010):

- *Model non-altering morphism* is supported in the traditional model-mappings, in this type of morphism changes to the source models are not applied. The relationship is identified among two or more existing models, pointing similarities and convergences;
- Model altering morphisms use a kind of a transformation function to transform the source model (operand) in a set of rules (operator), thus modifying the targeted output. The model altering can be divided in two categories, the model transformation and the model merging. The model transformation happened when a source model (A) is modified by a function F in order to obtain a new output model (B) expressed, or not, in the same language. The other one, that is the model merging, is when the input for the model transformation are multiple models, maintaining all original semantics from the input models.

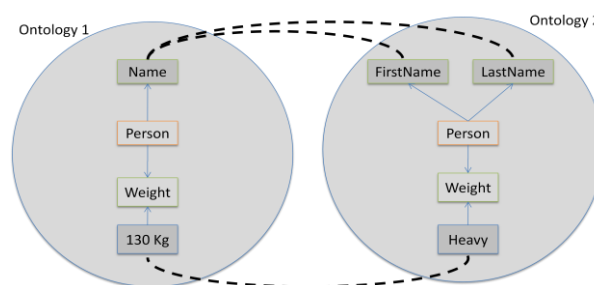


Figure 3-9 - Example of mappings between two ontologies

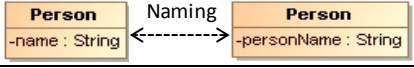
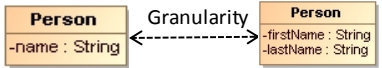
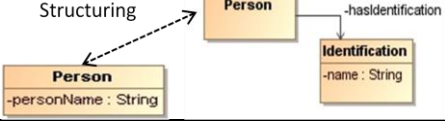
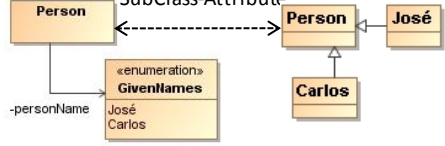
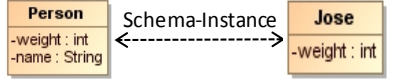
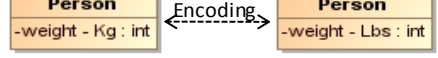
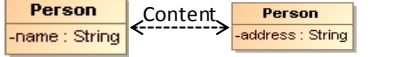
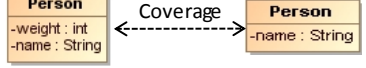
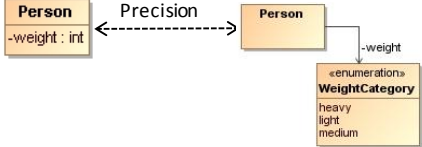
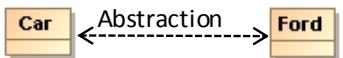
3.4.3. Model Mappings

Given two ontologies O1 and O2, Ehrig and Staab (Ehrig & Staab 2004) describe ontology mapping as the relationship each entity (concept C, relation R, or instance I) in ontology O1, has with a corresponding entity with the same intended meaning, in ontology O2. Or by definition a mapping is a relation between two models. This relation is made one meta-level higher than the transform, so the

mapping describes the rules used for the transformation (ATHENA 2010). In most of the cases the ontology has a higher entity, and above that one is represented by the relating entities and attributes, these morphisms that are related with a higher morphisms are called the sub-morphisms of that one, as an example is in the Figure 3-9 where the Person is the higher morphisms, and the relating entities are the Name and Weight.

In Figure 3-9 both information structures represent ontologies to describe a person. As an example of potential mapping difficulties, when it is needed to establish a relation between the two entities, conflicts between them can occur. In this case, one is about the name since in the second ontology it is divided in first and last name, the other is about the weight, where the second ontology is not very precise. Because of these issues, it is important to complement mappings with specific knowledge that identifies whether is they are complete, and if not identify the differences as better as possible.

Table 3-1 - Semantic Mismatches (based on (Carlos Agostinho et al. 2011))

Mismatch	Description	Examples
Lossless	Naming	Different labels for same concept of structure 
	Granularity	Same information decomposed in or composed by (sub)attributes 
	Structuring	Different design structures for the same information 
	SubClass-Attribute	An attribute, with a predefined value set represented by a subclass hierarchy (or vice-versa) 
	Schema-Instance	An attribute value in one model can be a part of the other's model schema (or vice-versa) 
	Encoding	Different formats of data or units of measure 
Lossy	Content	Different content denoted by the same concept 
	Coverage	Absence of information 
	Precision	Accuracy of information 
	Abstraction	Level of specialisation 

3.4.4. Semantic Matchmaking

When a mapping is created between two models, sometimes some inconsistencies of information will appear derived from the multiple conflicts between entities. Those are called Semantic Mismatches and can be classified as either lossy or lossless, as shown in Table 3-1.

With lossless mismatches, the relating element can fully capture the semantics of the related, while in the lossy mismatches a semantic preserving mapping to the reference model cannot be built (C. Agostinho, Malo, et al. 2007).

Due to this mismatch problem morphisms should be represented using formal expressions, facilitating a sustainable interoperability through the creation of intelligent systems able to reason, deduce and recommend mapping readjustments, as well as change the mismatch type when system requirements change (Carlos Agostinho & Ricardo Jardim-Goncalves 2009), (R. Jardim-Goncalves, C. Agostinho, et al. 2011).

3.4.5. Knowledge Enriched Tuple for Mappings Representation

The research community has developed many proposals to morphisms representations (InterOP NOE 2005), (OMG 2006). Graph theory has been used in some, although other theories can be considered to achieve the envisaged goals, e.g., set theory (Dauben & Cantor 1979), model management (Bernstein 2003), or semantic matching (J. Sarraipa et al. 2010). However there is not a single perfect solution that can be used to achieve all the morphisms goals at once. Some are ideal for structural issues, others for semantics providing good human traceability, and others are more formal and mathematical based. Agostinho et al. (Carlos Agostinho et al. 2011) proposes the usage of a 5-tuple mapping expression (Equation 4), with the objective to consolidate existent approaches to morphisms:

$$\text{MapT} = \langle ID, MElems, KMType, MatchClass, Exp \rangle \quad \text{Equation 4}$$

- *ID* is the unique identifier of the MapT;
- *MElems* is the pair (a,b) that indicates the mapped elements in the source and destination models;
- *KMType* stands for Knowledge Mapping Type, and is used to identify the morphism as “Conceptual” if mapping concepts or terms; “Structural” if mapping model schemas; or “InstantiableData” if the mapping instantiable properties;
- *MatchClass* stands for Match/Mismatch Classification and is used to classify with reference data, knowledge about the mapping mismatches, i.e., inconsistencies of information that can appear when a mapping between two models is created, derived from the multiple conflicts between the entities (as the one illustrated in Figure 3-9);
- *Exp* stands for the mapping expression that translates and further specifies the previous tuple components.

The idea of using a tuple brings many advantages, like being human traceable and readable,

adding knowledge concerning mismatch and when used by intelligent systems, the tuple's information enables automatic data transformations and exchange between two organizations working with/on different information models. Because of these advantages it was decided that the tuple would represent morphism.

According to the tuple philosophy, all the information about the mappings should be stored in a dedicated Knowledge Base (KB) so that it becomes computer processable, and readjustments can be easier to manage and data exchange re-established automatically in a sustainable environment. To reach these objectives Sarraipa et al. (J. Sarraipa et al. 2010) proposed the Communication Mediator (CM) to carry this task, and also proposed that all the business partners in the same Collaboration Network (CN) it embedded in their local system. An extension towards lifting the work of this dissertation is proposed and explained on section 4.3.2.

3.5. CAS-Based Framework to Support Sustainable Interoperability (CAS-SIF)

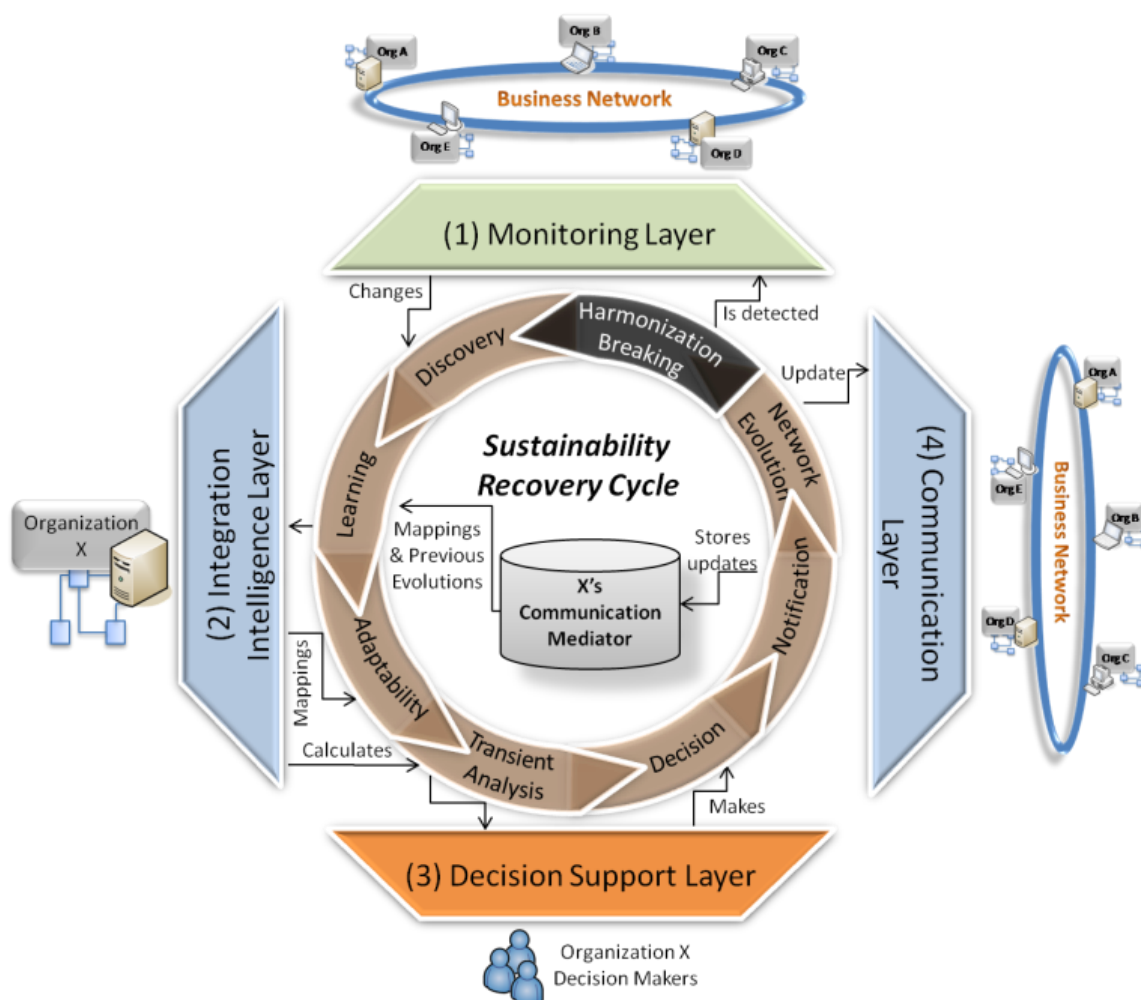


Figure 3-10 - CAS-SIF Framework (Carlos Agostinho & Ricardo Jardim-Goncalves 2009)

In section 3.1 several interoperability layers and maturity levels are explained, but this dissertation is about the last one, to implement a sustainability interoperability framework to demonstrate the interoperability layer proposed by Agostinho and Jardim-Goncalves (Carlos

Agostinho & Ricardo Jardim-Goncalves 2009), the same authors propose the framework that they called CAS to support sustainable Interoperability (CAS-SIF). The goal is to provide systems the capability of self-recognizing in the face of environmental (i.e. the market) adversities. However to avoid falling into chaotic behaviour, CAS-SIF uses an adaptive intelligence at both the network and systems level. In Figure 3-10 the framework is represented and decomposed into four layers:

- **Monitoring** - The monitoring layer addresses different stages, from capturing information to its analysis, and is structured into specific components in order to meet a set of requirements. The whole process has to be carried out balancing extensibility and self-description capabilities with compactness. When the interoperability is established by the organisation, they demonstrate stability by exchanging e-messages following established laws. But sometimes small fluctuations acting on a system may cause it to cross a critical point and evidence an unexpected behaviour, designated as harmonization breaking. In Figure 3-10 one can see that this is detected by the monitoring layer and then it is responsible for analysing it, discovering the changes that caused the anomaly;
- **Integration & Interoperability Intelligence** - After detecting the harmonisation breaking, a learning process should be triggered to learn more about the changes that have occurred and the nodes adaptation required. CAS-SIF enables the adaptation of systems and the optimisation of the maintenance process through dynamic model morphisms, using knowledge representation technologies applied to the model management domain, such as the tuple described before. As in Figure 3-10, each companies Communication Mediator provides information about similar mapping and previous evolutions that have occurred so that the system can learn and propose, in the adaptability process a new or an updated mapping;
- **Decision Support** - Changes at the internal or interfaces structures of the organisation's information systems can lead to unexpected situations. In these cases the CAS-SIF must consider some kind of decision support to allow the manager or any other decision maker to take the final word regarding whether or not to execute the mapping proposed in the adaptation. As illustrated, this layer calculates a transient indicator to understand how the network will suffer from the proposed (re)adaptation, and support the decision;
- **Communication** - The CAS-based sustainable interoperability framework is meant to be implemented among the several network nodes that are adopting this level of interoperability. This way, this layer is responsible for the re-adaptation of a network node, and the communication to the entire business network in such a way that it causes minimal disruption to the other members of the network.

3.6. Open Research Issues

We are in an era of globalization where organisations are represented in different parts of the world, and there is a need to cooperate between them without any sort of problem that will create a lack in the interoperability. To help solve this issue, several authors presented different solutions, the interoperability layers and the maturity levels, despite having several propositions to solve the

interoperability issue, yet few authors address the subject of maintaining this status, and this remains an open issue.

After an evaluation of the different interoperability levels and maturity layers, in order to determine which one best suits the objectives of this dissertation, it was decided to use the interoperability level of Agostinho and Jardim-Goncalves (Carlos Agostinho & Ricardo Jardim-Goncalves 2009), since it proposes a solution to help achieve and maintain the interoperability in the organisations which still needs further specification and development, that is the main focus of this dissertation.

4. MULTI-AGENT SYSTEMS (MAS) TO SUPPORT SUSTAINABLE INTEROPERABILITY

In a system, interoperation should be addressed at different levels. Thus, the diversity, heterogeneity, and autonomy of software components, application solutions, business processes, and the business context of an enterprise must be considered. Following that trend, interoperability's definition has been revisited a number of times. In Enterprise Interoperability (EI), each organization retains its independence but has the capability to communicate and interoperate with other organisations. Building upon this, the reference research projects on EI, INTEROP and ATHENA, have enlarged this definition to a holistic perspective where EI can only be achieved when addressed covering all levels of an enterprise (business, knowledge, applications and data) (ATHENA IP 2007).

In some cases (e.g. collaborative product development), SE is a particular case of EI since it requires integration of all the disciplines and areas of expertise from different enterprises into a team effort forming a structured development process that proceeds from concept to production to operation and a latter disposal. It also considers both the business and the technical requirements of all customers with the goal of providing a quality product that meets all user demands (INCOSE 2011). Therefore, MBSE needs EI to describe and integrate the LC of a product or system.

However, besides the challenge of integration models from different disciplines, achieving that inside heterogeneous networks is still an ongoing challenge hindered by the fact that they are, intrinsically, composed by many distributed hardware platforms, software tools and ICT systems (Ray & Jones 2006). Such engineering models could be defined in different languages or semantics, thus morphisms are needed to describe and formalise the relations between them, and sustainability methodologies are needed to cope with market dynamics: manufacturing systems are constantly adapting to new market and customer requirements, thus answering the need to respond with faster and better quality production; new organizations are constantly entering and leaving networks, leading to a constant fluctuation and evolution of system models. All these factors are making interoperability difficult to maintain (Carlos Agostinho & Ricardo Jardim-Goncalves 2009).

Due to this fact, this dissertation propose the MIRAI (Monitoring morphisms to support sustainable Interoperability of enterprise systems) framework to monitor the systems' interoperability through the morphisms previously defined and stored on a company CM. MIRAI detects changes in the interoperable environment, proposing the user morphism re-adaptations in the advent of harmonization breaking, thus contributing with a solution for the sustainability interoperability layer identified in the section 3.1.6. The MIRAI is stored in each enterprise that compose a CN and every time that a change is detected in one MIRAI, it will trigger a warning to the others MIRAI's in the network to see if that change has an impact, avoiding with this problems in the future.

This framework is an ongoing implementation of CAS-SIF, and although prepared for all layers, the parts that are explained in this section are focused on the validation of the 1st and 2nd layers, which are liable for monitoring and proposing a (re)integration of organisations. The system is prepared so

that in the future it can be possible to implement all the layers, for example the 3rd layer is the decision support, and in this version it was implemented a contribution to this layer, presented in section 4.5.

4.1. MAS Overview

Inside the scientific community exist several definitions to describe an agent, Wooldridge and Jennings in (Wooldridge & Jennings 1995) distinguish two general usages of the term agent: the first is weak and the second is stronger. In this dissertation it will focus on the weak notion, since this notion is a part that is incontestable within the community:

- *Autonomy* - agents make decisions without human intervention, these decisions are made with some kind of control over their actions and internal state;
- *Social ability* - agents interact with other agents via an agent communication language;
- *Reactivity* - agents react to changes in their environment;
- *Pro-Activeness* - agents have their own goals and besides reacting, they are capable of initiative.

Taking in consideration these descriptions, in particular the features of social and reactivity, it is common that the agent is capable of interacting with other agents, humans, or with the surrounding environment. This brings something new to the software technologies, i.e., communication and teamwork between software, in this case, between agents, and this is called Multi-Agent System (F. Bellifemine et al. 2007).

The great advantage on using MAS for the implementation of CAS-SIF is that they are capable of cooperation, collaboration, negotiation, etc., and they understand each other via an agent communication language based on speech act (F. Bellifemine et al. 2007) thus avoiding agent's interoperability issues. Therefore, due to these features, MAS is being used in different areas, from industrial applications, to telecommunication and multi-robotic systems.

4.1.1. MAS as a technology for CAS-SIF

Today, most of the organisations are worried about how to be interoperable, and how to gain that status, that is an important step to achieve. Then it is important to maintain this status, and at this point something to monitor and look for changes in the system is needed, this is what consists the sustainable interoperability (explain in the section 3.1). Every time that exist a harmonization breaking, means that the interoperability status was compromised, and need to recover as soon as possible as required and without the consumption of more resources (any kind of them). To help in this matter the authors propose the using of the MAS to do the monitoring of the system as an implementation of CAS-SIF.

One of the advantages of MAS that helped in this decision was the description of Wooldridge (Wooldridge 2009), and he says that decentralised multi-agents are considered to be an added value in the monitoring services implementation, assuring organisations' independence. This is an important advantage, because the main point was to create a sustainable interoperability and assuring

organisations' independence by creating MAS interoperability, this allows agents to communicate across multiple systems and achieve their goals. This communication is made by a standard protocol created by FIPA⁶ (Foundation for Intelligent Physical Agents) to assure that services and the ontology are common between the systems, because of this and due to being a widely used agent-oriented middleware, JADE was used.

4.2. MIRAI Framework and Architecture

As illustrated in Figure 4-1, MIRAI has the objective of monitoring the existing mappings and model versioning's stored in each enterprise's CM and timely detect the changes in the morphisms, proposing to the user a possible solution and preventing a significant transient period where interoperability in the network is not assured. The detection is carried as soon as CM changes, triggering an agent to search model differences. Indeed, when a versioning on one of the MBSE models is detected, MIRAI triggers a warning and automatically proposes a new mapping morphism to the user. This new suggestion is based on the 5-tuple mapping expression proposed by the authors in (Carlos Agostinho et al. 2011) and it is used to describe the relationship among the different models that are used in the MBSE during the LC stages, regardless of the language used. Within this framework, a mapping is created to respond to the evolution, where according to the tuple MapT. After that, since the user might accept the proposal or not, the authors decided to endow MIRAI a learning process based on weights to help in the choice of MatchClass for the new mapping, and increasing intelligence over time, and it will be explained later.

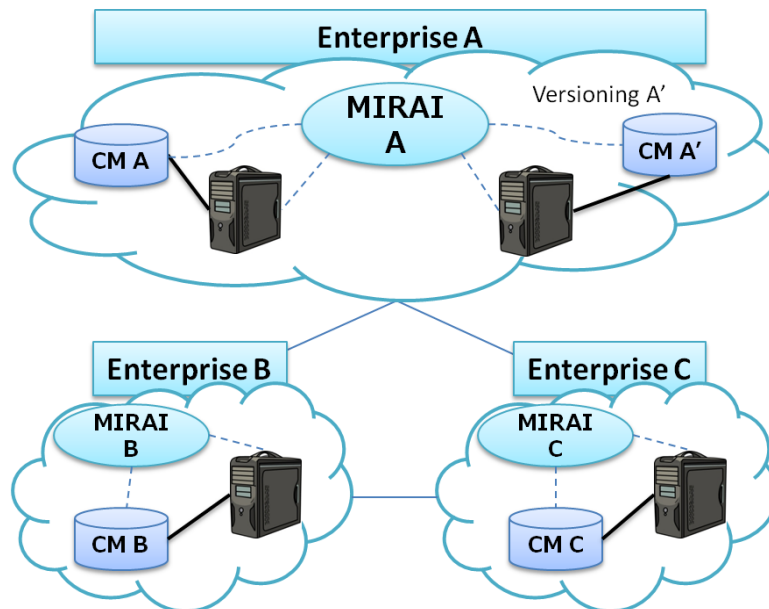


Figure 4-1 - MIRAI Network

There are various types of mappings, which MIRAI could propose (e.g. Structuring, Semantics,

⁶ FIPA: <http://www.fipa.org/>

Conceptual or Instantiable Data) to, relate such model elements and depending on the user choice, the proposal is accepted and is stored in the CM. The next time a similar situation occurs, MIRAI will provide a similar solution only if it (from the various possible mappings types) remains being the most weighted one according to user's choice pattern. For reaching such objectives, MIRAI is directly associated to each CM. Moreover, within the collaborative network of enterprises there will be a kind of sub-network, i.e. the MIRAI network (as in Figure 4-1) that enables to keep all CMs synchronized (interoperable) and maximizes the learning process as the whole distributed framework contributes with knowledge concerning user's selections, which is shared among the mediators.

To implement the framework that was explained until now, architecture was created. Keeping in mind four points to achieve the objective of this dissertation: Detection & Propose of the morphism, Communication between different enterprises, Monitor for dead agents and GUI to show the evolution of the system to the user. As a result it was created an architecture represented in Figure 4-2, divided in four important blocks that define the framework:

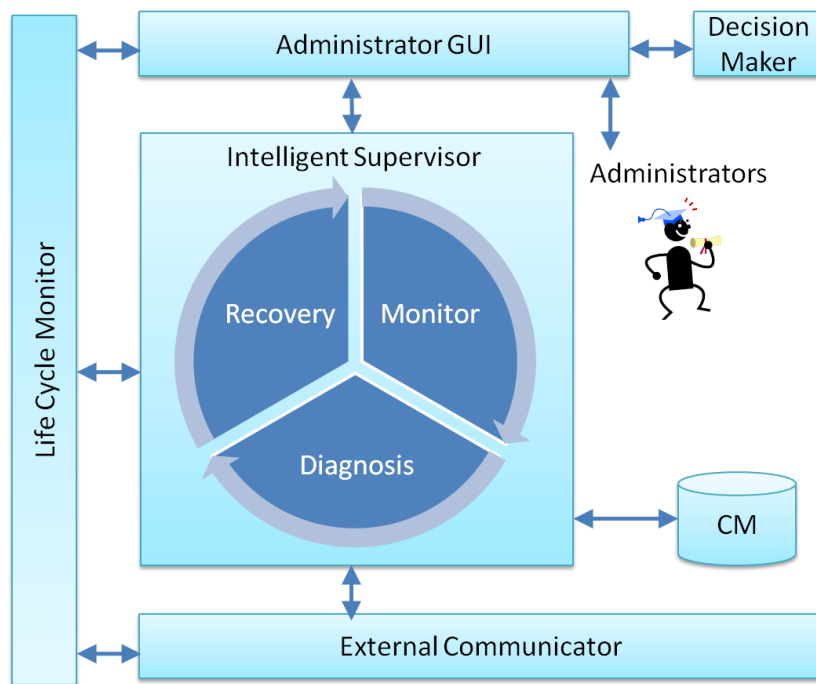


Figure 4-2 - MIRAI Architecture

- 1) *Intelligent Supervisor* is responsible for the detection of the harmonization breaks by performing a scan in the organization's CM in search for new morphisms or morphisms evolutions. These agents' block checks if the changes occurred have an impact in the system's interoperability status, and if it does, it proposes a possible solution. The Intelligent Supervisor follows the circle in Figure 4-2, where it starts with the Monitor the CM for changes, then do the Diagnosis of the problem, and in the end the Recovery of the problem, proposing a solution;
- 2) *External Communicator* which is responsible for the communication between different MIRAI's of a collaborative network;

- 3) *Administrator* is the interface between the MIRAI and the human user. In the future, it will be further developed into enclosing the decision support services envisaged in CAS-SIF, in the future the Administrator will have a new part. Called Decision Maker, which will help the Administrator in the decision of the new mappings, adapting to the choices of the user and learning with time. With that, the system will reduce the intervention of the user, making the system more independent and reliable;
- 4) *Life Cycle Monitor* which is a maintenance block to check if any agents died, and is responsible to resurrect them if required.

Following these description and as already explained, decentralised multi-agents are considered to be an added value in the monitoring services, thus it was decided that MIRAI would be composed with a six agents MAS divided by the four blocks:

- Agent Monitor Mediator and Agent MoMo belong to the Intelligent Supervisor;
- Agent User to the Administrator GUI;
- Agents Persistor and Agent Persistor Police to the Life Cycle Monitor;
- Agent Communicator the External Communicator.

4.3. Communication Mediator (CM)

In aim to grant a planned traceability to support intelligence and sustainability, it was required to store the morphisms in a parseable and structured knowledge-base. With this, every mapping between models or ontologies of business partners can be stored and accessed by their local systems. This allows communities to build systems with reasoning capabilities able to understand each other's representation format, without having to change their data and schema import or export processes (J. Sarraipa et al. 2010).

The proposed knowledge based (KB) is called Communication Mediator (CM) and is defined by ontology in OWL format. The knowledge is stored according to the tuple format, proposed in equation 4 and all the business partners in a collaborative network should have a CM in their local system, to act as a mediator for information exchange (Carlos Agostinho et al. 2011).

The structure of the CM is presented in Figure 4-3 and described as follows: the CM has two main classes: "Object" and "Morphism". The "Object" represents any "InformationModel" (IM) which is the model/ontology itself and "ModelElements" (also belonging to the IM) that can either be classes, properties or instances. The "Morphism" associates a pair of "Objects" (related and relating), and classifies their relationship with a "MorphismType", "KnowledgeMappingType" (if the morphism is a mapping), and "Match/Mismatch" class. The "Morphism" is also prepared to store transformation oriented "ExecutableCode" that will be written in the ATLAS Transformation Language (ATL) and can be used by several organizations to automatically transform and exchange data with their business partners as envisaged before (Correia 2010).

To help in the decision making that will be presented in section 4.4.4 a new class was created

called CompareMatchMismatch that has a field "weight" that will help the Intelligent Supervisor to choose the best solution to present to the user.

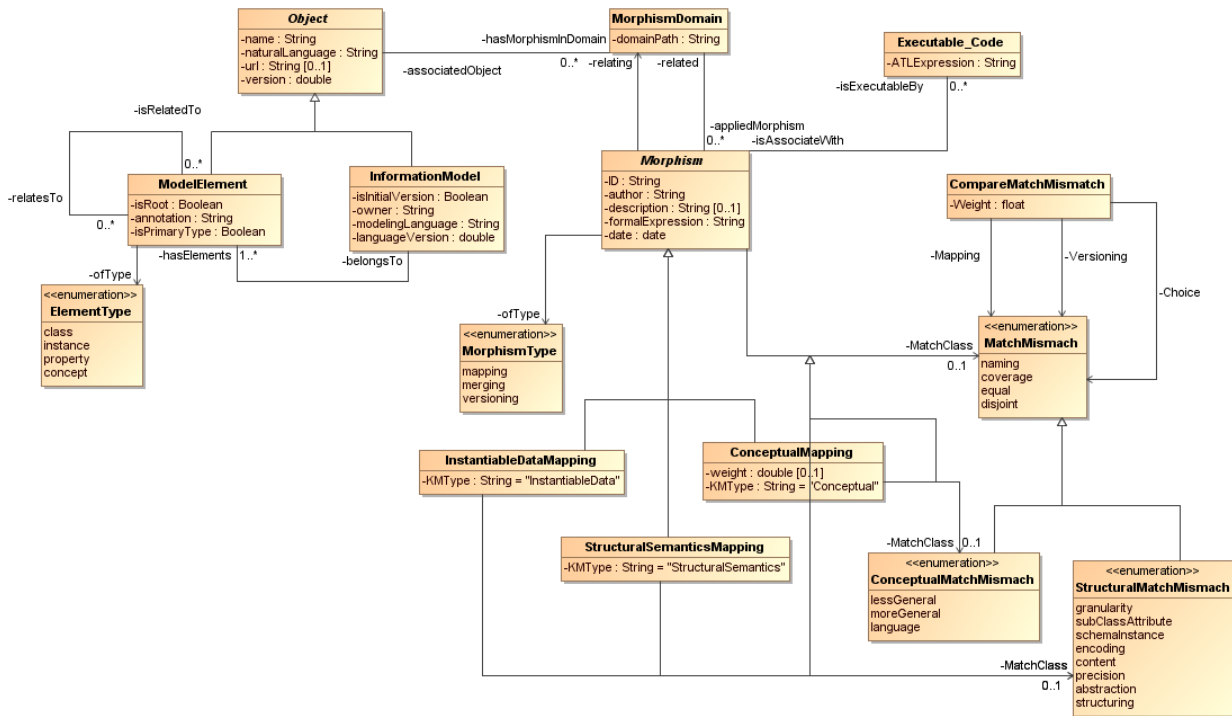


Figure 4-3 – Structure of Communication Mediator (J. Sarraipa et al. 2010)

4.4. MIRAI Intelligent Supervisor Block

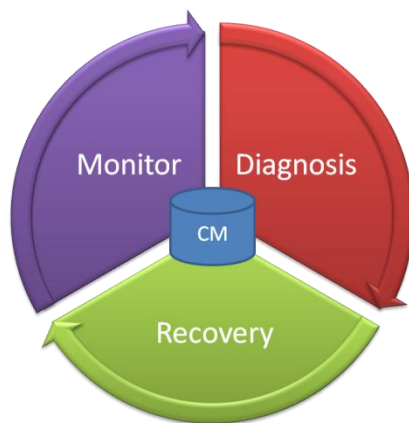


Figure 4-4 - System to re-adapt the mappings

This block is the more important block of this work as it is responsible for the detection of the harmonization breaks by performing a scan in the organization's CM in search for new morphisms or morphisms evolutions. In the Figure 4-4 how the system reacts to recovery from an evolution of a morphism is represented, in this block are two agents involved, the Agent Monitor Mediator and the Agent MoMo. They are responsible to search for changes (Monitor) and then to check if the changes

occurred have an impact in the system's interoperability status (Diagnosis), and if it does, it proposes a possible solution (Recovery), this is what is represented in the circle in the figure. This section will explain how this block works and with who it will interact, it starts explaining the Agents Monitor Mediator and MoMo, then the workflow explain how the block works, finally is explained how the morphism is created and the decision making in the creation of the morphism.

4.4.1. Agent Monitor Mediator

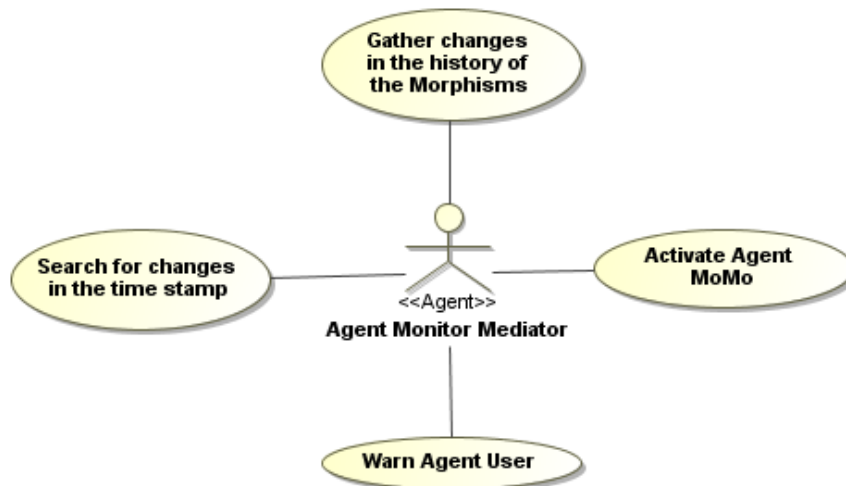


Figure 4-5 - Use Case of Agent Monitor Mediator

This agent detects changes in the CM, being responsible for the detection of the breaks in the harmonization. Basically, it does a scanning in the CM to search for changes in the mappings, this is made by finding an evolution of the same (versioning's). Every time it finds a new mapping it warns the Agent User and the Agent MoMo, these features can be seen in the use case in Figure 4-5, that shows all the action of this agent.

This agent needs to do two types of communications with other agents, one type is to inform about what happen, and the other is to ask the Agent MoMo to present a solution to the founded changes in the CM, this communications follows the FIPA protocol.

4.4.2. Agent MoMo

This agent acts when requested by the Agent Monitor Mediator. It is this agent that makes the decision in the MIRAI, in Figure 4-6 it is possible to see all the action that this agent will perform. The main objective is to check if the changes occurred have an impact in the system interoperable status, and if it does, it warns the Agent User and proposes a possible solution (i.e. a new morphism or a correction to the existing one). This new mapping is done by using MapT, comparing the two morphisms in use, and using rules in the decision of the matchclass. The communication used between Agent MoMo and Agent User is again FIPA protocol.

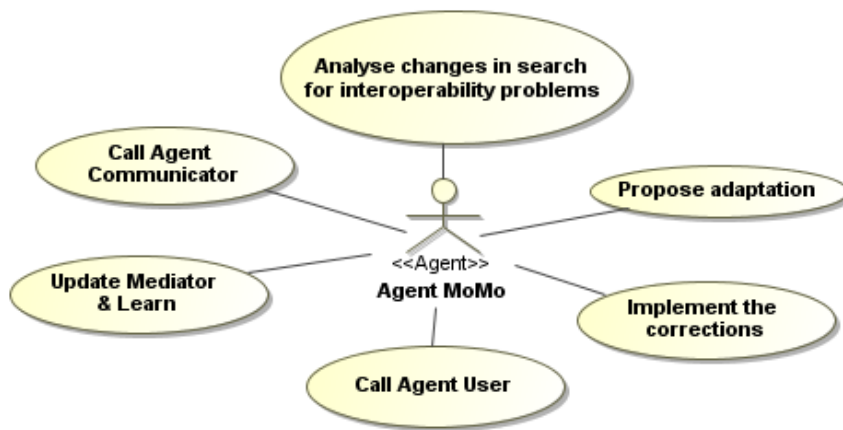


Figure 4-6 - Use Case of Agent MoMo

4.4.3. Workflow

Here it is explained the workflow of the Intelligent Supervisor Block, demonstrating how the agents will work and interact between them, that is represented in diagrams sequence and state machine that will appear during this section. In the Figure 4-7 is illustrated the workflow of this block, it is represented in a high level and demonstrates how the agents interact between them. In this figure it is possible to see that the agents are autonomous of each other, but need to cooperate between them to aim for the main objective, this will be demonstrated throughout this section.

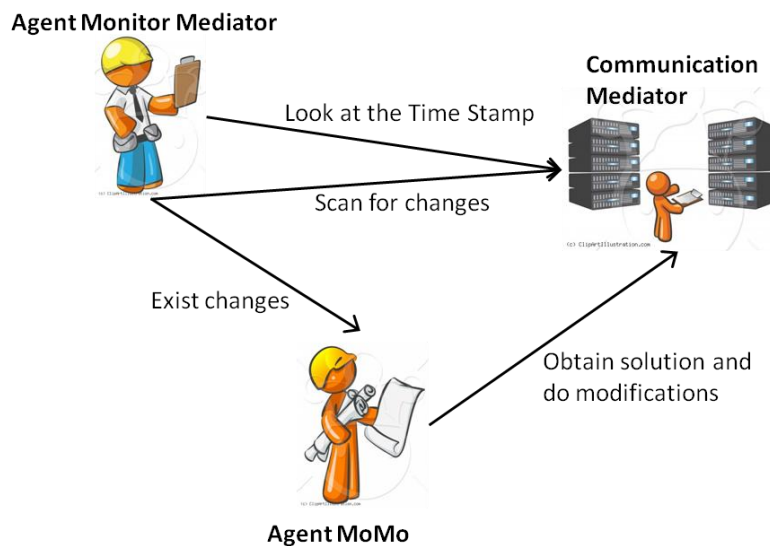


Figure 4-7 - MIRAI High Level Interaction

The Figure 4-7 demonstrates how the Intelligent Supervisor reacts to changes in the CM, it begins in the Agent Monitor Mediator, which look at the time stamp until it finds changes. Then it will scan the CM to search for new mappings, every time that finds one it will call for Agent MoMo, this is the routine of the Agent Monitor Mediator, repeating this process once a day. When the Agent MoMo is called, it will evaluate the received changes and propose a solution to the user. Then the user will

accept or not that suggestion.

Since the user can refuse the proposal presented, the system will react and ask the user to change the proposal to be like he wants, then Agent MoMo can do the update of the CM and ask the Agent Communicator to send a message informing about the updates to the Network. During this section it will explain this workflow with more detail.

The Figure 4-8 is about the state machine of the Intelligent Supervisor Block and demonstrates how it will work and the difficulties that it will find, beginning with the detection of a possible modification in the CM, until the system proposes a solution to the User.

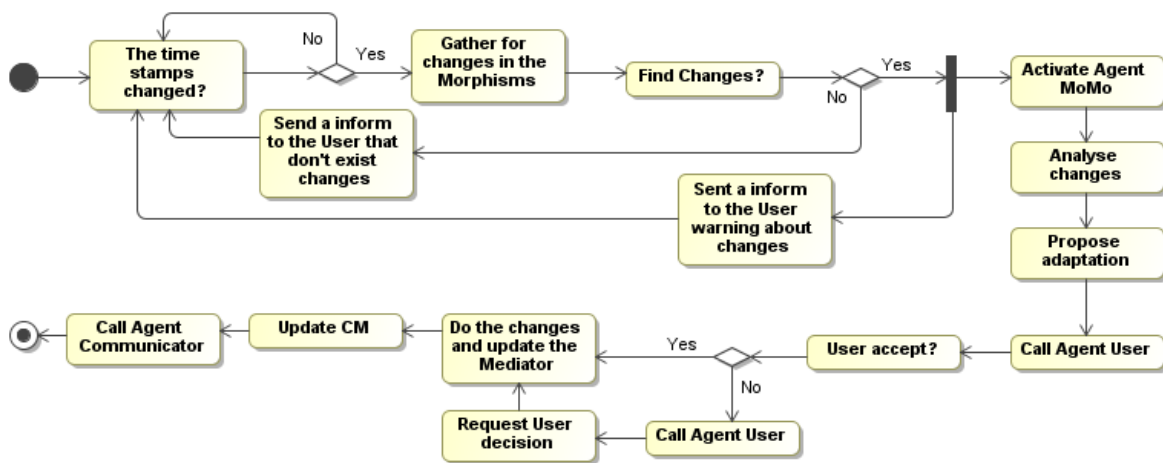


Figure 4-8 - State Machine of Intelligent Supervisor

The Intelligent Supervisor Block is explained in the Figure 4-8, it starts with the work of Agent Monitor Mediator, and it is that will trigger almost all of the actions. The process starts to verify if the changes in the time stamp of the CM exist, that happens because modifications occur in the CM, then the Agent Monitor Mediator will check if these changes are the type of versioning. Every time that it finds versioning's, it will ask for the help of the Agent MoMo. This agent is responsible to find a possible solution to the problem, and then proposing a new mapping to the user. This proposal must have the confirmation of the user, so depending of the answer of the user the agent updates the CM. In some cases the user can refuse the proposal, if that happens the Agent MoMo will ask the user to propose a new mapping.

After explaining how the Intelligent Supervisor block works what is left to explain is about how they communicate inside the block and with the others blocks, these are shown in the Figure 4-9 and 4-10. To explain this interaction it was needed to divide it in two cases, the 1st case shows when an internal problem appears, this means that a problem inside the same CM occurs, and the other with an external problem, will receive a warning from another CM.

The Figure 4-9 represents an internal problem, this occurs when the Intelligent Supervisor finds a change in the CM, this will trigger all the process explained in the Figure 4-8. Every time that the Agent Monitor Mediator or MoMo do something that is relevant to the user, they send a message to the Agent User informing the user about what they are doing at the moment. In the figure it is possible

to see two more communications made by the Intelligent Supervisor, one of these communications is made by the Agent Monitor Mediator to the Agent MoMo, informing that a change in the CM exists and with a list of the mappings. The other communication is made by the Agent MoMo to the Agent Communicator with the objective to inform the Network about the changes that occurred in the CM.

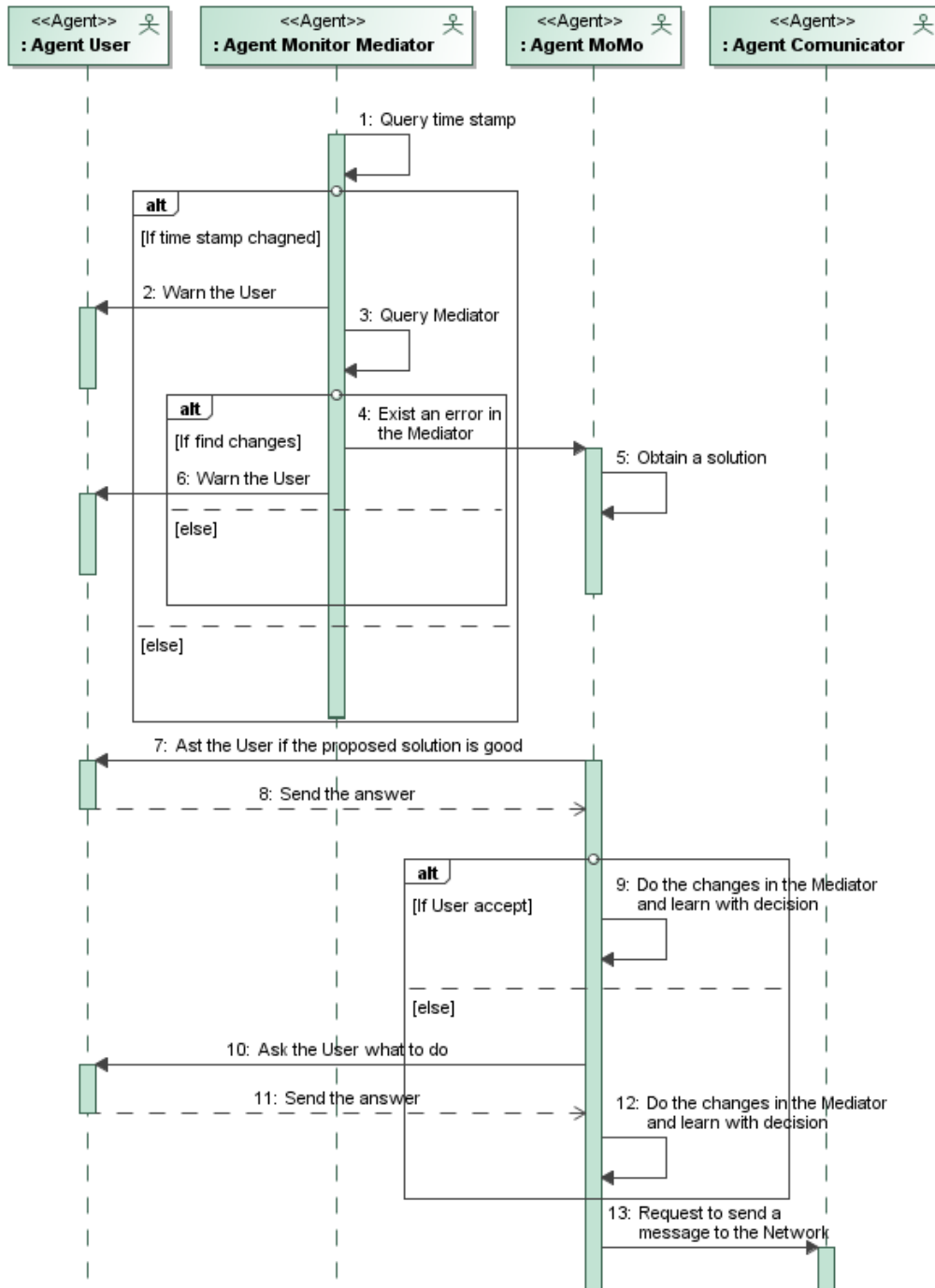


Figure 4-9 - Sequence diagram of Intelligent Supervisor and External Communicator with internal problem

The 2nd case is represented in Figure 4-10 is about an external problem, it occurs when other organisations send a message informing about a change in their CM. In this case the Agent Communicator receive a message from the Network warning about a change which may be needed to

modify, then the agent will send this message to Agent Monitor Mediator to verify if that change has an impact in the system, the rest of the process follow the explanation made in the Figure 4-8 and the communications are equal to the explanation in the Figure 4-9.

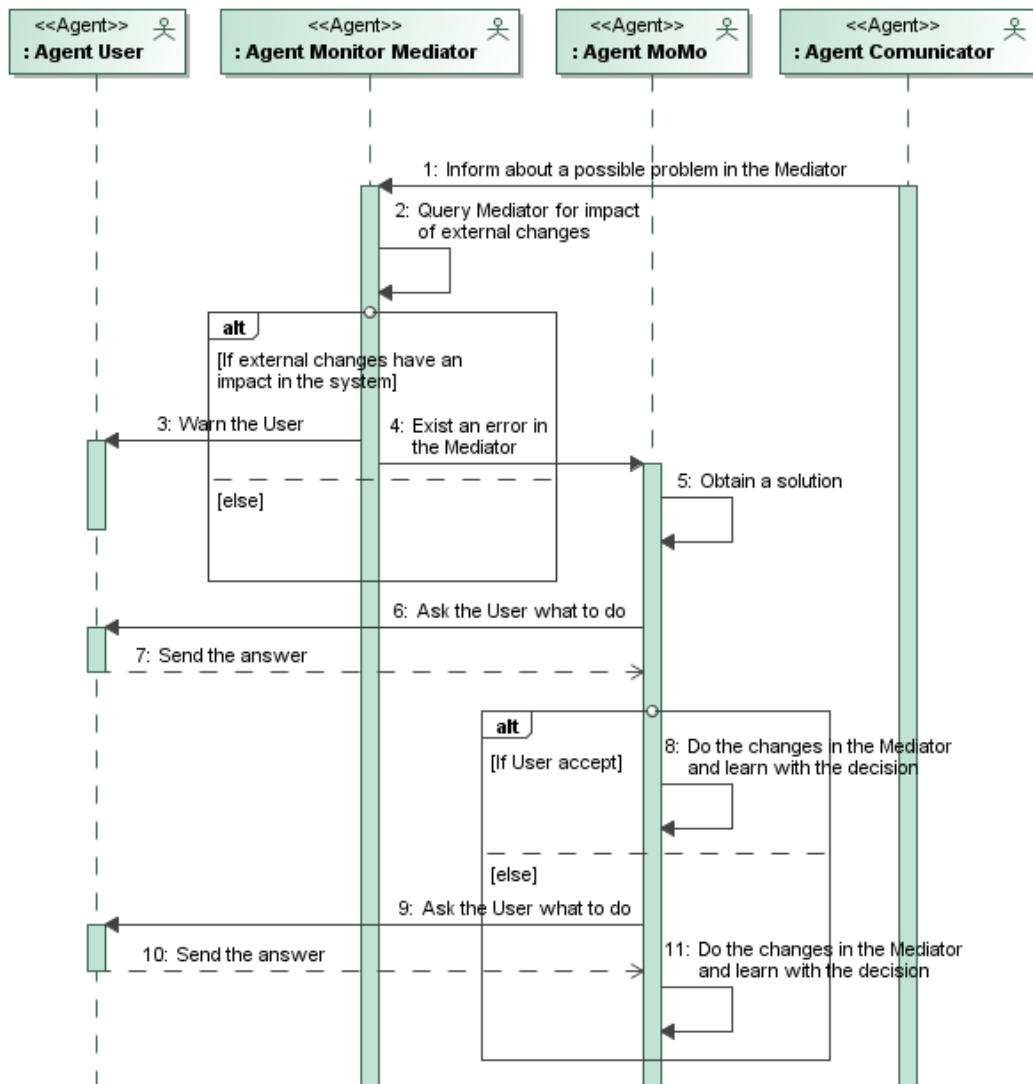


Figure 4-10 - Sequence diagram of Intelligent Supervisor and External Comunicator with external problem

4.4.4. Algorithm for Generation of new Morphisms

The earlier sections explain how MIRAI system detects changes within the morphisms a network of companies, while here it will be explained how the Agent MoMo generates a new morphism to respond to a harmonization breaking situation. Thus, every time that a versioning is detected by MIRAI, it is required to evaluate the morphism and generate a new mapping to recover the interoperability status. Figure 4-11 illustrates the process of how the proposing this new morphism is calculated, which following the explanation of section 3.4, is described using a tuple (MapT) represented by equation 4 (section 3.4.5), and presented to the user.

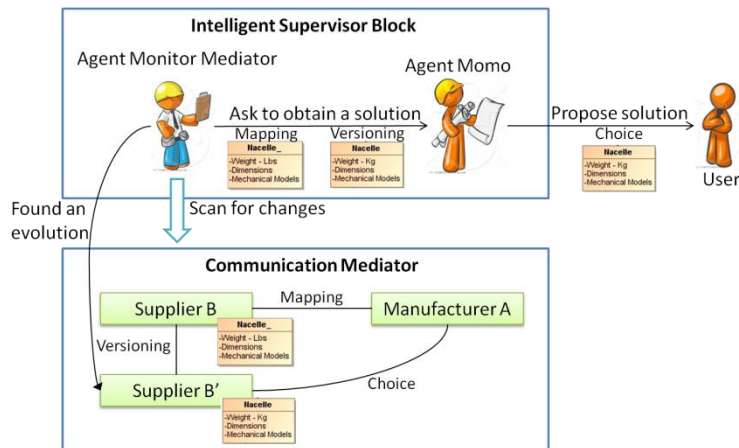


Figure 4-11 - Detection and propose of morphisms

In the process of generating a new MapT, some arguments are easy to fill, such as the relating elements (MElems), but others are dependent on more complex calculations based on the two existing morphisms (Mapping and Versioning). Figure 4-12 shows an abstraction of a situation where MIRAI needs to respond and propose a new morphism (w), after an evolution of a specific model ($g = M_1 \rightarrow M_1'$) is detected. The figure also represents the existing relation (f) between the two models (M_1 and M_2), and the respective model elements that are related (x from M_1 , y from M_2 , and z from M_1').

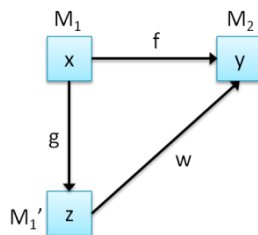


Figure 4-12 - Mappings between the models

Equation 5 formalizes a transformation in a generalized way, where for every x and y that are elements of Model, f is the tuple (belonging to MapT domain) that represents the morphism. In order to calculate the remaining constituents of MapT, namely the MatchClass and the Exp. The Equation 6 is used to describe the Exp, and It has three model element variables, x , y and z , as well as three morphisms f , g and w to represent the existing mapping, the versioning, and the new mapping proposed by MIRAI, which follows the same principle as the transitivity property of mathematics (Binary relation). Having a relation from the element z to the element x (designated by the inverse morphism g^{-1}), and a relation from the element x to the element y (designated by the morphism f), then it is reasonable to say that exists a relation from the element z with the element y (designated by the morphism w), which is a composition of g^{-1} and f (g^{-1} of). Therefore the new mapping proposed, namely the elements MatchClass and the Exp are calculated following Equation 6. In Annex 9.1 it is included the mapping patterns of morphism w for MatchClass.

Every time that a versioning appears it is needed to create a new mapping, as it is possible to see in the Figure 5-10. This new mapping was called Choice, since it depends on the acceptance of

the user, and the fill of the fields of this new mapping are directly dependent of the Mapping and the Versioning, after the confirmation of the user the Agent MoMo saves the mapping in the CM. To help the user it was implemented a learning ability that is explained in the next section.

$$\forall x, y \in \text{Model} \wedge f \in \text{MapT} \exists f: M \rightarrow M \wedge f(x) = y \quad \text{Equation 5}$$

$$\forall x, y, z \in \text{Model}, w \in \text{MapT} \wedge f(x) = y \wedge g(x) = z \exists w = f \circ g^{-1}(x) \wedge w(z) = y \quad \text{Equation 6}$$

4.4.5. Decision Support

This work is an implementation of CAS-SIF, and implements the layer 1 and 2 of the framework. In this section it is explained as an approach of a solution to the layer 1, which is the Integration & Interoperability Intelligence Layer of the CAS-SIF (Figure 3-10) that enables the system to adapt to the environment and to be able to recover the interoperability status, to achieve better solutions and a more accuracy system, therefore it was decided to use a learning capability. This learning capability has the aim to analyse new changes, and propose a solution to those changes, based in the past decisions, while progressively updating the CM. With this is possible to provide a more educated mapping adaptation that will fit better the user's needs, facilitating its decision and minimizing the transient periods. This learning ability is based on weights that help in the choice of MatchClass for the new mapping, and increasing intelligence over time.

4.4.5.1. Learning Methodology

This implementation was made to help the MIRAI in the proposal of the new morphism, with the aim of learning with the choices of the user and in time to avoid poor decisions by MIRAI. Every time that an evolution appears the MIRAI will propose a possible solution to the user, but it could happen that the user does not agree with that, and changes the proposal. The intention in the use of the learning ability is so that MIRAI learn the decisions of the user and use it in the future. This is made using a "weight", in the CM a list with all the possible cases exists, and every time the user changes the proposal, the weight of the user choice is incremented. So the next time the MIRAI will choose the one with the higher weight, gaining a more reliable decision.

4.5. MIRAI External Communicator Block

The External Communicator Block is responsible for the communication between the MIRAI's, in this section it is explained how the communications is made, how the Agent Communicator works, and the technology involved.

4.5.1. Agent Communicator

This agent is responsible for the communication between different MIRAI's of a collaborative network, in Figure 4-13 it is possible to see all the actions performed by this agent. This agent has some tasks to do, one of them is to inform the Network every time that exist changes in the CM, thus enabling them to react as well.

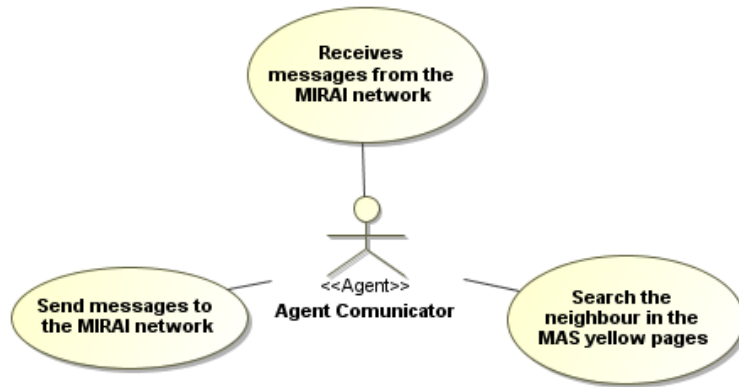
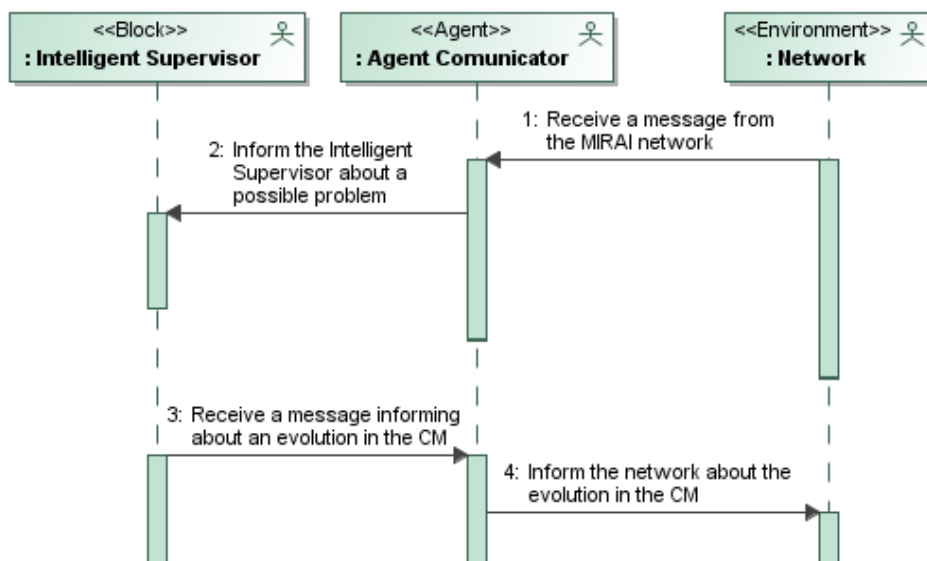


Figure 4-13 - Use Case of Agent Communicator

4.5.2. Workflow

Since this block only takes care of the communications between the MIRAI's, his only purpose is to send or receive messages from another MIRAI's. These communications are made using web services provided by JADE and the web service is known by the name of Web Service Integration Gateway (WSIG). The WSIG is an agent that sends and receives messages from other agents, do the subscription of the agents in the Director Facilitator (DF) and publishing the service in the UDDI (Universal Description, Discovery and Integration) registry.



4-14 - Actions of the Agent Communicator

In the Figure 4-14 it is possible to see the communications made by the Agent Communicator, the agent receives from the Network news about changes that occur in another MIRAI, and then will send that change to the Agent Monitor Mediator to see if the received information will have impact in the system. Then if this information creates some changes, the Agent MoMo will ask the Agent Communicator to warn the Network about that, all this is made with web services.

4.6. MIRAI Administration Block

The Administration Block is where the user does all the decisions in relation to the mappings, and it is possible to see how the MIRAI is working. In this section it will be explained what is the Agent User, how it works and interacts with the rest of the agents.

4.6.1. Agent User

This agent is the interface between the MIRAI and the human user. Through this agent the human is informed of the solution proposed by the Agent MoMo to decide whether if he / she accept it, or propose a new one, in the Figure 4-15 it is possible to see the Use Case showing the actions that the agent will perform. This agent communicate with all the other agents, since the only proposition is to be informed except when the user has to chose a solution, which was decided to use the FIPA protocols.

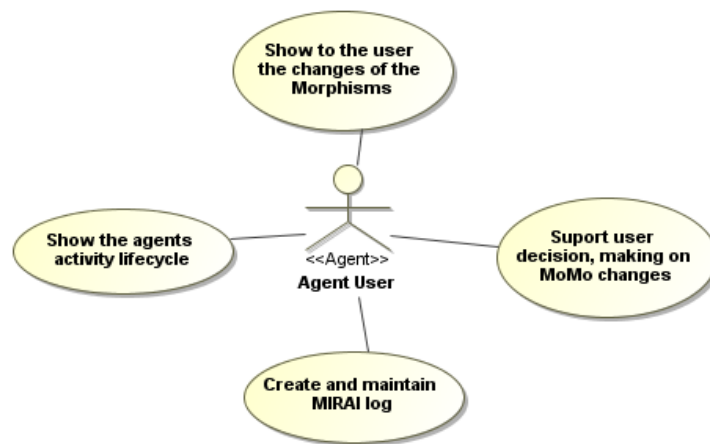


Figure 4-15 - Use Case of Agent User

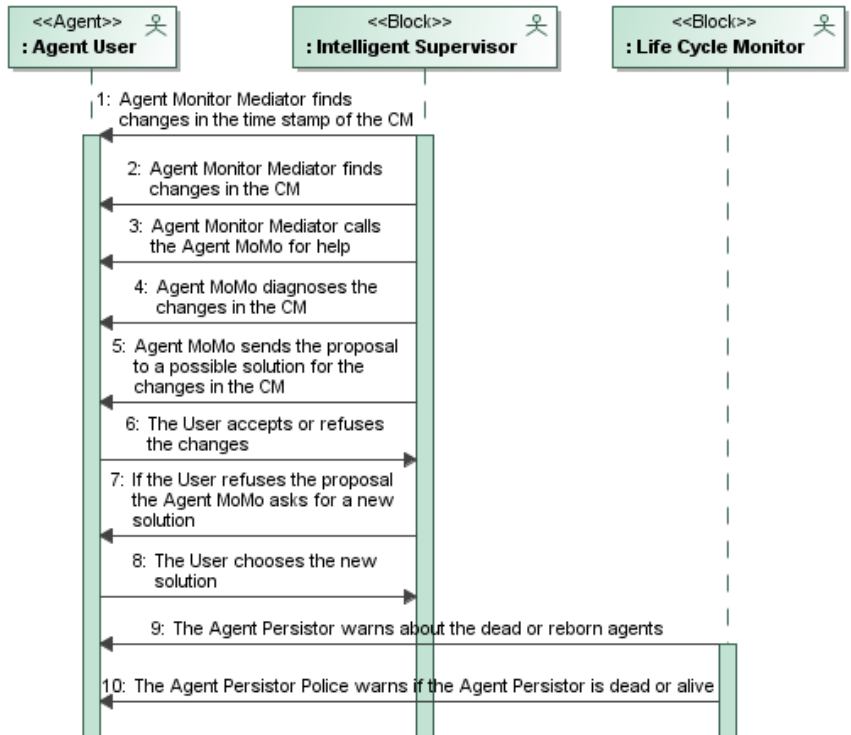
4.6.2. Workflow

This block is the connection between the user and the MIRAI, in here all the interaction is made. It is responsible for showing all the messages sent by the other blocks to him, these messages can be informing messages, but in one special case is to ask an opinion to the user.

In the Figure 4-16 it is possible to see all the messages exchanged by the agents. Almost all of the messages are informative, for example the Agent Monitor Mediator send messages informing about what it is doing, creating a log of all the steps, the Agent Persistor is sending messages every time that it finds a dead agent or a reborn one, the Agent Persistor Police is the same but only in the specific case of Agent Persistor.

Only in the case of the Agent MoMo has two type of messages, the informing messages and the messages that need decision made by the user, in the Figure 4-16, it is possible to see these messages, the agent informs that it received from the Agent Monitor Mediator a list of changes, so it will diagnose the problem, these are the messages 4 represented in the figure. In the moment that the Agent MoMo has a solution, it will send a proposal to the Agent User to let the user decide about that

(message 5). Then it receives the answer from the user, if the user accept it, the agent will save this new mapping in the CM, if not, it will ask the user a possible solution to that problem.



4-16 - Interaction between the Agent User and the rest of the agents

4.7. MIRAI Life Cycle Monitor Block

This block is responsible for the maintenance of the MIRAI interoperable, if one of the agents dies the harmonisation of the systems breaks. For example, if Agent MoMo dies, it is impossible to have a diagnosis and a recovery of the system, so if a new mapping occurs the MIRAI will identify the problem, but will not propose a solution, so in this case it is not doing his job. Because of this it is important to have an agent monitoring the MIRAI for the case that an agent dies it will resurrect him, avoiding a harmonisation break. The Agent Persistor Police exist to assure that the Agent Persistor doesn't dies and continues to do the monitor to the system.

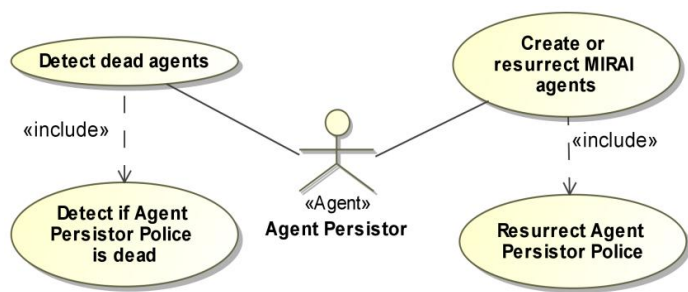


Figure 4-17 - Use Case of Agent Persistor

4.7.1. Agent Persistor

This agent controls if any agents dies, to resurrect them if required, in Figure 4-17 is possible to see all the actions involving this agent, and how it will interact with the rest. Since it will be responsible for the resurrection of the dead agent, something to help in that manner is needed, so the JADE provides an agent that does that work, and that agent is the Agent Management System (AMS). This agent receives the registration of all agents while managing their life cycle.

4.7.2. Agent Persistor Police

This agent has a similar role but only controls the Agent Persistor, thus adding some redundancy to the MIRAI, in Figure 4-18 is possible to see the actions performed by the agent.

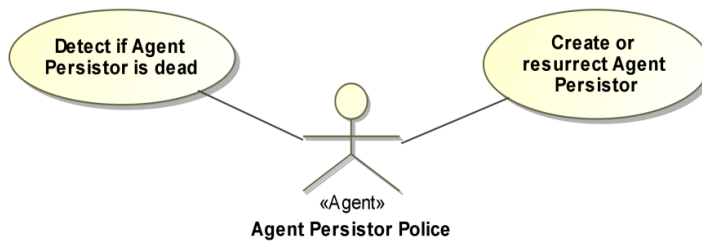


Figure 4-18 - Use Case of Agent Persistor Police

4.7.3. Workflow



Figure 4-19 - State Machine of Life Cycle Monitor

In Figure 4-19 it is represented the State Machine of Life Cycle Monitor, in this block exist two agents that are responsible to maintain all the agents working, sometimes an agent can die, and that creates an interoperability break and a malfunction of the system. To avoid that, every time it is detected that an agent dies, these agents resurrect them in that moment, preventing further damage.

Now it will be show the Diagrams Sequence that demonstrates how the process flows and the communications made between the different agents. The diagram to describe is in Figure 4-20 and represents the Life Cycle Monitor, this block contains the Agents Persistor and Persistor Police, and it shows the process to resurrect the agents. The first thing the agents do is a subscription in the ENS (Event Notification Service), this service will inform every time an agent die or is created, when one of the agents receive a message informing that an agent dies, it will resurrect him in that moment using the ENS, than inform the user.

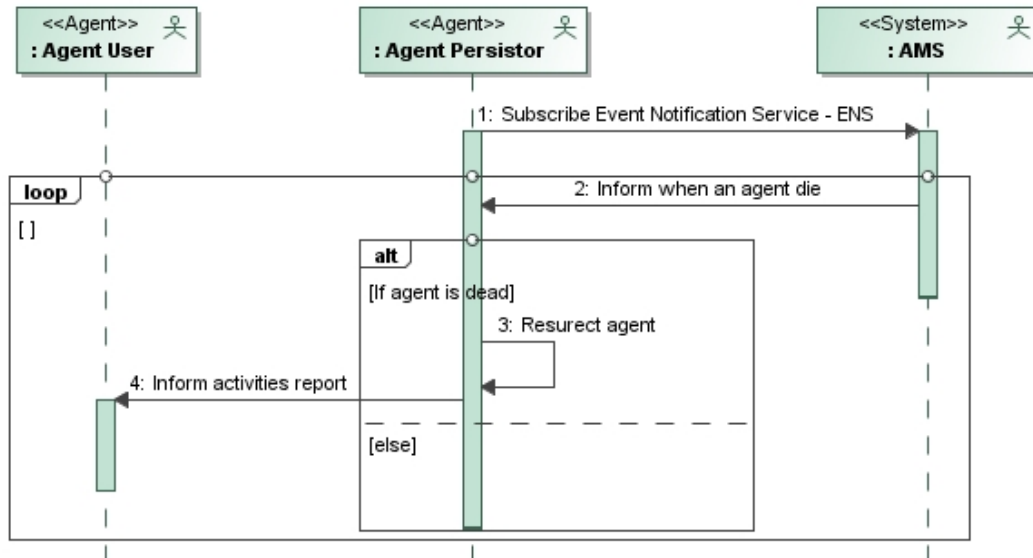


Figure 4-20 - Diagram Sequence of Life Cycle Monitor

4.8. MIRAI Usability Cases

Earlier it was explained how the MIRAI works, in this section it will be explain the scenarios that the MIRAI will help in the interoperability status. Four scenarios to implement in the framework will be presented, and they are very important to the organisations, because they will grant a great autonomy of the system.

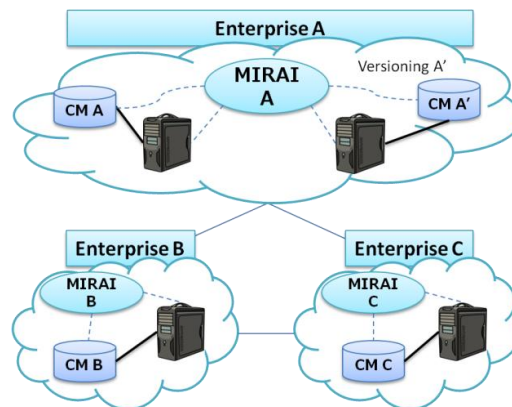


Figure 4-21 – Morphism Evolution Scenario

In Figure 4-21 it is where the Morphism Evolution Scenario is presented, this scenario is the base of this work, since it is the answer to the research problem made in the section 1.3. It is about a change of one of the nodes, so the MIRAI has to find a change in the CM and be able to propose an adaptation to the user, finally that node sends a message informing all the others nodes about that change, so they are able to adapt.

The New Enterprise Scenario is presented in Figure 4-22, this scenario is about the entry of a new node to the Network, so the MIRAI identifies a new enterprise (Enterprise D) and is able to propose the mappings between the enterprise A and D, preventing the user to do all that work.

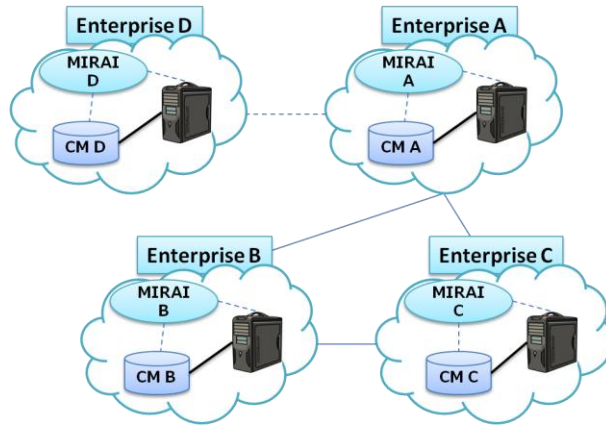


Figure 4-22 – New Enterprise Scenario

The New Connection Scenario is about the Figure 4-23, this scenario is about a new connection between two nodes, for some reason the node D and B need to be connected, so the MIRAI will propose all the mappings needed to establishing the connection. Avoiding again that the user has to do all these mappings, since it is a slow and complicated work.

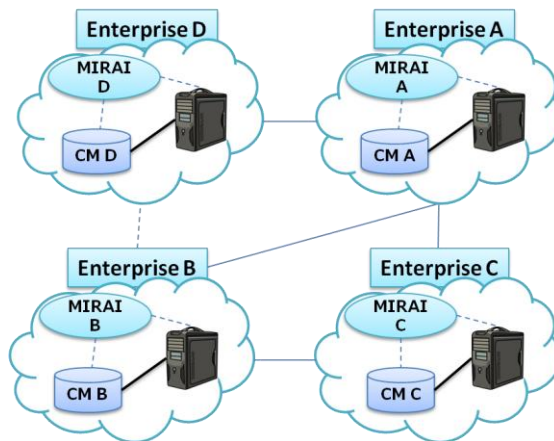


Figure 4-23 – New Connection Scenario

The last scenario is called the Removal of an Enterprise Scenario and is presented in Figure 4-24, this is about the removal of one of the nodes. When for some reason an enterprise gets out from the organisation, the MIRAI is able to adjust the mappings, removing all the mappings related with the removed enterprise.

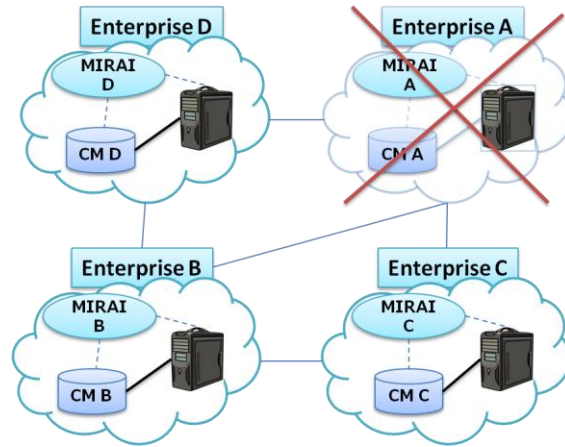


Figure 4-24 – Removal of an Enterprise Scenario

5. PROOF-OF-CONCEPT IMPLEMENTATION

In order to validate the viability of the framework proposed in section 4.3, it was necessary to implement it. Since this dissertation is an implementation of the CAS-SIF (Figure 3-10), it was also chosen to be the proof-of-concept implementation. On the other hand, this dissertation was developed integrated in the Group for Research in Interoperability of Systems (GRIS) at UNINOVA and in the CRESCENDO Project.

The CRESCENDO project is a project for the aeronautical industry with the aim to help different aspects in the industry, this work has the aim to help in the Enterprise Collaboration Capabilities through maintaining the interoperability status of the CN. The proof-of-concept consists in an application scenario based in the CRESCENDO project, simulating a CN that cooperate to manufacture a turbine engine for an aeroplane.

To develop MIRAI a different technology was used, starting with the CAS-SIF framework that it was a work of Agostinho et al. (Carlos Agostinho & Ricardo Jardim-Goncalves 2009), the Communicator Mediator that was developed by Sarraipa et al. (J. Sarraipa et al. 2010) and it used to store the morphisms that use the tuple proposed by Carlos et al. (Carlos Agostinho et al. 2011). Others technology in used and very important is the JADE to implement the agents, the OWL that is the ontology language used in the CM and it was used the Protégé to modelling the OWL file, these technologies is explained in the section 5.2.2.

5.1. Application Scenarios

The proposal to help maintain the interoperability status of business networks with the help of MAS, presented in section 4.3, relies on a framework that monitors mappings in pursuit of changes in the CM. To demonstrate the reliability of this framework, an application scenario based on Crescendo project is presented, a scenario that simulates a CN that are cooperating to manufacture a turbine engine for an aeroplane was created.

When a CN is formed in the beginning it is decided who are the involved partners and the tasks that each one will fulfil, in this case the objective is to manufacture a turbine engine, and to produce three parts are needed, the nacelle, the engine and the pylon. So it was decided to have four partners:

- *Supplier B* where the Nacelle is made;
- *Supplier C* where the Pylon is made;
- *Supplier D* where the Engine is made;
- *Manufacturer A* where all the three parts are assembled to produce the Turbine.

In a big production like this case coordination and discipline between the parts involved in the CN are required, because a mistake can be catastrophic to the production and create big loss in time

and consequently, a loss in money. So, teams from different disciplines from design to aerodynamics, simulation, etc. need to cooperate, describing and modelling the several stages of the development LC of the product. In order to achieve this objective, some enterprises use MBSE strategies to help in that development and attain seamless integration between engineering processes. However, each enterprise has its own workflows and answers to several customer requirements, thus sometimes an interoperability disruption can occur, which complicates collaboration and data exchange.

In this scenario the four enterprises applied the MBSE paradigm, and in addition some have even worked with different modelling languages for structural models. After storing the mappings in the corresponding CM's following the MapT description of equation 4, the manufacture end ups with three high level morphisms and the rest of the suppliers have one high level morphisms (CMa: A→B, A→C and A→D; CMb: B→A; CMc: C→A; CMd: D→A) and the corresponding submorphisms to each of the relating entities and attributes.

This study case will have three possible scenarios, the 1st one is about the change in the requisites in a turbine, 2nd is about a software change, and the 3rd is to demonstrate that the MIRAI is prepared to map concepts, although it is not covered during this dissertation. These three different scenarios are only examples, it was made in order to be the most real as possible, in the next subsections each of these scenarios will be described.

5.1.1. First Scenario - Change in the Requisites in a Turbine Engine

This scenario demonstrates how the MIRAI reacts to changes in the requisites in a turbine like it is shown in Figure 5-1, in the figure where the CN is represented by the four enterprises and how they are collaborating to accomplish the final product. They are following the agreement made when the creation of the CN, and have the MIRAI integrated in the system.

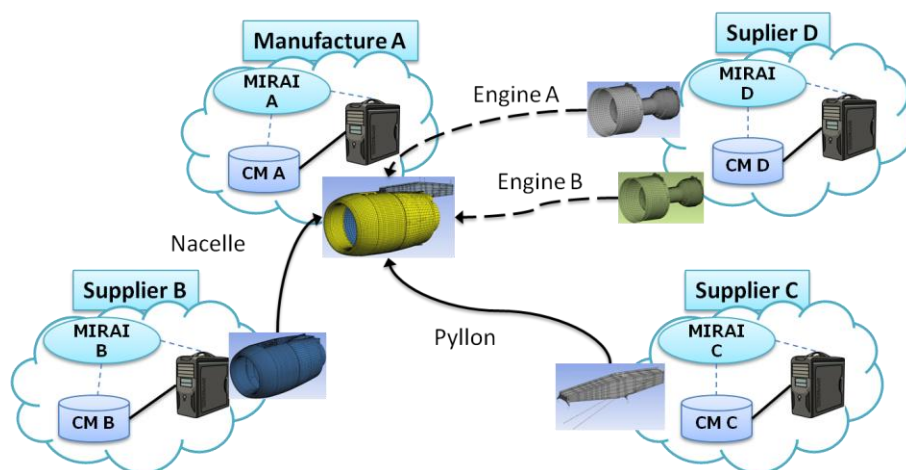


Figure 5-1 - MIRAI Network for the Crescendo Scenario

In this scenario a new requisite to the aeroplane was made, the buyer wants the aeroplane to be more economic and eco-friendly, so in order to achieve this, the manufacturer was forced to change the engine that was a Turbojets to a Turbofan engine. The modification was made because the Turbofan is a more economic, efficient and makes less noise than the other, thus it is in the

requests parameters. In the Figure 5-1 a change of the Engine A to Engine B was made, and this forced an evolution in the model A to A', which after detection by Agent Monitor Mediator triggers a warning about the change, this has to be handled swiftly with the risk of jeopardizing the full network.

In the Figure 5-2 the models to produce an aeroplane are presented, this is following the MBSE paradigm. The 1st one is the Requirement Model where the requirements of the user are described, and it is possible to see that the user wants to construct a commercial aeroplane. The other model is the Structural Model where the turbine engine that is to be used in the aeroplane is described, then a connection between the two models is needed, these connections are to describe the aeroplane following the requirements of the user, these connections are represented in the figure with dash line.

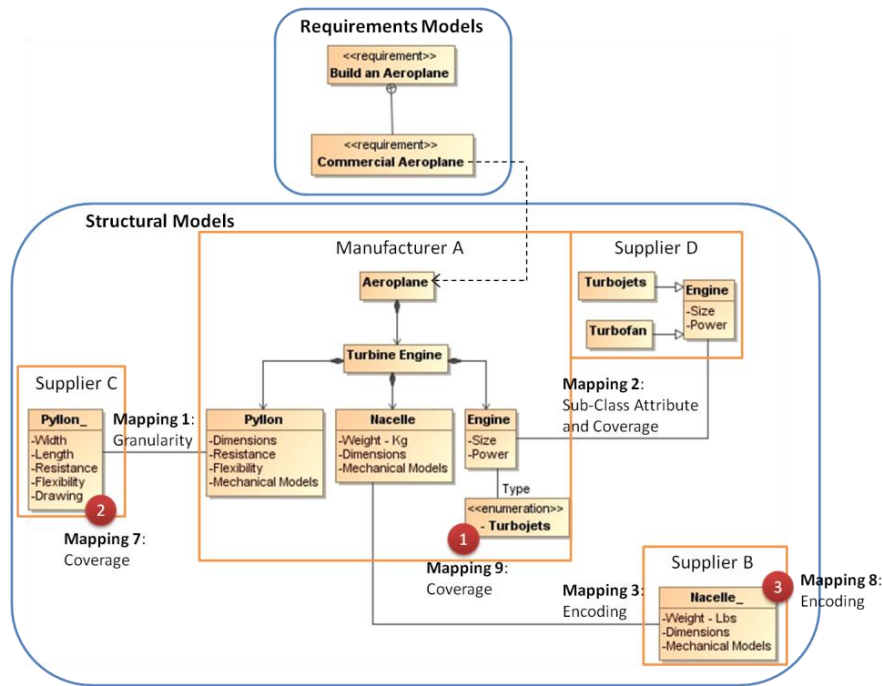


Figure 5-2 – Representation of the mappings between the models without the evolution in the system

The Figure 5-3 represents a new evolution resulting of a new requirement of the user, this time the user is asking for an aeroplane more economic and friendly for the environment, to accomplish this, a modification in the aeroplane is required. To create an economic and eco-friendly aeroplane the turbojet engine was switched to a turbofan engine, like it was showned in the figure (dashed lines).

Until now the models to construct the aeroplane were explained, to answer the request of the client, the model was modified. Although all the process follows the MBSE paradigm this does not mean that the different enterprises use the same model languages, for example the Manufacturer A uses Model Language X and Supplier B use Model Language Y, this will create some interoperability issues. So each enterprise have the mappings relating the models represented in the CM, for example the Manufacturer A in this case has four mappings, A→A', A→B, A→C and A→D, that are divided in different mappings between the two figures, the mappings 1, 2 and 3 is in Figure 5-1, the mappings 4, 5 and 6 is represented in Figure 5-2, the mappings 7, 8 and 9 are in both figures and are represented with red circles because they are connected between the two structural models. Below the first 6

mappings are explained, the red circles are explained in the test cases, since that is the result of the MIRAI responding to the changes:

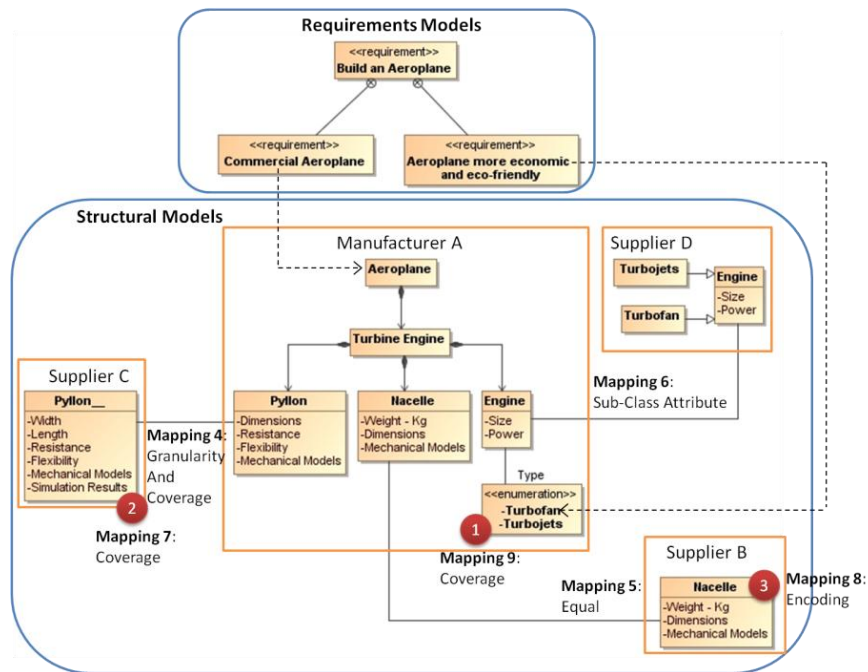


Figure 5-3 - Representation of the mappings between the models with the evolution in the system

- *Mapping 1* - represents a part of the initial morphism $A \rightarrow C$, that is represented by one mismatch, which is the type of Granularity. This is due to the same information (Dimensions) from A is decomposed in B (Width and Length);
- *Mapping 2* - represents a part of the initial morphism $A \rightarrow D$, and two mismatches are represented: 1) One (Sub-Class Attribute) is due to the enumeration attribute “type” of engine from A being represented in D by a subclass hierarchy; 2) The other (Coverage) refers to the same attribute and is about the A don't know what is a Turbofan;
- *Mapping 3* - represents a part of the initial morphism $A \rightarrow B$, and a morphism that is represented, that in this case is Encoding. This is due to in A is using Kg as a format and in B is using Lbs, causing a difference in the formats between the A and B;
- *Mapping 4* - this map is a consequence of the evolution of the Mapping 1, in this case the Granularity continue to exist, but another mismatch appear, that is a Coverage consequence of a change in the model of C (Simulation Results) that is do not exist in A, this bring loss of information;
- *Mapping 5* - is a consequence of the evolution of the Mapping 3, in this one a change in B was made, but this change made the model of B to be equal to A, so this mismatch is Equal;
- *Mapping 6* - is a consequence of the evolution of the Mapping 2, in this one because of the evolution it has only a mismatch, the Sub-Class Attribute as in the other mapping;

5.1.1.1. Test Case 1 of the Mapping 9

This test case is to explain the red circle 1, which is a mismatch of the type Coverage, consequence of the change in the requirements. Since it was needed to change the type of the engine, and in the Model A the type Turbofan didn't exist. This forced to have an evolution in the model of the Manufacturer A ($A \rightarrow A'$). So the MIRAI detected the Versioning in the model, and proposed a new mapping (Choice), that is represented in the Figure 5-3 with the Mapping 6. With this change comparing it is possible to see in the Mappings 2 and 6, that one of the Mismatch disappear, the Coverage, since the Model A' is have the same information than the Model D.

5.1.1.2. Test Case 2 of the Mapping 7

This test case is about a change in the Model C (red circle 2), this is the supplier that produce the pylon for the turbine. This supplier started to do some tests in the pylon, and introduced in the model a field with the result of this tests (Simulation Results), by doing this it was made an evolution ($C \rightarrow C'$). This versioning caused a break in the system, the Model A doesn't know what this Simulation Results are, so the MIRAI reacts and proposes to create a morphism with the Mismatch of the type Coverage (Mapping 7), this new mapping will avoid problems of interoperability in the future ($C' \rightarrow A$).

	Mapping	Versioning	Choice																																																			
Supplier B:	<table border="1"> <tr><td>3</td><td>ID</td><td>Model_X_1.1_1</td></tr> <tr><td rowspan="2">Melems = (a,b)</td><td>A</td><td>Weight - Kg</td></tr> <tr><td>B</td><td>Weight - Lbs</td></tr> <tr><td>KMType</td><td colspan="2">Instantiable Data</td></tr> <tr><td>MatchClass</td><td colspan="2">Encoding</td></tr> <tr><td>Exp</td><td colspan="2">"A = 2.2046 * B"</td></tr> </table>	3	ID	Model_X_1.1_1	Melems = (a,b)	A	Weight - Kg	B	Weight - Lbs	KMType	Instantiable Data		MatchClass	Encoding		Exp	"A = 2.2046 * B"		<table border="1"> <tr><td>8</td><td>ID</td><td>Model_X_1.1_2</td></tr> <tr><td rowspan="2">Melems = (a,b)</td><td>A</td><td>Weight - Lbs</td></tr> <tr><td>B</td><td>Weight - Kg</td></tr> <tr><td>KMType</td><td colspan="2">Instantiable Data</td></tr> <tr><td>MatchClass</td><td colspan="2">Encoding</td></tr> <tr><td>Exp</td><td colspan="2">"B = 0,4536 * A"</td></tr> </table>	8	ID	Model_X_1.1_2	Melems = (a,b)	A	Weight - Lbs	B	Weight - Kg	KMType	Instantiable Data		MatchClass	Encoding		Exp	"B = 0,4536 * A"		<table border="1"> <tr><td>5</td><td>ID</td><td>Model_X_1.1_3</td></tr> <tr><td rowspan="2">Melems = (a,b)</td><td>A</td><td>Weight - Kg</td></tr> <tr><td>B</td><td>Weight - Kg</td></tr> <tr><td>KMType</td><td colspan="2">Instantiable Data</td></tr> <tr><td>MatchClass</td><td colspan="2">Encoding</td></tr> <tr><td>Exp</td><td colspan="2">"B = A"</td></tr> </table>	5	ID	Model_X_1.1_3	Melems = (a,b)	A	Weight - Kg	B	Weight - Kg	KMType	Instantiable Data		MatchClass	Encoding		Exp	"B = A"	
3	ID	Model_X_1.1_1																																																				
Melems = (a,b)	A	Weight - Kg																																																				
	B	Weight - Lbs																																																				
KMType	Instantiable Data																																																					
MatchClass	Encoding																																																					
Exp	"A = 2.2046 * B"																																																					
8	ID	Model_X_1.1_2																																																				
Melems = (a,b)	A	Weight - Lbs																																																				
	B	Weight - Kg																																																				
KMType	Instantiable Data																																																					
MatchClass	Encoding																																																					
Exp	"B = 0,4536 * A"																																																					
5	ID	Model_X_1.1_3																																																				
Melems = (a,b)	A	Weight - Kg																																																				
	B	Weight - Kg																																																				
KMType	Instantiable Data																																																					
MatchClass	Encoding																																																					
Exp	"B = A"																																																					
Supplier C:	<table border="1"> <tr><td>1</td><td>ID</td><td>Model_X_1.2_1</td></tr> <tr><td rowspan="2">Melems = (a,b)</td><td>A</td><td>Dimensions</td></tr> <tr><td>C</td><td>Width</td></tr> <tr><td>KMType</td><td colspan="2">Instantiable Data</td></tr> <tr><td>MatchClass</td><td colspan="2">Granularity</td></tr> <tr><td>Exp</td><td colspan="2">"A \subseteq C"</td></tr> </table>	1	ID	Model_X_1.2_1	Melems = (a,b)	A	Dimensions	C	Width	KMType	Instantiable Data		MatchClass	Granularity		Exp	"A \subseteq C"		<table border="1"> <tr><td>7</td><td>ID</td><td>Model_X_1.2_3</td></tr> <tr><td rowspan="2">Melems = (a,b)</td><td>C</td><td>Pylon</td></tr> <tr><td>C'</td><td>Pylon'</td></tr> <tr><td>KMType</td><td colspan="2">Instantiable Data</td></tr> <tr><td>MatchClass</td><td colspan="2">Coverage</td></tr> <tr><td>Exp</td><td colspan="2">"C \subseteq C' "</td></tr> </table>	7	ID	Model_X_1.2_3	Melems = (a,b)	C	Pylon	C'	Pylon'	KMType	Instantiable Data		MatchClass	Coverage		Exp	"C \subseteq C' "		<table border="1"> <tr><td>4</td><td>ID</td><td>Model_X_1.2_4</td></tr> <tr><td rowspan="2">Melems = (a,b)</td><td>A</td><td>Pylon</td></tr> <tr><td>C'</td><td>Pylon'</td></tr> <tr><td>KMType</td><td colspan="2">Instantiable Data</td></tr> <tr><td>MatchClass</td><td colspan="2">Coverage</td></tr> <tr><td>Exp</td><td colspan="2">"A \subseteq C' "</td></tr> </table>	4	ID	Model_X_1.2_4	Melems = (a,b)	A	Pylon	C'	Pylon'	KMType	Instantiable Data		MatchClass	Coverage		Exp	"A \subseteq C' "	
	1	ID	Model_X_1.2_1																																																			
	Melems = (a,b)	A	Dimensions																																																			
		C	Width																																																			
KMType	Instantiable Data																																																					
MatchClass	Granularity																																																					
Exp	"A \subseteq C"																																																					
7	ID	Model_X_1.2_3																																																				
Melems = (a,b)	C	Pylon																																																				
	C'	Pylon'																																																				
KMType	Instantiable Data																																																					
MatchClass	Coverage																																																					
Exp	"C \subseteq C' "																																																					
4	ID	Model_X_1.2_4																																																				
Melems = (a,b)	A	Pylon																																																				
	C'	Pylon'																																																				
KMType	Instantiable Data																																																					
MatchClass	Coverage																																																					
Exp	"A \subseteq C' "																																																					
	<table border="1"> <tr><td>1</td><td>ID</td><td>Model_X_1.2_2</td></tr> <tr><td rowspan="2">Melems = (a,b)</td><td>A</td><td>Dimensions</td></tr> <tr><td>C</td><td>Length</td></tr> <tr><td>KMType</td><td colspan="2">Instantiable Data</td></tr> <tr><td>MatchClass</td><td colspan="2">Granularity</td></tr> <tr><td>Exp</td><td colspan="2">"A \subseteq C"</td></tr> </table>	1	ID	Model_X_1.2_2	Melems = (a,b)	A	Dimensions	C	Length	KMType	Instantiable Data		MatchClass	Granularity		Exp	"A \subseteq C"			<table border="1"> <tr><td>4</td><td>ID</td><td>Model_X_1.2_5</td></tr> <tr><td rowspan="2">Melems = (a,b)</td><td>A</td><td>Dimensions</td></tr> <tr><td>C'</td><td>Width</td></tr> <tr><td>KMType</td><td colspan="2">Instantiable Data</td></tr> <tr><td>MatchClass</td><td colspan="2">Granularity</td></tr> <tr><td>Exp</td><td colspan="2">"A \subseteq C' "</td></tr> </table>	4	ID	Model_X_1.2_5	Melems = (a,b)	A	Dimensions	C'	Width	KMType	Instantiable Data		MatchClass	Granularity		Exp	"A \subseteq C' "																		
1	ID	Model_X_1.2_2																																																				
Melems = (a,b)	A	Dimensions																																																				
	C	Length																																																				
KMType	Instantiable Data																																																					
MatchClass	Granularity																																																					
Exp	"A \subseteq C"																																																					
4	ID	Model_X_1.2_5																																																				
Melems = (a,b)	A	Dimensions																																																				
	C'	Width																																																				
KMType	Instantiable Data																																																					
MatchClass	Granularity																																																					
Exp	"A \subseteq C' "																																																					
			<table border="1"> <tr><td>4</td><td>ID</td><td>Model_X_1.2_6</td></tr> <tr><td rowspan="2">Melems = (a,b)</td><td>A</td><td>Dimensions</td></tr> <tr><td>C'</td><td>Length</td></tr> <tr><td>KMType</td><td colspan="2">Instantiable Data</td></tr> <tr><td>MatchClass</td><td colspan="2">Granularity</td></tr> <tr><td>Exp</td><td colspan="2">"A \subseteq C' "</td></tr> </table>	4	ID	Model_X_1.2_6	Melems = (a,b)	A	Dimensions	C'	Length	KMType	Instantiable Data		MatchClass	Granularity		Exp	"A \subseteq C' "																																			
4	ID	Model_X_1.2_6																																																				
Melems = (a,b)	A	Dimensions																																																				
	C'	Length																																																				
KMType	Instantiable Data																																																					
MatchClass	Granularity																																																					
Exp	"A \subseteq C' "																																																					
Supplier D:	<table border="1"> <tr><td>2</td><td>ID</td><td>Model_X_1.3_1</td></tr> <tr><td rowspan="2">Melems = (a,b)</td><td>A</td><td>Engine</td></tr> <tr><td>D</td><td>Engine</td></tr> <tr><td>KMType</td><td colspan="2">Instantiable Data</td></tr> <tr><td>MatchClass</td><td colspan="2">Coverage</td></tr> <tr><td>Exp</td><td colspan="2">"A \subseteq D"</td></tr> </table>	2	ID	Model_X_1.3_1	Melems = (a,b)	A	Engine	D	Engine	KMType	Instantiable Data		MatchClass	Coverage		Exp	"A \subseteq D"		<table border="1"> <tr><td>9</td><td>ID</td><td>Model_X_1.3_3</td></tr> <tr><td rowspan="2">Melems = (a,b)</td><td>A</td><td>Engine</td></tr> <tr><td>A'</td><td>Engine</td></tr> <tr><td>KMType</td><td colspan="2">Instantiable Data</td></tr> <tr><td>MatchClass</td><td colspan="2">Coverage</td></tr> <tr><td>Exp</td><td colspan="2">"A \subseteq A' "</td></tr> </table>	9	ID	Model_X_1.3_3	Melems = (a,b)	A	Engine	A'	Engine	KMType	Instantiable Data		MatchClass	Coverage		Exp	"A \subseteq A' "		<table border="1"> <tr><td>6</td><td>ID</td><td>Model_X_1.3_4</td></tr> <tr><td rowspan="2">Melems = (a,b)</td><td>A'</td><td>Engine</td></tr> <tr><td>D</td><td>Engine</td></tr> <tr><td>KMType</td><td colspan="2">Instantiable Data</td></tr> <tr><td>MatchClass</td><td colspan="2">Sub-Class Attrib</td></tr> <tr><td>Exp</td><td colspan="2">"A' \subseteq D"</td></tr> </table>	6	ID	Model_X_1.3_4	Melems = (a,b)	A'	Engine	D	Engine	KMType	Instantiable Data		MatchClass	Sub-Class Attrib		Exp	"A' \subseteq D"	
	2	ID	Model_X_1.3_1																																																			
Melems = (a,b)	A	Engine																																																				
	D	Engine																																																				
KMType	Instantiable Data																																																					
MatchClass	Coverage																																																					
Exp	"A \subseteq D"																																																					
9	ID	Model_X_1.3_3																																																				
Melems = (a,b)	A	Engine																																																				
	A'	Engine																																																				
KMType	Instantiable Data																																																					
MatchClass	Coverage																																																					
Exp	"A \subseteq A' "																																																					
6	ID	Model_X_1.3_4																																																				
Melems = (a,b)	A'	Engine																																																				
	D	Engine																																																				
KMType	Instantiable Data																																																					
MatchClass	Sub-Class Attrib																																																					
Exp	"A' \subseteq D"																																																					
	<table border="1"> <tr><td>2</td><td>ID</td><td>Model_X_1.3_2</td></tr> <tr><td rowspan="2">Melems = (a,b)</td><td>A</td><td>Engine</td></tr> <tr><td>D</td><td>Engine</td></tr> <tr><td>KMType</td><td colspan="2">Instantiable Data</td></tr> <tr><td>MatchClass</td><td colspan="2">Sub-Class Attrib</td></tr> <tr><td>Exp</td><td colspan="2">"A \subseteq D"</td></tr> </table>	2	ID	Model_X_1.3_2	Melems = (a,b)	A	Engine	D	Engine	KMType	Instantiable Data		MatchClass	Sub-Class Attrib		Exp	"A \subseteq D"																																					
2	ID	Model_X_1.3_2																																																				
Melems = (a,b)	A	Engine																																																				
	D	Engine																																																				
KMType	Instantiable Data																																																					
MatchClass	Sub-Class Attrib																																																					
Exp	"A \subseteq D"																																																					

Figure 5-4 - Mappings of the scenario

5.1.1.3. Test Case 3 of the Mapping 8

This test case is about a change in the Model B (red circle 3), this supplier is using a unit that do not belong to the international system of units, so it was made an update and the unit was changed. It was passed the unit pound (Lbs) to kilogram (KG), and this is a versioning in the system (Encoding), the model suffer an evolution (B→B'). The MIRAI will react to this change and will propose a new morphism, that is the mapping 5 (Equal), its equal since at this moment the two models have the same unit. Now this new morphism creates a relation between the Model A and B'.

With this example it is possible to see how the MIRAI reacts when some modification in the CM occurs, and the advantages that MIRAI brings when used to monitoring models in SE relations. In Figure 5-4 it is possible to see all the mappings explained above, and it is following the mapping described in the Figure 5-10. In the CM several Mapping's that represented the relations between two or more models exists, when a Versioning in that model occur, this forces the Mapping to be represented by another mapping which is his evolution, this new mapping was called the Choice, because the user is the one that has to choose.

5.1.2. Second Scenario

This scenario is a possible solution to the MIRAI, and is about a change in the software that is used by the enterprise, for example the enterprise did an upgrade in the version of the software, sometimes this brings some modifications that will have some impact in the models and in consequence in the mappings. In the Figure 5-5 is an example of this problem, the Manufacturer A has done an upgrade in the Software A, changing the version of V10.0 to V10.1, this creates two problems, the 1st one is inside the Manufacturer A that needs to create new mappings to relate the different versions, the 2nd is to warn the others about the change, and what kind of change is that.

To solve the 1st problem it was required to create versioning's for all the data of the model, this is made by creating mappings for each of all data, this is needed because when a model is created it is also required to create a new object, this force to relate all the mappings with this object, and to create new mappings to relate the versioning's.

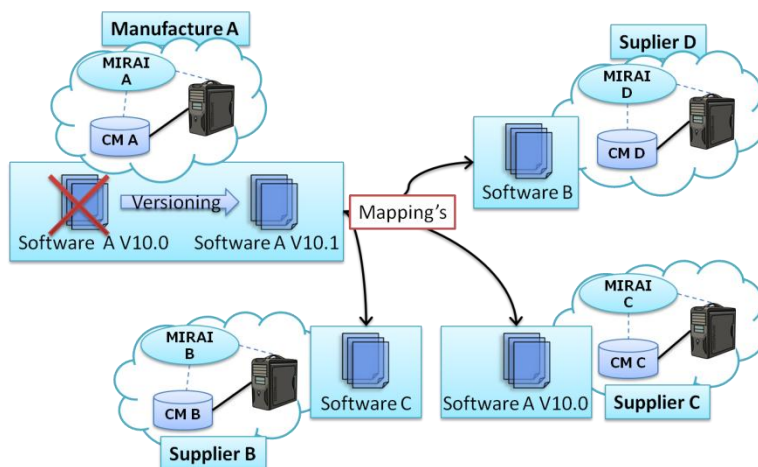


Figure 5-5 - Reacting of the network caused by a change of software

5.1.3. Third Scenario

This scenario is also a possible solution to the MIRAI and this scenario is about the MIRAI to be able to map concepts. The difference in the concepts is about the mappings, since a concept is an abstract idea, so it can have one or more instance to relate the concept, for example the concept “red” can be used to be a characteristic of an “apple”, or “cherries” or even “blood”.

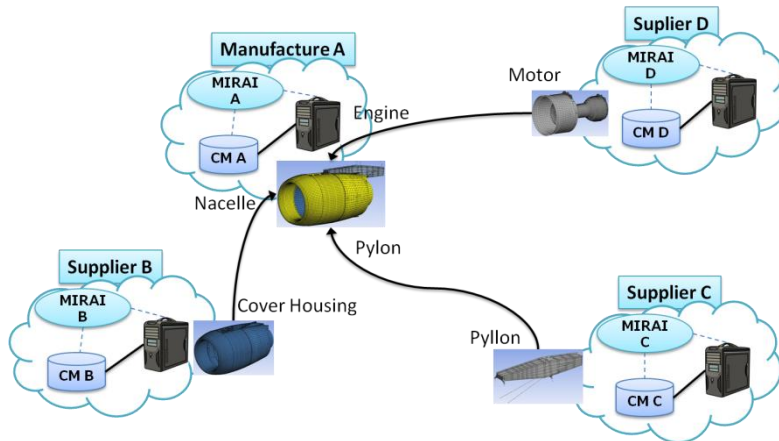


Figure 5-6 – Scenario of concepts mappings

The big problem in working with concepts is that they need some learning ability to control the mappings, since it can appear some connections which are not correct and create some entropy caused by bad mappings that are relating different concepts, example of that is appearing a mapping relating chair with hat, this is not a correct concept. This problem was covered in (Ferreira et al. 2012) where it was proposed with the help of the MIRAI and a learning ability program to manage the concepts and propose new relations between them.

Manufacturer A		
Supplier B:	ID	Model_X_1.2_1
	Melems = (a,b)	A Engine
		D Motor
	KMType	Concept
	MatchClass	language
	Exp	"A = C"
Supplier C:	ID	Model_X_1.2_2
	Melems = (a,b)	A Pylon
		C Pylon
	KMType	Concept
	MatchClass	language
	Exp	"A = C"
Supplier D:	ID	Model_X_1.2_3
	Melems = (a,b)	A Nacelle
		B Cover Housing
	KMType	Concept
	MatchClass	language
	Exp	"A = C"

Figure 5-7 – Mappings of the concepts

In Figure 5-6 it is possible to see an example of concepts, where the different enterprises are in different countries and because of that the names are told in different manners. For example the Manufacturer A is in Country X, and has the responsibility to construct the turbine engine, it needs an

engine, a pylon and a nacelle, the Supplier B constructs the nacelle, but because in Country Y it is called nacelle by the name of cover housing, the supplier C is in Country W and they write pylon as “pyllon” and finally the supplier D constructs the engine and is in Country Z but, and they call the engine by “motor”.

This is a problem of concept, both have the same definition, but with different names. So a mapping of the type of concept to relate the different names is needed, in the Figure 5-7 the three mappings used to relate the different concepts are represented. The 1st mapping is to relate the manufacturer A with the supplier B, the type is a concept with the mismatch of language, so this represents that an engine is a motor, the following mappings follow the same principle.

5.2. Implementation Steps

The main objective of this proof-of-concept consists on the implementation of the MIRAI architecture represented in the Figure 4-2 which implements the monitoring of the mappings. The next sections it will explain in more detail the functional requirement, which was divided in four steps used to describe how the MIRAI works. The Step 1 shows how the agents are distributed in the system, Step 2 describe the process of monitoring and detection of the morphisms, Step 3 explain how the morphism is proposed, Step 4 is the decision making when the creating of a new morphisms and the last one, the Step 5 explain how will work the communication between the different enterprises.

5.2.1. Step 0 – MIRAI setup

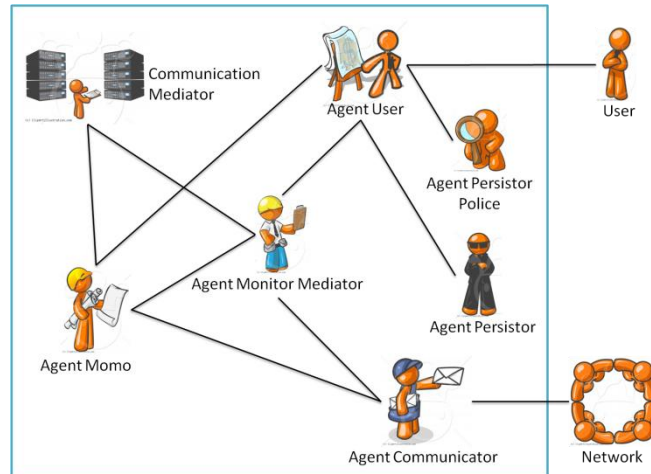


Figure 5-8 - MIRAI Architecture

In section 4.3 was explained how the MIRAI framework works, in this section it will be presented the setup of the agents, following the explanation of the section 4.7.3. In the Figure 5-8 is represented the six agents of this framework, it is possible to see how they are distributed and with whom they communicate. The MAS starts to work with the Agent Persistor that will create all the agents in the moment, then its life cycle is to monitor all the agents to see if they are alive, sending all the time informing messages to the Agent User to inform the user about the news. The Agent Persistor Police do the same thing, but only controls one specific agent, the Agent Persistor, since if this agent dies

there is no other agent to do his job. At this point all the agents are alive and doing their job, cooperate with each other to achieve the big goal of maintains the interoperability status.

5.2.2. Step 1 – Monitoring and Detection of Evolution of the Morphisms

The monitoring and detection of morphisms is the most important part of this work, to accomplish this goal two agents are involved, the Agent Monitor Mediator and the Agent MoMo, this section follows the explanation of the section 4.4.3. This process starts with the Agent Monitor Mediator searching for changes in the CM and ends with the Agent Momo saving the morphisms in the CM, in Figure 5-9 is the activity diagram of all the process, it shows how they work, and how they interact between them.

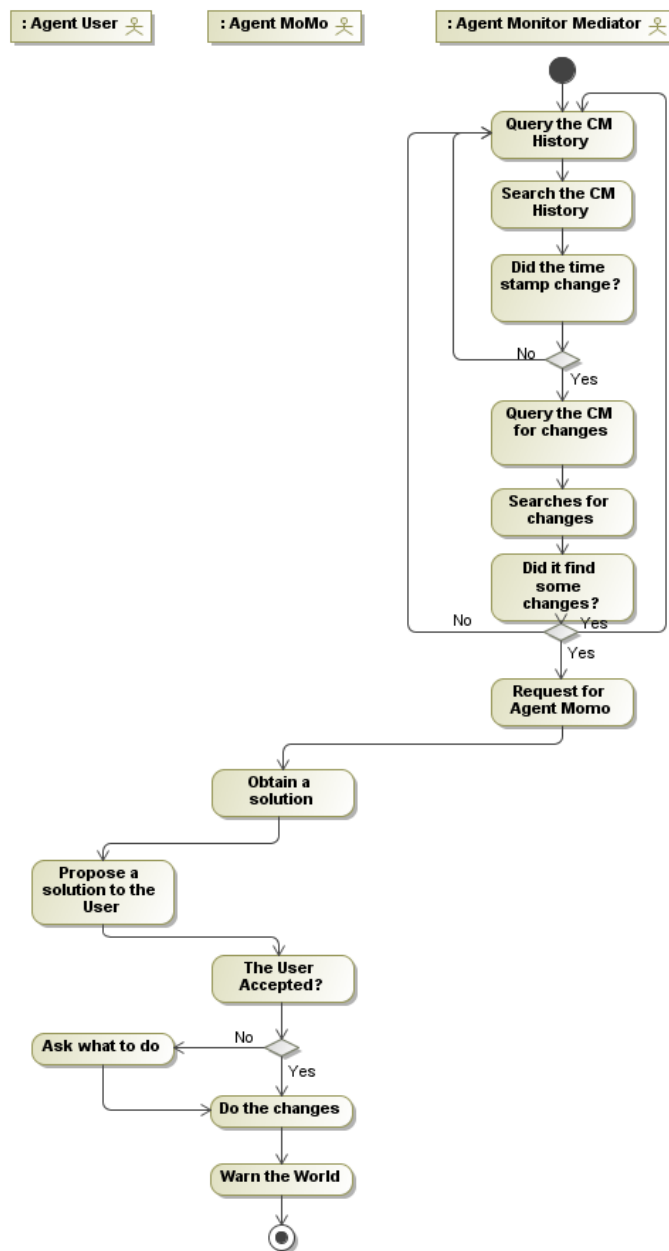


Figure 5-9 - Activity Diagram of Agent Monitor Mediator and Agent MoMo

Using the figure as a reference to explain the process, it is possible to verify that the process begins with the Agent Monitor Mediator doing a query to the CM with the aim to find when the time stamp of the file was modified. If it was modified it is because the file suffered some changes, so the next step is to query the CM searching for new morphisms. This is the routine of this agent, repeating this once a day. Every time that the agent found changes in the CM it will call for the Agent MoMo to help to diagnosis the problem. As it is possible to see, this part only cares about the search of new mappings, and taking the needed information to the other agent.

The 2nd part of the process only activates if the Agent Monitor Mediator asks for help, this begins with the Agent MoMo receiving a message from Agent Monitor Mediator with the list with the versioning's. The agent deals with one mapping at a time, evaluating the Mapping and the Evolution and proposing a Choice to the user, this process follows the explanation of the section 4.4. After proposing the Choice to the user, it has to wait for the validation, which can be accept or rejected. In the case that the user rejects the proposal the agent will ask to indicate a better solution. This process ends with the agent saving the changes in the CM.

At the end the Agent MoMo will send the list with the Choices to the Agent Communicator to send the message to the Network, with the aim of warning them, and with this avoiding a break of the harmonization. The technology involved in this step is the JADE to create the agents and libraries of the Protégé to help to access to the OWL file.

5.2.3. Step 2 – Propose a New Morphism

This step is about the proposal of a new morphism made by the Agent MoMo, when the agent diagnose the problem it will create a proposal to send to the user, this step follows the explanation of the section 4.4.4. In Figure 5-10 is about a mapping taken from the first scenario of the section 5.1.1, this mapping does the relation of the Weight of the Nacelle between the Manufacturer A and the Supplier B.

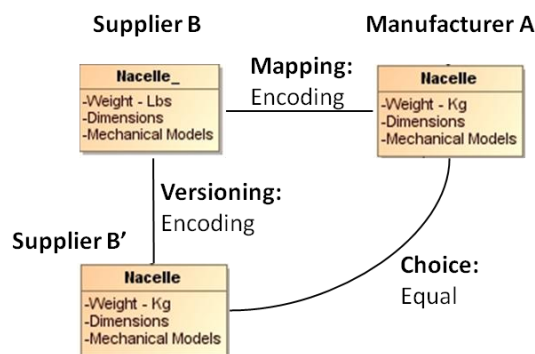


Figure 5-10 - Example of new MatchClass creation

First exists a mapping relating the two enterprises, where the Weight from A was in kilogram (Kg) and in B was in pounds (Lbs), this is a mismatch of the type of Encoding. Then B evolved to B' (versioning), and at this point the pre-existing mapping B→A needed to be reevaluated since it might no longer be valid. MIRAI detected the evolution and proposed a new mapping, relating B' with A.

This new mapping is called the Choice, like it is possible to see in the figure, and the Agent MoMo will create a new morphism using the 5-tuple mapping explained in section 3.4.5. This mapping have five fields that is needed to fill (Figure 5-11), some of them are directly without much of work, like the ID is a random number without repeating, the MElems are the two mapped entities that are depended of the Mapping and the Versioning, the KMType is the same as the two mappings, the Exp is evolved according with the MatchClass. The last field is the MatchClass, which is not directly and need some decision making and it will be explained in the next step.

ID	Model_X_1.1_2	
Melems = (a,b)	A	Weight - Kg
	B	Weight - Lbs
KMType	Instantiable Data	
MatchClass	Encoding	
Exp	"A \subseteq B"	

Figure 5-11 - Example of a mapping

5.2.4. Step 3 – Decision Making and the Creation of the New Morphism

The objective of this step is to explain how the decision in the creation of a new morphism is made by the Agent MoMo, this follows the explanation of the section 4.4.5. When the creation of a new mapping the big difficulty is about to choose what to put in the MatchClass, since normally this is like a function ($X+Y=Z$), in other words, the MatchClass of the Choice is depended of the MatchClass of the Mapping and the MatchClass of the Versioning. The problem in this matter is to decide the Choice because this is a little abstract is depended of the person that is choosing in that moment.

To help in this, it was created a learning ability to decide what the chosen MatchClass is, in the CM was created a table with all the possible solutions to this function, this list contains $11^3 = 1331$ entries, because this is not commutative. The author decided what the best result for all the possible answer is and set to 1 the correct ones, the rest are 0, this is the default. Every time that the Agent MoMo has to do a decision, it go to the list and choose the one with the higher weight. When the user accepts the answer the Agent MoMo will increment the weight by 1.

The great advantage of using this method is because in some cases it was verified that the choice presented by the Agent MoMo is not the best one, so the user will modify it. So the agent learns from it and the weight will increase, this will make that with time the choices will be different, adapting to each case, and creating a more customize system.

5.2.5. Step 4 – Communication between Enterprises

This section approaches the communication between different enterprises, following the explanation of the section 4.5.2 and explains the functionalities of the Agent Communicator. In earlier sections it was explained that every time the MIRAI finds a change it will warn the network about that, this is made by asking the Agent Communicator to communicate with the network.

In Figure 5-12 a scenario of the communications between different enterprises is presented, each enterprise has an active web service that is always available. So, every time the MIRAI A detects

changes in the CM, the Agent Communicator activates the web service of the all MIRAI on the network to let them know about those changes. Whenever the web service is activated the Agent Communicator will get the information and sends it to the Agent Monitor Mediator in order to verify if the updates have an impact in the CM, following the procedure explained previously.

A problem that was detected during the research and it will be done in future work, is when the MIRAI dies and take some time to restore it. The other MIRAI's continue to work and provide the changes, removing the oldest ones, this causes it to lose the previous changes. Every time that it loses an update, it will almost certainly going to cause a break in the harmonization. To solve this, there are several proposals, that will allow the problem to be solved, and is later detected or is made a history of the changes. These are the ideas to solve this issue, but a study to see what is the best solution to put in practice is needed.

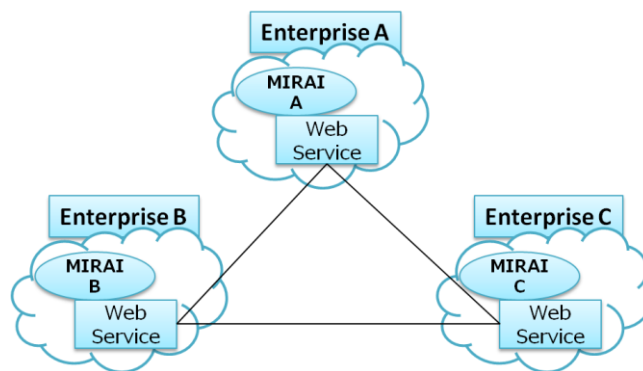


Figure 5-12 - Web Service Scenario

5.3. Implementation Overview and Technology Used

Until now the importance and how the MIRAI works was presented, at this point what was left to explain was the implementation and the technology used, this is the aim of this section. MIRAI is a framework to monitor the changes in the morphisms in the CM, to do the monitoring an agent is used, they are implemented using the technology JADE, which is based on JAVA. So six agents in JADE were made that will monitor the CM and send messages to the others organisations, two features to solve it appeared.

The 1st one is the monitoring of the CM that is an OWL file, this file was created in the Protégé. Then it was required to create a connection between the JAVA and the OWL, some libraries in JAVA were created using the Protégé. With this technology it was possible to monitor the OWL and modify the file every time changes were detected. The 2nd feature is about sending messages, the protocols described by FIPA were used, and they describe all the process to use in the messages.

To summarize the technology used throughout the proof-of-concept implementation, a short purpose of those is presented in the Table 5-1.

Table 5-1 – Purpose of the used technology by the Proof-Of-Concept Implementation

Technology	Purpose
JADE	Framework to develop agents technology
FIPA	Protocols to specify the communications between the agents
Protégé	Tool for construct models and knowledge-based applications with ontologies
OWL	Knowledge representation languages for authoring ontologies

5.3.1. Technology Used

In this section it will be described and explained the propose of use of the technology used in this dissertation. Like explained in previous sections, to implement the framework CAS-SIF it was decided to use MAS, because of the advantages of the agents. In the community exists different platforms of agents, with different specifications, but it was decided to use JADE, given it is the most widely used and is supported by JAVA, others advantages of JADE are explained in section 5.2.2.1.

Other used technology is the OWL format that defines the ontology of the CM, since the OWL is a standard. To implement the OWL it was used the Protégé and a more detailed explanation is made in section 5.2.2.3.

5.3.1.1. JADE

In (Fabio Bellifemine et al. 2001) Bellifemine et al. describe JADE (Java Agent Development Environment) as an Open Source software framework to make easy the development of agent applications in compliance with the FIPA specifications for interoperable intelligent multi-agent systems. The goal of JADE is to simplify development while ensuring standard compliance through a comprehensive set of system services and agents. To achieve such a goal, JADE offers the following list of features to the agent programmer:

- Foundation for Intelligent Physical Agents (FIPA) - compliant Agent Platform, which includes the AMS (Agent Management System), the default DF (Directory Facilitator), and the ACC (Agent Communication Channel). All these three agents are automatically activated at the agent platform start-up.
- Distributed agent platform. The agent platform can be split on several hosts. Only one Java application, and therefore only one Java Virtual Machine, is executed on each host. Agents are implemented as one Java thread and Java events are used for effective and lightweight communication between agents on the same host. Parallel tasks can be still executed by one agent, and JADE schedules these tasks in a cooperative way.
- A number of FIPA - compliant additional DFs can be started at run time in order to build multi-domain environments, where a domain is a logical set of agents, whose services are advertised through a common facilitator.

- Java API to send/receive messages to/from other agents; ACL messages are represented as ordinary Java objects.
- FIPA97 - compliant IIOF protocol to connect different agent platforms.
- Lightweight transport of ACL messages inside the same agent platform, as messages are transferred encoded as Java objects, rather than strings, in order to avoid marshalling and unmarshalling procedures.
- Library of FIPA interaction protocols ready to be used.
- Support for agent mobility within a JADE agent platform.
- Library to manage user-defined ontologies and content languages.
- Graphical user interface to manage several agents and agent platforms from the same agent. The activity of each platform can be monitored and logged. All life cycle operations on agents (creating a new agent, suspending or terminating an existing agent, etc.) can be performed through this administrative GUI (Graphical User Interface).

5.3.1.2. FIPA

The Foundation for Intelligent Physical Agents (FIPA) (FIPA 1999) is an international nonprofit association of companies and organisations sharing the effort to produce specifications for generic agent technologies. FIPA does not just promote a technology for a single application domain but a set of general technologies for different application areas that developers can integrate to make complex systems with a high degree of interoperability.

The first output documents of FIPA, named FIPA97 specifications, state the normative rules that allow a society of agents to exist, operate and be managed. First of all they describe the reference model of an agent platform: they identify the roles of some key agents necessary for managing the platform, and describe the agent management content language and ontology. Three mandatory roles were identified into an agent platform. The Agent Management System (AMS) is the agent that exerts supervisory control over access to and use of the platform; it is responsible for maintaining a directory of resident agents and for handling their life cycle. The Agent Communication Channel (ACC) provides the path for basic contact between agents inside and outside the platform. The ACC is the default communication method, which offers a reliable, orderly and accurate message routing service. The Directory Facilitator (DF) is the agent that provides yellow page services to the agent platform. Other specification is the Agent Communication Language (ACL), used by agents to exchange messages. FIPA ACL is a language describing message encoding and semantics, but it does not mandate specific mechanisms for message transportation.

5.3.1.3. Protégé

In (Knublauch et al. 2004) the author describe Protégé as an open-source tool developed at Stanford Medical Informatics. It has a community of thousands of users. Although the development of Protégé has historically been mainly driven by biomedical applications (Gennari et al. 2003), the

system is domain-independent and has been successfully used for many other application areas as well.

Like most other modeling tools, the architecture of Protégé is cleanly separated into a “model” part and a “view” part. Protégé’s *model* is the internal representation mechanism for ontologies and knowledge bases. Protégé’s *view* components provide a user interface to display and manipulate the underlying model.

Protégé’s *model* is based on a simple yet flexible metamodel (Noy et al. 2000), which is comparable to object-oriented and frame-based systems. It basically can represent ontologies consisting of classes, properties (slots), property characteristics (facets and constraints), and instances. Protégé provides an open Java API to query and manipulate models. An important strength of Protégé is that the Protégé metamodel itself is Protégé ontology, with classes that represent classes, properties, and so on.

6. TESTING AND HYPOTHESIS VALIDATION

In this section the implementation testing will be addressed, validating the requirements and functionalities of the system. Testing is the process of trying to find errors in a system implementation by means of experimentation. The experimentation is usually carried out in a special environment, where normal and exceptional use is simulated. The aim of testing is to gain confidence that during normal use the system will work satisfactory: since testing of realistic systems can never be exhaustive, because systems can only be tested during a restricted period of time, testing cannot ensure complete correctness of an implementation. It can only show the presence of errors, not their absence (Tretmans 2001).

Protocol conformance testing is a kind of testing where the implementation of a protocol entity is tested with respect to its specification. The aim is to gain confidence in the correct functioning of the implementation with respect to a given specification, and hence to improve the probability that the protocol implementation will communicate successfully with peer protocol entities (Tretmans 2001).

In the next sections will be presented some methodologies and the one which has been chosen to best approach the test definition applied to this particular proof-of-concept implementation. After some tests formalisation will be presented its results based on the performance of the various tested modules. Finally, in the last section a scientific and industrial context validation is presented.

6.1. Testing Methodologies

There are many testing methodologies available to test software engineering, many of them are abstract concepts like white / black / grey box testing, unit testing, conformance testing, etc. Testing in general but particularly in software testing (White 1987), (Myers 1979), functional and structural testing are distinguished from each other.

Structural testing is based on the internal structure of a computer program, where all program code is analysed and each line executed at least one time, covering all possible paths of execution. This type of analysis is also known as the white-box testing, where tests are derived from the program code. On the other hand, functional testing is about testing the externally observed phenomena of a program, regarding to its specification. Also known as black-box testing, in functional testing the functionality is evaluated by observing the box externally with no reference of its internal details or implementation at all. Since the functional tests are derived from the specification, the main goal is to analyse if the product is in fact working accordingly with the specification. Consequently, due to the nature of each of these testing methods, while structural testing is used in the early stages of the software development, functional tests are more often concentrated in the later stages of development.

Conformance testing is a kind of black-box testing, being only concerned with the correctness of a protocol implementation. This means that a developed software is evaluated regarding to its specification and its correct implementation, directly implying that correct, valid and clear specification

was provided in advance. Evaluating the correctness of a specification is referred to as “protocol validation” and involves checking that the implementation correctly behaves accordingly to the specification and the intended behaviour is indeed present. One problem regarding this procedure is inherently close to the specification: if it contains a design error and if the conformance testing process is correctly performed, each conforming implementation will have that same error (Tretmans 2001).

To define proved methods which apply these testing concepts, many standards were defined and revised throughout the years based on the expertise of using them and their practical results. On the other hand, sometimes a superimposition of a multitude of these is used in order to cover the necessities of a larger or more specific project. An example of this is the evaluation method and its definition which was used on the European Project iSurf (ISurf 2010), since it was based on the SQuaRE series of standards.

6.1.1. iSurf Functional and Non-Functional Evaluation Methodology

The iSurf European Project is integrated in the European Community's Seventh Framework Programme (FP7/2007-2013), and has as main objective the development of *“an environment [which] needs to be created to facilitate the collaborative exploitation of distributed intelligence of multiple trading partners in order to better plan and fulfil the customer demand in the supply chain”* (ISurf 2010). As a response to this need, the iSURF project (“An Interoperability Service Utility for Collaborative Supply Chain Planning across Multiple Domains Supported by RFID Devices”) provides a knowledge-oriented inter-enterprise collaboration environment to SMEs to share information on the supply chain visibility, individual sales and order forecast of companies, current status of the products in the manufacturing and distribution process, and the exceptional events that may affect the forecasts in a secure and controlled way.

The iSurf evaluation and testing framework follows the standard process defined on the evaluation reference model and guide ISO/IEC CD 25040 (ISO/IEC CD 25030 2007) of the SQuaRE series of standards and not limited to. Some of the used standards were: ISO/IEC 9126-1 (ISO/IEC 9126-1 2001), ISO/IEC 14598-1 (ISO/IEC 14598-1 1999), ISO/IEC CD 25010 (ISO/IEC CD 25010 2011), ISO/IEC CD 25030 (ISO/IEC CD 25030 2007), ISO/IEC 14598-5 (ISO/IEC 14598-5 1998), ISO/IEC CD 8402-1 (ISO/IEC CD 8402-1 1997) and ISO 9241 (ISO 9241 2006). ISO/IEC CD 25040 details the activities and tasks providing their purposes, outcomes and complementary information that can be used to guide a software product quality evaluation. The outcomes of applying a standard process approach for the evaluation activities in iSurf are the repeatability, reproducibility, impartiality and objectivity of all process.

Deliverable series 8 of the iSurf (ISurf n.d.) project present the principal standard steps for iSurf evaluation strategy (prepare, establish, specify, design, execute, report) and also describe in detail the procedures used to generate the evaluation criteria that were applied for the functional and non-functional characteristics (functionality, reliability, usability, efficiency, maintainability and portability). The project also identified the following techniques which were applied for evaluation of the iSurf components and architecture: functional tests, unit tests, fault tolerance analysis, user interface analysis, execution time measurements, documentation inspection and analysis of software

installation procedures.

These techniques and their evaluation criteria were modularised as recommended in ISO/IEC 25041 former ISO/IEC 14598-6 (ISO/IEC 14598-6 2001) in order to have a structured set of instructions and data used for the evaluation. It specifies the evaluation methods applicable to evaluate a quality characteristic (functional / non-functional) identifying the evidence it needs, defining the elementary evaluation procedure and the format for reporting the measurements resulting from the application of the technique.

Functional and non-functional evaluation criteria modules provide a flexible and structured approach to define criteria for monitoring the quality of intermediate products during the development process and for evaluation of final products. The purpose of using evaluation modules is to ensure that software evaluations can be repeatable, reproducible and objective.

These modules define a set structured instructions and data used for an evaluation. It specifies the criteria applicable to evaluate a quality characteristic and it identifies the evidence of it needs. It also defines the elementary evaluation procedure and the format for reporting the measurements resulting from the application of the technique.

The modules described specify the criteria for making the measurement as well as the preconditions and accuracy of the measurement. The aim is to make the various aspects (principles, metrics, activities, etc.) of evaluation visible and to show how they are handled. They are documented as specified on the standard ISO/IEC 14598-6:

1. Provides formal information about the evaluation module and gives an introduction to the evaluation technique described in the evaluation module;
2. Defines the scope of applicability of the evaluation module;
3. Specifies the input products required for the evaluation and defines the data to be collected and measures to be calculated;
4. Contains information about how to interpret measurement results.

The evaluation modules define the criteria for the evaluation of the iSurf components considering the functional and non-functional quality characteristics specified on the SQuaRE series of standards:

Functional:

- Functionality: Functional Test Cases;
- Functionality: Unit Tests.

Non-functional:

- Reliability: Fault tolerance Analysis;
- Usability: User interface;
- Efficiency: Execution time measurement;

- Maintainability: Inspection of development documentation;
- Portability: Analysis of software installation procedures.

6.1.2. Tree and Tabular Combined Notation (TTCN) – Test Notation Standard

The Tree and Tabular Combined Notation (TTCN (Vassiliou-Gioles 2007)(Ulrich & Grabowski 2004)) is a notation standardised by the ISO/IEC 9646-1 for the specification of tests for communicating systems and has been developed within the framework of standardised conformance testing. Based on the black-box testing model, the tests are defined through tables which are divided in general description, constraints, behaviour and verdict.

With TTCN, the test behaviour is defined by a sequence of events which represent the test *per se*. The sequence of events can be approached as tree, containing branches of actions based on evaluation of the system output after one (or a series of) executed event. Each event has its own respective level of indentation and can be of one of two types: action or question. Actions are preceded by an exclamation point before its brief description, and represent actions performed on the System Under Test (SUT). Questions are preceded by an interrogation point, and represent evaluations of the output of the SUT after one or more actions are completed. Since the answer can be positive or negative, multiple questions can exist at the same indentation level, covering all possible outputs of the system. After a completion of a TTCN test table a verdict must be deliberate: “Success”, “Failure” or “Inconclusive”. This verdict is based on the sequence of events which travel through the tree, and was conditioned by the outputs of the system and evaluated by the question events.

Depicted in Table 6- is a simplified example of a phone call can establishment evaluation. After a series of actions and evaluations (questions events) a different verdict is attained. The table can textually read as:

1. The user picks up the headphone;
2. Tests if the dialling tone is present;
3. If the dialling tone is present, then the user must dial the other phone’s number. Otherwise, if the dialling tone is absent, the verdict is a “Failure” of the possibility of establishing a phone call;
4. If there is a calling tone after dialling the number, the user may test if the line is in fact connected;
5. If the line is connected, the user may hung up the headphone and the verdict is set as “Success” on establishing a phone call, otherwise the verdict is a “Failure” of the possibility of establishing a phone call;
6. If the dialling tone is not heard, but a busy tone instead, then the user may hung up the headphone and the verdict is set as “Inconclusive” on establishing a phone call;
7. If none of the tones corresponds to calling or busy, then the verdict is set as “Failure” on

establishing a phone call.

Table 6-1 - Simplified example of a TTCN table test

Test Case		
Test Case: Basic Connection		
Group:		
Purpose: Check if a phone call can be established		
Comments:		
Behaviour	Constraints	Verdict
! Pick up headphone		
? Dialling tone		
! Dial number		
? Calling tone		
? Connected line		
! Hung up headphone		Success
OTHERWISE		Failure
? Busy tone		
! Hung up headphone		Inconclusive
OTHERWISE		Failure
? Dialling tone absent		Failure

Some more examples and tutorials are available at the TTCN-3 website (ETSI 2008)

6.1.3. Adopted Test Methodology

Section 5 addresses the section 4 frameworks' implementation design and structure, but it is intended to be a proof-of-concept of the framework. Unlike a commercial product, the proof-of-concept is not supposed to be flawless and a complete solution, but a working proof of feasibility of a full solution. This way, by applying such a complex test methodology as the one applied on the iSurf project only does not make sense, since it is too expensive for such kind of implementation. With this, a mix of validation tests was chosen in order to try to validate the proof-of-concept implementation.

Based on the iSurf test methodology and the TTCN tables proposed by ISO/IEC 9646, a series of functional test cases and unit tests described by TTCN tables were designed and applied to the various units of the implementation steps (depicted in Figure 4-2). On the other hand, non-functional tests such as reliability, efficiency and portability were also addressed. All the results and tests definitions are published in section 6.3.

6.2. Requirements and Functionalities

In this section the requirements and functionalities of the system will be presented. These were defined during the framework design and were the main objectives of a full implementation of the framework. In order to evaluate the extent of the proof-of-concept implementation, a mapping between

the requirements and functionalities of the system and the implementation are presented:

Requirements:

- **The architecture of the system should use the agent technology:** The MIRAI Architecture developed is composed by six agent cooperating each other;
- **The system should have a Communicator Mediator able to represent all foreign models of the network, regardless of the type of model:** In each enterprise a CM where is possible to represent a model with morphisms exist, with the morphisms is able to represent any kind of model, facilitating the representation;
- **The system should have all foreign models represented in the Communicator Mediator, represented with morphisms:** In a network to exchange data between the enterprises is needed, so it is important to have all the foreign models represented in the CM of the enterprise;
- **The system should be able to identify changes in the system and propose a solution to the user and in the end save the new mapping in the Communicator Mediator:** It's important to have a system that is able to adapt to the evolution of the models, so the system is able to identify changes in the CM and propose solutions to the user, and save the new mapping's in the CM with the consent of the user;
- **The system should be able to identify changes in the CM:**
 - **Identify the versioning's in the CM:** The system is able to scan the CM seeking for new versioning's in the CM;
 - **Obtain a solution to solve the versioning:** Is able to look at the old mapping and the new versioning and obtain a solution of a new mapping;
 - **Proposition of a solution to the user:** is able to propose a solution to the user, and will wait for the decision of the same. In case that the user refuse the proposition, it is able to ask the user for a new solution;
 - **Save the new mapping:** After the user chooses what will be the new mapping, the system is able to save in the CM.

Functionalities

- **Architecture of the system should be developed with agents:** through the execution of Step 1 of the proof-of-concept implementation it is possible to see the architecture developed with JADE and constitute by the six agents;
- **Identify changes in the Communicator Mediator:** through the execution of Step 2 of the proof-of-concept implementation it is possible to search for the versioning's in the CM;
- **Propose a solution to that change:** through the execution of Step 3 of the proof-of-concept implementation is able to do the decision of the mapping;

- **Save the mapping in the Communicator Mediator:** through the execution of Step 3 of the proof-of-concept implementation is able to save the mapping in the CM.

Through the analyses of these mappings it is possible to conclude that, all the requirements and functionalities are presented on the proof-of-concept implementation.

6.3. Testing

To address the functional and non-functional testing of the proof-of-concept implementation, all the step that were explained in the section 5.3 will be demonstrated and followed by showing how the MIRAI works. The following sub-sections will demonstrate four steps and show results of the tests made in, the functional tests will be presented with the Table 6-1, and the non-functional tests is about the reliability of the system, that is about the ability of the software to perform a required function under given conditions for a given time interval.

6.3.1. Step 0 - MIRAI Setup

To test the MIRAI a simple example was made, and that is represented in the Figure 6-1, this example is about a data base of a family, represented in two different enterprises that use different models, one uses the EXPRESS and the other the OWL. In the Enterprise A the family has the Name of the person, the field Family and the sex, the Enterprise B has the same but represents it in a different manner, along with the hasPet.

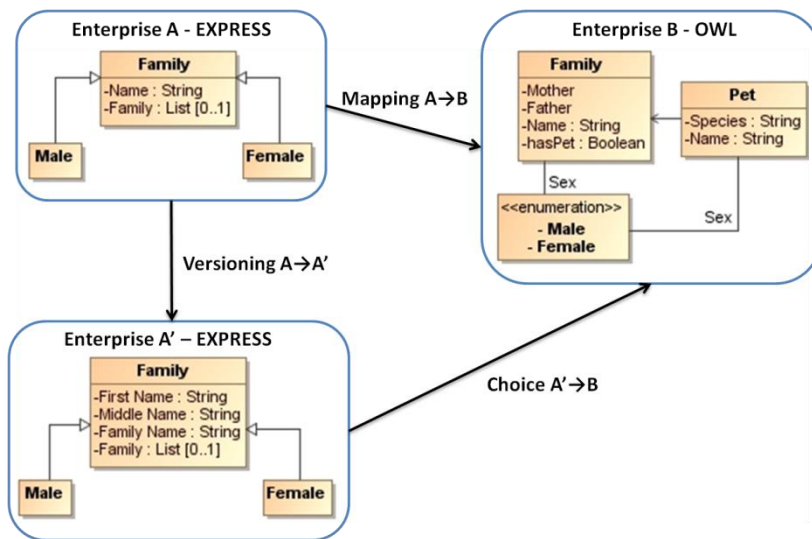


Figure 6-1 - Family example

To connect these two enterprises it is needed to relate all the fields, which are made in the CM with mappings $A \rightarrow B$, an example of a mapping is in the Figure 6-2. This mapping relates the Name in the EXPRESS of the A with the Family name in the EXPRESS of the A', is a mismatch of the type granularity, since the information is decomposed.

Then the Enterprise A suffers an evolution in the model $A \rightarrow A'$, the Name in EXPRESS is divided in three names: First, Middle and the Family name. In this case an intervention of the user was

required, to create the mappings about this change, but this created a break in the harmonization. Until now, it was preparing the example to test the MIRAI, next will be on trial.

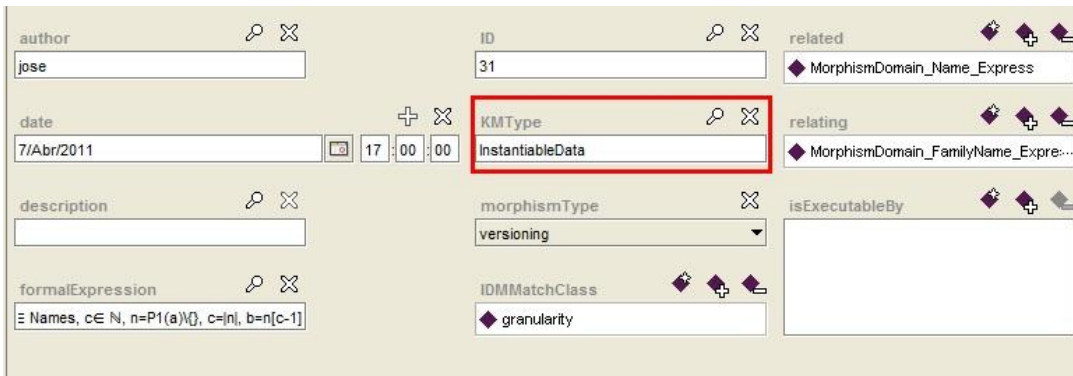


Figure 6-2 - Mapping relating Name in EXPRESS with Family Name in EXPRESS 2

The first thing that the MIRAI do is to create all the six agents, and create the GUI that will help the user in the process, in the Figure 6-3 it is possible to see the GUI. The red square shows the messages from these two Agent Persistor, it shows when an agent is created or when it dies, the blue square shows the same, the green squares are the agents that are alive, the red ones are dead. Here it is possible to see the step 1 represented in the MIRAI, all the agents are working together to achieve one goal, detect changes in the morphisms.

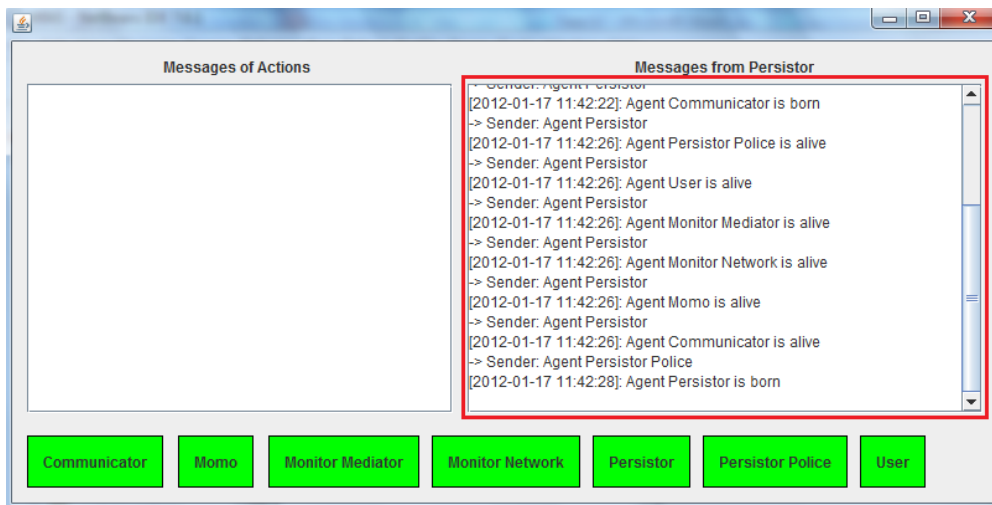


Figure 6-3 - MIRAI GUI and creating Agents

To test the Agent Persistor the system was forced and killed the Agent Monitor Mediator, to see if the system would recover and thus continue to work, in the Figure 6-4 in the red square it is possible to see the agent dying and then the Agent Persistor resurrect him. Several tests to search for fails were made, and a test by killing the Agent Persistor in order to verify if the Agent Persistor Police resurrect him or not was made. Through the various tests the system always recovered, thus achieving the desired goal to keep the interoperability status.

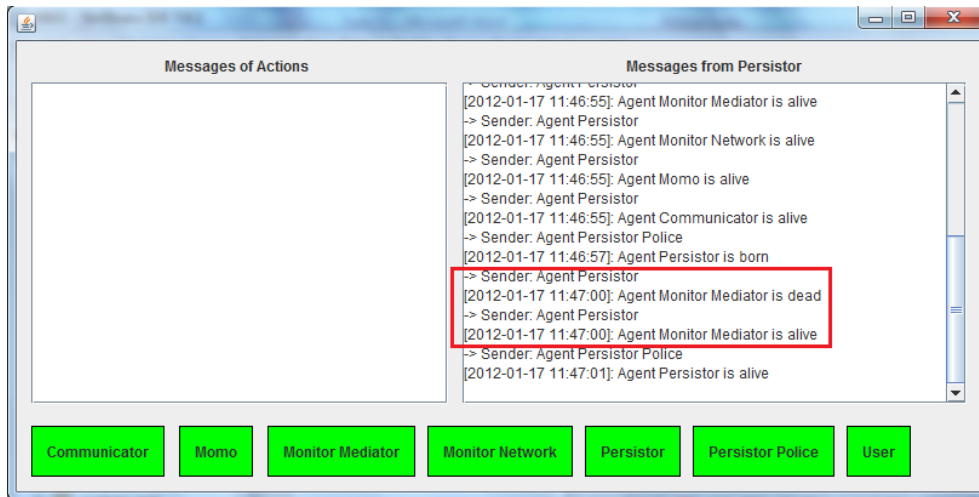


Figure 6-4 - Agent Persistor resurrecting a Dead Agent

Now the results of the tests will be presented, the first one is the functional test and it is divided in two tests. The first is to test the Agent Persistor, it was made several tests in this matter, since to kill one agent to all of them, but in the end the results were all the same, the agent was successful to resurrect the others agents, in the Table 6-2 is the results of the tests.

Table 6-2 - Agent Persistor functional test

Test Case		
Test Case:	Agent Persistor	
Group:	Resurrect agents test	
Purpose:	Check if the agent resurrect the agents	
Comments:	Step 0	
Behaviour	Constraints	Verdict
! Receive message from AMS		
? An agent died		
? What agent		
! Resurrect the agent		
? The agent is alive		Success
OTHERWISE		Success
OTHERWISE		Success
OTHERWISE		Success

Next it will be tested the Agent Persistor Police following the same procedure, but in this case it was killed the Agent Persistor, and in some cases the Agent Persistor and another agent, to checks how the system reacts, in the Table 6-3 is the results of the tests.

Table 6-3 - Agent Persistor Police functional tests

Test Case		
Test Case:	Agent Persistor Police	
Group:	Resurrect the Agent Persistor	
Purpose:	Check if the agent resurrect the Agent Persistor	
Comments:	Step 0	
Behaviour	Constraints	Verdict
! Receive message from AMS		
? An agent died		
? Is the Agent Persistor		
! Resurrect the Agent Persistor		
? The Agent Persistor is alive		Success
OTHERWISE		Success
OTHERWISE		Success
OTHERWISE		Success

At this point only the non-functional tests are missing, this test shows how long it took the system to fulfil their task. Here it is possible to see how much time it was needed for the MIRAI to react and resurrect the agent, three types of tests were made. In the case of Agent Persistor one agent was killed, then two agents were also killed, the reaction of the agent was good, it took less than one second. In the case of the Agent Persistor Police it can say that was two tests in one, since first it was killed the Agent Persistor, but in other two tests it was the Agent Persistor and other agent, the results were good again, and again it took less than one second.

Table 6-4 - Agents Persistor's non-functional tests

		Test 1	Test 2	Test 3
Persistor	Agent Killed	Persistor Police	Monitor Mediator	Monitor Mediator + MoMo
	Time to Resurrect	less 1 second	less 1 second	less 1 second
Persistor Police	Agent Killed	Persistor	Persistor + MoMo	Pesistor + Monitor Mediator
	Time to Resurrect	less 1 second	less 1 second	less 1 second

6.3.2. Step 1 - Monitoring and Detection of Evolution of the Morphisms

In the Figure 6-5 it is possible to see the step 1 working, in the red square are the actions made by the Agent Monitor Mediator. Following the Activity Diagram of the Figure 5-9, the first thing that the agent does is to verify the time stamp and then search for new morphisms. So, in the red square it is possible to see that first the agent detected changes the time stamp and then starts searching for the versioning's in the CM. The last message is about the Agent Monitor Mediator sending the detected versioning's to the Agent MoMo, and in consequence active it.

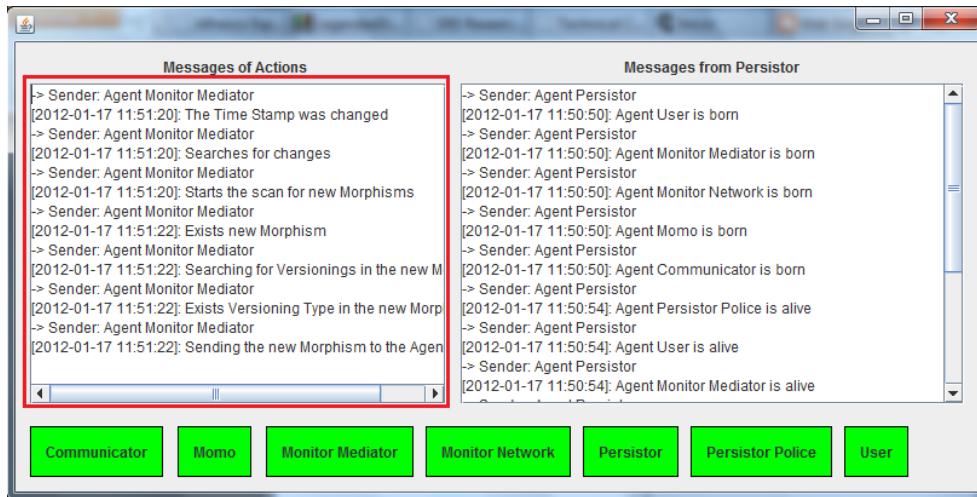


Figure 6-5 - Agent Monitor Mediator working

The rest of this section is to describe the tests and the results, the first is introduced in the Table 6-5 that shows the results of the functional test, that tests the Agent Monitor Mediator. In the beginning of this sub-section it was presented an example that was created and added to the CM to use in the simulation, then it was made an evolution to that models. At this point the MIRAI starts to search for changes, and will send the versioning's to the Agent MoMo, all this process is represented in the table.

Table 6-5 - Agent Monitor Mediator functional test

Test Case		
Test Case:	Agent Monitor Mediator	
Group:	Alone	
Purpose:	Check if exist changes in the CM	
Comments:	Step 1	
Behaviour	Constraints	Verdict
! Once a day search for changes in the CM		
? Time Stamp changed		
? Exist versioning's in the CM		
? Was detected versioning's		Success
OTHERWISE		Success
OTHERWISE		Success
OTHERWISE		Success

The other test is the non-functional test that evaluate the time that the MIRAI take to execute the step 1. Using the same example about families, it used the MIRAI to detect changes, in the system it was introduced seven versioning's. It was tested three times to see how much time it takes, in the Table 6-6 is possible to see the results, taking in average 3 seconds to find that versioning's.

Table 6-6 - Agent Monitor Mediator non-functional test

	Test 1	Test 2	Test 3
Monitor Mediator	2 seconds	3 seconds	3 seconds

6.3.3. Step 2 & 3 - Propose, Decision Making and Creation of the New Morphism

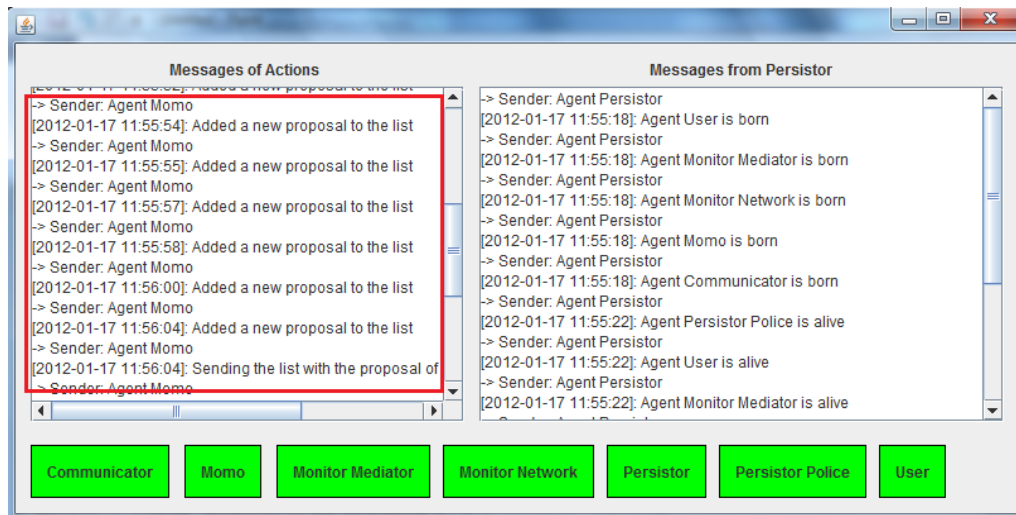


Figure 6-6 - Agente MoMo proposing a possible solution

Every time that the Agent MoMo receives messages with new versioning's it will start it's work, continuing to follow the Activity Diagram of the Figure 5-9, thus initiating step 3, where decision to create the new morphism is made. In the Figure 6-6 (red square) it is possible to see Agent MoMo working, every time that the agent evaluates the mapping and the versioning, it proposes a new morphism, and sends a message to the Agent User informing of that, when it finishes to evaluate all the versioning's that it finds, it will then send the list to the Agent User, so that the user can give its approval for the proposed.

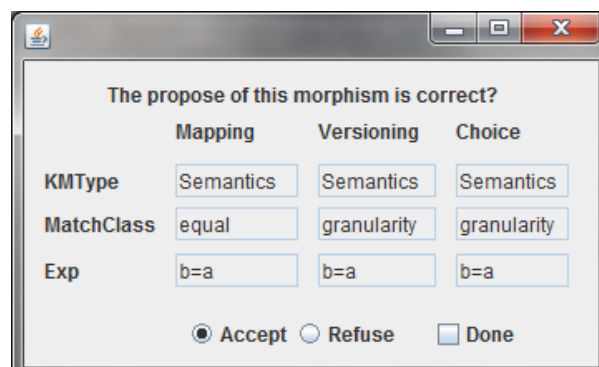


Figure 6-7 - Proposal of a new morphism

The next step is step 3, where the user will decide which mapping will be used to solve the problem of interoperability. Here the user has two options, accept the proposal of the Agent MoMo or refuse it, in the Figure 6-7 is an example of a propose made by the Agent MoMo, as is possible to see,

the user can accept or refuse the proposal, after the user answer the question the Agent User will sent it to the Agent MoMo and then the agent will see what to do.

In case of accepting the proposal the Agent MoMo will save the new morphism in the CM, if it was refused the agent will ask the user a better solution, that is possible to see in the Figure 6-8, allowing the last column to change and with that the user choose the best one. After that, the agent will save the new mapping in the CM, and with this maintain the interoperability status.

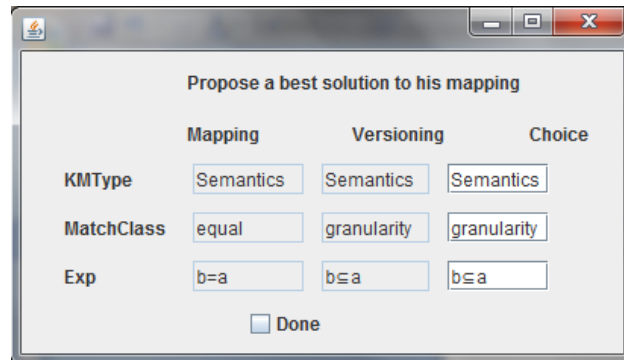


Figure 6-8 - Asking to the user to choose solution

To finish this evaluation there are tests that are missing. In the Table 6-7 is presented the results of the functional test, it is possible to see all the process execute by the Agent MoMo, starting with the evaluation of the morphisms until the saving of the morphisms in the CM.

Table 6-7 - Agent MoMo functional test

Test Case		
Test Case:	Agent MoMo	
Group:	Alone	
Purpose:	Find a solution and propose to the user	
Comments:	Step 2 & 3	
Behaviour	Constraints	Verdict
! Receive a list of versioning's to evaluate		
! Evaluate the versioning		
? Is a problem		Success
! Obtain solution		
! Propose to the User		
? Accept		Success
! Save new morphism		
? Refuse		Success
OTHERWISE		Success
OTHERWISE		Success

The last test is the non-functional test of the Agent MoMo, in this case it was counted the time needed to propose the new morphisms. This time started since the time that the Agent MoMo received the versioning's until the time when it propose to the user, in the Table 6-8 is possible to see that it takes more or less than 11 seconds.

Table 6-8 - Agent MoMo non-functional test

		Test 1	Test 2	Test 3
MoMo	Good Proposal	yes	yes	yes
	Time to Propose (7 Morphism)	11 seconds	12 seconds	11 seconds

6.4. Hypothesis Validation

In section 1.4 the hypothesis of this dissertation was defined, and the objectives of this work were presented, the question made in the hypothesis was achieved during this dissertation. The MIRAI framework fulfilled the main objectives and it was able to monitor and detect changes in the system model, proposing in the moment a solution to recover the interoperability status, and making the final decision to the user. The last point of this hypothesis was to create an interoperability network using web services for communication between enterprises giving them the changes that were made at the time, at this point this was presented as an idea to implement, but a needed to test and validate is required.

6.5. Scientific Validation

Two scientific publications were made to validate the proposed, the 1st was published in the OnTheMove 6th International Workshop on Enterprise Integration, Interoperability and Networking (EI2N 2011), from the 21th to 23th of October 2011 in Crete - Greece, the 2nd was accepted to the 14th IFAC Symposium on Information COnTrol Problems in Manufacturing (INCOM 2012), from the 23rd to 25th of March 2012 in Bucharest - Romania. Below are the two papers following the earlier description:

- Ferreira J., Agostinho C., Sarraipa J. and Jardim-Goncalves R., "Monitoring Morphisms to Support Sustainable Interoperability of Enterprise Systems", Accepted in: OnTheMove 6th International Workshop on Enterprise Integration, Interoperability and Networking (EI2N 2011). Oct 21-23, Crete, Greece, 2011.
- Ferreira J., Agostinho C., Sarraipa J. and Jardim-Goncalves R., "Monitoring Morphisms to Support Sustainability of Interoperability in the Manufacturing Domain", Accepted in: 14th IFAC Symposium on Information COnTrol Problems in Manufacturing (INCOM 2012). May 23-25, Bucharest, Romania, 2012.

6.6. Industrial Validation

Industrial acceptance and validation is equally important as the acceptance by peers. If industry does not approve the results, most likely they will never be used. This way, since the beginning, research has been scoped along two industry-led initiatives from two different sectors. Each of them has its own prototyping scenario, as described in the following sub-sections.

6.6.1. CRESCENDO's Behavioural Digital Aircraft

This dissertation is contributing to the developments proposed in the project CRESCENDO, that is a large scale research project involving 62 partners organisations across the EU and funded by the European Commission since May 2009 for the duration of 3 years (CRESCENDO 2009). This research has the initiative to answer to some problems existing in a large enterprise-ruled sector which are already used standards in the product data.

The customers each day demand for better and more complex products, forcing the market expectations demanding a more efficient aircraft behaviour, and in result forcing the enterprises to develop the products in a shorter time and with a more cost effectiveness. These force the enterprises to be global and in answer to this, the aircraft manufacturer & suppliers need collaboration solutions to work together in multi-disciplinary teams across the extended enterprise. These collaboration solutions have to follow all the development of the PLC, with good management to avoid rework and loss of time and money. One solution to avoid to do some rework is making often an analysis to the system to eliminate risks during all the phases and being able to accurately predict functional behaviour. Another thing reduces the time consumption of time, and in consequence of money, is to simulate the products before it goes to the market, to avoid problems. These are the challenges faced by the aeronautical industry and CRESCENDO proposes to solve this situation with the Behavioural Digital Aircraft (BDA).

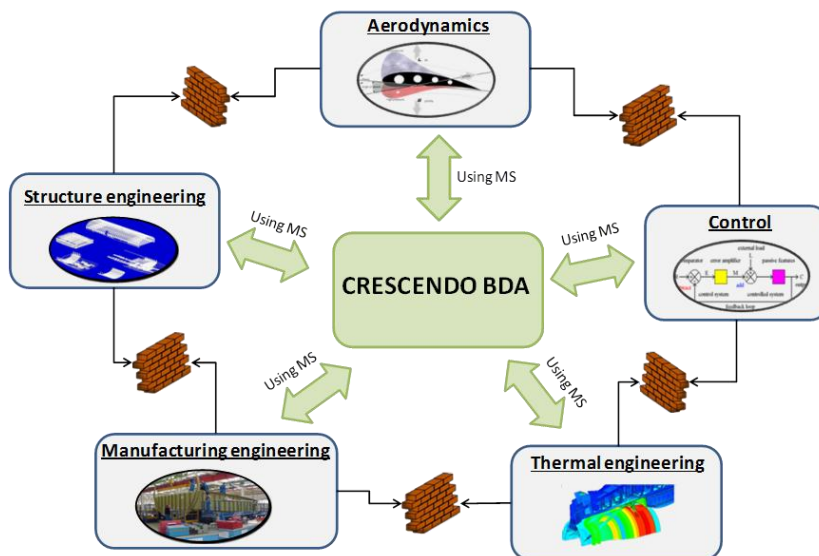


Figure 6-9 - Current barriers and envisaged CRESCENDO solution

Among the several components of the BDA, the Model Store (MS) is the core and will be the result that satisfies the CRESCENDO objective to organize modelling and simulation for

multidisciplinary purposes. It proposes a common architecture of models that can be configured to provide the complete behavioural and functional view of the product, and will be an enabler to organize and share all models in the Extended Enterprise. The Model Store will deliver innovation in terms of multi-level, ontology-based model architectures, supported by a semantic technology based modelling and simulation dictionary, together with adapted modelling languages, management and control processes. These results will allow models associativity, evolution and context to be managed more effectively throughout the lifecycle. More specifically, the Model Store results will illustrate (CRESCENDO 2010):

- How to link models developed for different purposes (or by different organizations), in a secure, federated, retrieval and storage system, so that the combined models can be used to simulate larger applications;
- How to link models in a hierarchical manner to provide consistent views of a product at differing levels of abstraction or life cycle stages;
- How to allow a model to understand its own behaviour, input needs, and output capabilities, such that when a new element is added to the system, it will negotiate with the models of all the other elements of the system and fit in the overall common models architecture.

6.6.2. Contribution of this dissertation

The contribution of this dissertation to the CRESCENDO project is to help to solve the situation with the BDA, using the CAS-SIF framework so the system can be capable of autonomously adapting to environmental changes in the form of new requirements that force to change simulation models and sometimes, even data models caused by software evolutions. The MIRAI being an implementation of the CAS-SIF is able to detect problems in the models and to readjust those models. This way the MIRAI is an answer to one of the challenges faced by the aeronautical industry, and this work answer to making the analysis to the system to eliminate risks during all the phases avoiding some rework. This is made with a constant monitor to the models avoiding unknown changes that will create problems in the near future.

7. FINAL CONSIDERATIONS AND FUTURE WORK

The practice of systems engineering has been transitioning from a document-based approach to a model-based approach like many other engineering disciplines. MBSE offers significant potential benefits to specification and design precision, it helps to improve design quality, productivity, and reduce development risk. The emphasis in MBSE is on producing and controlling coherent system models, and using them to specify a complete product throughout its lifecycle (Ogren 2000).

In the last years some languages for modelling systems that support MBSE have been developed. These support modelling of requirements, structure, behaviour and parametric to provide a robust description of a system (Friedenthal et al. 2008). Until now, the best way to deal with interoperability depended on the usage of dedicated knowledge models and international standards acting as information regulators (Carlos Agostinho & Ricardo Jardim-Goncalves 2009). Yet, for different reasons some enterprises remain using traditional models for engineering analysis and simulation models, where each one tends to use its own data format and business rules, and handles as many mappings as the number software tools it needs to execute. This brings some issues in interoperability, becoming difficult to handle existing mappings, even when following MBSE paradigm.

The MBSE is a good implementation that supports the systems engineering processes, more exactly, to support all the life cycle of the product that covered all kind of the processes. This was a great achievement to the enterprises, because it is a step to reach interoperability, however it is not enough. Some authors start to describe the interoperability in levels or layers, so that the enterprises knows how much interoperable they are, being given a level and the higher the level the more interoperable is the enterprise, in the last years, several proposals of interoperability layers and maturity levels of different authors with the aim to specify when an enterprise is not interoperable until it achieves a full interoperability status.

Another problem is in the creation of a collaborative network, an enterprise has the need to work with other enterprises. In some cases issues appear, even if the enterprises have a good interoperability. The easiest to identify is the problem of the language, which is a problem to solve, since it can be possible to define a reference language to communicate between them, normally English. However, in CN environments it can reach a big puzzle of heterogeneity between the enterprises systems and applications, causing serious problem in the interoperability.

Knowing the level of enterprise interoperability helps to know how much effort the enterprise has to do to reach a good interoperability status. At this point it is required to get down to business and create solutions to help in this matter. In the Enterprise Interoperability research community different solutions appear, which in this dissertation it was decided to implement the CAS-SIF proposed by Agostinho and Jardim-Goncalves (Carlos Agostinho & Ricardo Jardim-Goncalves 2009). They proposed a framework to implement a sustainable interoperability system. In order to reach that goal the framework proposes to follow four layers. Then, after the full implementation of these four layers, the enterprises will achieve a very good interoperability status, and more important being able to

maintain that status.

The MIRAI is an implementation of this framework, covering two of the four layers, i.e., one is the Monitoring and the other is the Integration Intelligence Layer.

The role of MIRAI is to monitor all the mappings that exist among the several models used by business partners in the same collaborative network, controlling the changes, warning and proposing new mappings. Thus, preventing interoperability problems that could cause a destabilization of the network harmony. Enterprises' privacy is assured since each one has its own MIRAI associated to an internal CM that tracks the morphisms it maintains with their direct partners.

Such solution facilitates the creation of a network sustainable interoperability. This means that the systems are self-organized and capable of responding to environment changes, and network evidence system of systems behaviour. Systems' communication with the environment allows all necessary exchanges through its own boundaries, even though transformation mechanisms are influenced by internal rules, values, beliefs, constraints, culture, and internal models. This dynamicity is essential to support and sustain the interoperability applied to global business networks.

To create a network sustainable interoperability, four critical scenarios to simulate a collaborative network were identified: 1st scenario is about an evolution in a morphism in one of the enterprises; 2nd scenario about an entry of a new enterprise in the network; 3rd about a new connection between two enterprises; and the last one is about a removal of an enterprise of the network. The MIRAI in this dissertation was validated for the 1st scenario that is about an evolution in morphism, and being prepared to implement the rest of the scenarios.

Another issue that normally appear in a network is the difference in the culture, this makes the decisions to change depending of the person involved. To solve this, the MIRAI is able to adapt and learn from the user over time, gaining a more reliable decision to the user, being more customizable.

7.1. Future Work

For future work, the author intends to enhance the proposed framework with functionalities able to manage the dynamic inclusion or exclusion of enterprises in the network, and to manage with new connection between nodes that are already in the CN. With these functionalities combined with an automatic detection of evolutions stored in the CM knowledge base. With all these functionalities, the MIRAI will be able to follow all the LC of the CN. Starting with the formation of the collaboration, until the disposal of the same, being capable to add or remove the enterprises without human intervention.

Also, the Agent *MoMo* has space for improvement, enabling it with capabilities to analyse the network transients and provide more knowledgeable response. Another improvement is to give a learning ability to Agent *MoMo*, turning it possible to learn from the past, avoiding mistakes in the future.

Other improvement to implement is in the Administrator Block, where in the Figure 4-2 is already represented, and is called the Decision Maker. This part of the Administrator Block has the aim to help the user in the decision to choose the new morphism (Choose). This new part learns with

the answer of the user and with time starts to have a more reliable decision, avoiding intervention of the user.

Also, it is considered to implement the step 4 of the section 5.2.5, that is the communication between the different MIRAI's, solving the issue that was explained in the same section.

8. REFERENCES

- ATHENA, 2010. MDI - Model Driven Interoperability.
- ATHENA IP, 2007. ATHENA Public Deliverable D.A4.2 - Specification of Interoperability Framework and Profiles, Guidelines and Best Practices.
- Agostinho, C., Malo, P., et al., 2007. Harmonising Technologies in Conceptual Models Representation. , 2(2), pp.187-205.
- Agostinho, Carlos, Sarraipa, Joao, et al., 2007. Enhancing STEP-Based Interoperability Using Model Morphism. *3rd International Conference Interoperability for Enterprise Software and Applications (I-ESA2007)*.
- Agostinho, Carlos et al., 2011. Tuple-Based Semantic and Structural Mapping for a Sustainable Interoperability. In *Ifip International Federation For Information Processing*. pp. 45-56.
- Agostinho, Carlos, Correia, F. & Jardim-Goncalves, Ricardo, 2010. Interoperability of Complex Business Networks by Language Independent Information Models. In *17th ISPE International Conference on Concurrent Engineering CE2010*. Springer-Verlag. Available at: <http://www.springerlink.com/content/q775010855762526/>.
- Agostinho, Carlos & Jardim-Goncalves, Ricardo, 2009. Dynamic Business Networks : A Headache for Sustainable Systems Interoperability. In *4th Int. Workshop on Enterprise Integration, Interoperability and Networking (EI2N'2009) of the OTM Conference*.
- Athena IP, 2004. EC FP6-507849.
- Beca, M.F. et al., 2011. Tuple-Based Morphisms For E-Procurement Solutions. In *ASME 2011 International Design Engineering Technical Conference Computers and Information in Engineering Conference IDETCCIE 2011*.
- Bellifemine, F., Caire, G. & Greenwood, D., 2007. *Developing Multi-Agent Systems with JADE*,
- Bellifemine, Fabio, Poggi, A. & Rimassa, G., 2001. Developing multi-agent systems with a FIPA compliant agent framework. *Software: Practice and Experience*, 31(2), pp.103-128. Available at: [http://doi.wiley.com/10.1002/1097-024X\(200102\)31:2<103::AID-SPE358>3.0.CO;2-O](http://doi.wiley.com/10.1002/1097-024X(200102)31:2<103::AID-SPE358>3.0.CO;2-O).
- Bernstein, P.A., 2003. Applying Model Management to Classical Meta Data Problems. In *1st Biennial Conference on Innovative Data Systems Research*. California.
- Bourey, J.-P. et al., 2006. Deliverable DTG2.2 - Report on Model Interoperability. *InterOP*.
- Bourey, J.-P. et al., 2007. Deliverable DTG2.3 - Report on Model Driven Interoperability. *InterOP*.
- C4ISR, 1998. Levels of Information Systems Interoperability (LISI). , (March).
- CRESCENDO, 2010. *BDA Model Store Specifications Dossier – (intermediate version M12). Public Deliverable D5.2.4.*,
- CRESCENDO, 2009. *EU FP7 - PROJECT NO.:234344*,
- Camarinha-Matos, L., 2010. Scientific Research Methodologies and Techniques - Unit 2: Scientific Method. *Phd Program in Electrical and Computer Engineering*.

- Chestnut, H., 1965. *Systems Engineering Tools*, Wiley.
- Correia, F., 2010. *Model Morphisms (MoMo) to Enable Language Independent Information Models and Interoperable Business Networks*.
- Dauben, J.W. & Cantor, G., 1979. *His Mathematics and Philosophy of the Infinite*, Harvard University Press.
- Davenport, T.H., 1998. Putting the enterprise into the enterprise system. *Harvard business review*, 76(4), pp.121-31. Available at: <http://www.ncbi.nlm.nih.gov/pubmed/10181586>.
- ETSI, 2008. TTCN-3 Tutorials. Available at: <http://www.ttcn-3.org/Tutorials.htm> [Accessed January 15, 2012].
- Ehrig, M. & Staab, S., 2004. QOM - Quick Ontology Mapping. In *3rd International Semantic Web Conference (ISWC2004)*. pp. 1-28.
- Elvesæter, B. et al., 2006. Towards an Interoperability Framework for Model-Driven Development of Software Systems. *Framework*, pp.409-420.
- Estefan, J.A., 2007. *Survey of Model-Based Systems Engineering (MBSE) Methodologies*, California.
- FIPA, 1999. Foundation for Intelligent Physical Agents. Available at: <http://www.fipa.org/> [Accessed September 16, 2011].
- Faulconbridge, R.I. & Ryan, M.J., 2005. *Engineering a System: Managing Complex Technical Projects*, Argos Press Pty LTD.
- Ferreira, J. et al., 2012. Monitoring Morphisms to Support Sustainability of Interoperability in the Manufacturing Domain.
- Friedenthal, S., Moore, A. & Steiner, R., 2008. *A Practical Guide To SysML: The Systems Modeling Language*,
- Frisch, H.P. et al., 2007. ISO STEP-AP233 Transition Development to Enablement. *AP233 System Engineering*.
- Gennari, J.H. et al., 2003. The Evolution of Protégé - 2000: An Environment for Knowledge-Based Systems Development. *International Journal of Human-Computer Studies*, 58(1), pp.89–123.
- Geraci, A. et al., 1991. *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*, IEEE Press.
- Van Gigch, J.P., 1991. *System Design Modelling and Metamodeling*, Springer.
- Hailpern, B. & Tarr, P., 2006. Model-Driven Development : The Good, the Bad, and the Ugly. *IBM Systems Journal*, 45(3), pp.451-461. Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5386628>.
- Haskins, C., 2006. *Systems Engineering Handbook*,
- Hassell, L., Fogler, M. & Russell, S., 2006. Collaborative Networks: Helping Rural Laboratories Achieve Quality. *Journal for healthcare quality : official publication of the National Association for Healthcare Quality*, 28(5), pp.44-9. Available at: <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3091732&tool=pmcentrez&rendertype=abstract>.

- INCOSE, 2011. INCOSE. Available at: www.incose.org.
- ISA, 2010. European Interoperability Framework for European Public Services (EIF).
- ISO 19760, 2002. *Systems Engineering – Guide for ISO / IEC 15288 (System Life Cycle Processes)*,
- ISO 9241, 2006. Ergonomics of Human System Interaction. 2006.
- ISO TC184/SC4, 2004. Product Data Representation and Exchange - Part 11: Description Methods: The EXPRESS Language Reference Manual.
- ISO/DIS 10303-233, 2011. Industrial automation systems and integration -- Product data representation and exchange. Available at: www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=55257.
- ISO/IEC 14598-1, 1999. *Software product evaluation-Part 1: General overview*,
- ISO/IEC 14598-5, 1998. Information technology – Software product evaluation – Part 5: Process for evaluators.
- ISO/IEC 14598-6, 2001. *Software engineering – Product evaluation – Part 6: Documentation of evaluation modules*,
- ISO/IEC 9126-1, 2001. *Software Engineering-Software product quality-Part 1: Quality model*,
- ISO/IEC CD 25010, 2011. *Software engineering-Software product Quality Requirements and Evaluation (SQuaRE) Quality model*,
- ISO/IEC CD 25030, 2007. *Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Quality requirements*,
- ISO/IEC CD 8402-1, 1997. *Quality Concepts and Terminology Part One: Generic Terms and Definitions*,
- ISurf, Deliverable 8.1.1 – Functional and Non-Functional Evaluation Criteria for i-Surf Components and Integrated Platform. 2009. Available at: <http://www.srdc.com.tr/isurf/> [Accessed January 20, 2012].
- ISurf, 2010. EU FP7, no. 213031. Available at: <http://www.srdc.com.tr/isurf/> [Accessed January 20, 2012].
- InterOP NOE, 2005. Deliverable DTG3.2: TG MoMo Roadmap, Report on Model Driven Interoperability.
- Jardim-Goncalves, Ricardo, Agostinho, Carlos, et al., 2011. *Standards Framework for Intelligent Manufacturing Systems Supply Chain* S. Renko, ed., InTech.
- Jardim-Goncalves, R., Agostinho, C. & Steiger-Garcia, A., 2011. A Reference Model for Sustainable Interoperability in Networked Enterprises: Towards the Foundation of EI Science Base. *Under revision for IJCM*.
- Knublauch, H. et al., 2004. The Protégé OWL Plugin : An Open Development Environment for Semantic Web Applications. *The Semantic Web – ISWC 2004*, 3298, pp.229-243.

- Kotze, P. & Neaga, I., 2010. Towards an Enterprise Interoperability Framework. In *Advanced Enterprise Architecture and Repositories and Recent Trends in SOA Based Information Systems: In Conjunction with ICEIS 2010*.
- Lee, Y.T., 1999. Information Modeling: From Design to Implementation. In *Proceedings of the Second World Manufacturing Congress*. pp. 315-321.
- Miller, J. & Mukerji, J., 2003. MDA Guide Version 1.0.1. *OMG*, (June).
- Miller, J. & Mukerji, J., 2001. Model Driven Architecture. *Architecture Board ORMSC*, pp.1-31.
- Missikoff, M., Schiappelli, F. & Taglino, F., 2003. A Controlled Language for Semantic Annotation and Interoperability in e-Business Applications. In *2nd Int. Semantic Web Conference (ISWC-03)*. Florida, pp. 1-6.
- Myers, G., 1979. *The Art of Software Testing*, John Wiley & Sons Inc.
- Nallon, J., 2003. An Overview of AP-233 : The Systems Engineering Data Exchange Standard based on STEP (ISO-10303). *Systems Engineering*.
- Noy, N., Ferguson, R. & Musen, M., 2000. The knowledge model of Protégé - 2000: Combining interoperability and flexibility. In *2nd International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000)*.
- Nuseibeh, B. & Easterbrook, S., 2000. Requirements Engineering: A Roadmap. In *ISCE '00 Proceedings of the Conference on the Future of Software Engineering*. New York.
- OMG, 2009. An Ontology Definition Metamodel. Available at:
<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4019068>.
- OMG, 2011a. MBSE Wiki Launched. Available at: www.omg.sysml.org.
- OMG, 2011b. Modeling and Simulation Interoperability (MSI) Challenge Team. Available at:
<http://www.omgwiki.org/MBSE/doku.php?id=mbse.modsim>.
- OMG, 2006. OMG Model Driven Architectures. Available at: www.omg.org/mda/.
- OMG, 2010. OMG Systems Modeling Language (OMG SysML).
- OSJTF, T.D., 2006. AP233 Concepts for Mapping.
- Ogren, I., 2000. On Principles for Model-Based Systems Engineering. *Systems Engineering Journal*, 3(1), pp.38-49. Available at: [http://doi.wiley.com/10.1002/\(SICI\)1520-6858\(2000\)3:1<38::AID-SYS3>3.0.CO;2-B](http://doi.wiley.com/10.1002/(SICI)1520-6858(2000)3:1<38::AID-SYS3>3.0.CO;2-B).
- Ojo, A., Janowski, T. & Estevez, E., 2009. Semantic Interoperability Architecture for Electronic Government. *Proceedings of the 10th Annual International Conference on Digital Government Research*, (Section 2).
- Operations, T. & Crisp II, H., 2007. Systems Engineering Vision 2020. *INCOSE*, Version 2.(September).
- Panetto, H., 2007. Towards a Classification Framework for Interoperability of Enterprise Applications. *International Journal of Computer Integrated Manufacturing*, 20(8), pp.727-740. Available at:
<http://www.tandfonline.com/doi/abs/10.1080/09511920600996419>.

- Peristeras, V. & Tarabanis, K., 2006. The Connection, Communication, Consolidation, Collaboration Interoperability Framework (C 4 IF) For Information Systems Interoperability Defining interoperability. *Ibis*, 1(1), pp.61-72.
- Petzmann, A. et al., 2007. Applying MDA ® Concepts to Business Process Management. *Interchange*, pp.103-116.
- Ray, S. & Jones, A., 2006. Manufacturing Interoperability. *Journal of Intelligent Manufacturing*, 17, pp.681-688.
- Sacevski, I. & Veseli, J., 2007. Introduction to Model Driven Architecture (MDA). *Architecture*, (June).
- Sarraipa, J. et al., 2007. Annotation for Enterprise Information Management Traceability. In *ASME 2007 IDETC & CIE*.
- Sarraipa, J., Jardim-Goncalves, R. & Steiger-Garcia, A., 2010. MENTOR: An Enabler for Interoperable Intelligent Systems. *International Journal of General Systems*, 39, pp.557-573.
- Schmidt, D.C., 2006. Model-Driven Engineering. *IEEE Computer Society*.
- Schrage, M., 1990. *Shared Minds: The New Technology of Collaboration*,
- Selic, B., 2003. The Pragmatics of Model-Driven Development. *IEEE Software*, 20(5), pp.19-25. Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1231146>.
- Selic, Bran, 2003. The Pragmatics of Model-Driven Development. *IEEE Software*, 20(5), pp.19-25. Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1231146>.
- Smolik, P.C., 2000. *MAMBO Metamodeling Environment*.
- Tretmans, J., 2001. An Overview of OSI Conformance Testing. *University of Twente*.
- Truyen, F., 2006. The Fast Guide to Model Driven Architecture.
- UKOLN, U. of B., 2006. *DELOS - A Network of Excellence on Digital Libraries: Deliverable 5.3.1*,
- US Department of Energy, 2011. DOE Standard - Content of System Design Descriptions.
- Ulrich, A. & Grabowski, J., 2004. An Introduction to TTCN-3. Available at: <http://www.ttcn-3.org/Tutorials.htm> [Accessed January 15, 2012].
- Vassiliou-Gioles, T., 2007. The TTCN-3 Language - An Introduction. Available at: <http://www.ttcn-3.org/> [Accessed January 15, 2012].
- White, L., 1987. Software Testing and Verification. *Advances in Computers, Academic Press*, 26.
- Wooldridge, M., 2009. *An Introduction to MultiAgent Systems*, John Wiley and Son.
- Wooldridge, M. & Jennings, N., 1995. Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review*, 10:2, pp.115-152.
- Xu, J. et al., 2009. Model Driven Interoperability through Semantic Annotations using SoaML and ODM. *Information Control Problems in Manufacturing*, 13(0314).

9.1. List of MisMatches

Num	Versioning	Mapping	Choice	Example
1	Naming	Naming	If (nameVer == nameOldMap) equal else naming	
2	Naming	Granularity	Granularity	
3	Naming	Structuring	Structuring	
4	Naming	SubClass-Attribute	SubClass-Attribute	
5	Naming	Schema-Instance	Schema-Instance	
6	Naming	Encoding	Encoding	
7	Naming	Content	Content	

Num	Versioning	Mapping	Choice	Example
8	Naming	Coverage	Coverage	
9	Naming	Precision	Precision	
10	Naming	Abstraction	Abstraction	
11	Granularity	Granularity	If (nameVer == nameOldMap) equal, else Granularity	
12	Granularity	Structuring	Schema Instance	
13	Granularity	SubClass-Attribute	Schema Instance	
14	Granularity	Schema-Instance	Schema Instance	
15	Granularity	Encoding	Granularity + Encoding	

Num	Versioning	Mapping	Choice	Example
16	Granularity	Content	Granularity + Precision	<p>UML diagram for example 16: Person A (Name: String) is connected to Person B via a Content relationship. Person A is also connected to Person A' (LastName: String, FirstName: String) via a Granularity relationship. Person B is connected to Jose Carlos (enumeration) via a Granularity relationship.</p>
17	Granularity	Coverage	Coverage	<p>UML diagram for example 17: Person A (Name: String) is connected to Person B (LastName: String, MiddleName: String, FirstName: String) via a Coverage relationship. Person A is also connected to Person A' (LastName: String, FirstName: String) via a Granularity relationship.</p>
18	Granularity	Precision	Precision	<p>UML diagram for example 18: Person A (Weight: int) is connected to Person B via a Precision relationship. Person A is also connected to Person A' (Weight: Unit: int, Weight: Decimas: int) via a Granularity relationship. Person B is connected to Heavy Medium Ligth (enumeration) via a Granularity relationship.</p>
19	Granularity	Abstraction	Granularity	<p>UML diagram for example 19: Person A (Name: String) is connected to Jose via an Abstraction relationship. Person A is also connected to Person A' (LastName: String, FirstName: String) via a Granularity relationship.</p>
20	Structuring	Structuring	Structuring	<p>UML diagram for example 20: Person A (Name: String) is connected to Person B via a Structuring relationship. Person A is also connected to Person A' via a Structuring relationship. Person A' is connected to Identification and Name. Person B is connected to Identification and Name.</p>
21	Structuring	SubClass-Attribute	Schema Instance	<p>UML diagram for example 21: Person A (Name: String) is connected to Person B via a SubClass-Attribute relationship. Person A is also connected to Person A' via a Structuring relationship. Person A' is connected to Identification and Name. Person B is connected to Jose Carlos (enumeration) and Carlos via a Granularity relationship.</p>
22	Structuring	Schema-Instance	Structuring	<p>UML diagram for example 22: Person A (Name: String) is connected to Person B via a Schema Instance relationship. Person A is also connected to Person A' via a Structuring relationship. Person A' is connected to Identification and Name. Person B is connected to Name.</p>
23	Structuring	Encoding	Structuring + Encoding	<p>UML diagram for example 23: Person A (Weight: kg: int) is connected to Person B (Weight: lbs: int) via an Encoding relationship. Person A is also connected to Person A' via a Structuring relationship. Person A' is connected to Characterist and Weigth.</p>

Num	Versioning	Mapping	Choice	Example
24	Structuring	Content	Structuring	
25	Structuring	Coverage	Structuring	
26	Structuring	Precision	Structuring	
27	Structuring	Abstraction	Structuring	
28	SubClass-Attribute	SubClass-Attribute	SubClass-Attribute	
29	SubClass-Attribute	Schema-Instance	Schema Instance	
30	SubClass-Attribute	Encoding	SubClass-Attribute + Encoding	
31	SubClass-Attribute	Content	Content	
32	SubClass-Attribute	Coverage	SubClass-Attribute	

Num	Versioning	Mapping	Choice	Example
33	SubClass-Attribute	Precision	SubClass-Attribute	
34	SubClass-Attribute	Abstraction	SubClass-Attribute	
35	Schema-Instance	Schema-Instance	Schema-Instance	
36	Schema-Instance	Encoding	Schema-Instance + Encoding	
37	Schema-Instance	Content	Content	
38	Schema-Instance	Coverage	Coverage	
39	Schema-Instance	Precision	Schema-Instance	
40	Schema-Instance	Abstraction	Schema-Instance	
41	Encoding	Encoding	if (measure equal) equal else Encoding	

Num	Versioning	Mapping	Choice	Example
42	Encoding	Content	Content + Encoding	<p>Diagram for row 42: Person A (Weight: kg) and Person B (Height) are connected by 'Content'. Person A is encoded into Person A' (Weight: lbs).</p>
43	Encoding	Coverage	Coverage + Encoding	<p>Diagram for row 43: Person A (Weight: kg) and Person B (Height, Weight) are connected by 'Coverage'. Person A is encoded into Person A' (Weight: lbs).</p>
44	Encoding	Precision	Precision + Precision	<p>Diagram for row 44: Person A (Weight: kg) and Person B (Weight) are connected by 'Precision'. Person A is encoded into Person A' (Weight: lbs). Person B is refined into Weight' (Heavy, Light).</p>
45	Encoding	Abstraction	Abstraction + Encoding	<p>Diagram for row 45: Person A (Weight: kg) and Jose are connected by 'Abstraction'. Person A is encoded into Person A' (Weight: lbs).</p>
46	Content	Content	Content	<p>Diagram for row 46: Person A (Weight) and Person B (Age) are connected by 'Content'. Person A is mapped to Person A' (Height).</p>
47	Content	Coverage	Coverage	<p>Diagram for row 47: Person A (Weight) and Person B (Weight, Height) are connected by 'Coverage'. Person A is mapped to Person A' (Height).</p>
48	Content	Precision	Content	<p>Diagram for row 48: Person A (Weight) and Person B (Weight) are connected by 'Precision'. Person A is mapped to Person A' (Height). Person B is refined into Weight' (Heavy, Light).</p>
49	Content	Abstraction	Content	<p>Diagram for row 49: Person A (Weight) and Jose are connected by 'Abstract'. Person A is mapped to Person A' (Height).</p>
50	Coverage	Coverage	Coverage	<p>Diagram for row 50: Person A (Weight) and Person B (Weight, Age) are connected by 'Coverage'. Person A is mapped to Person A' (Height, Weight).</p>

Num	Versioning	Mapping	Choice	Example
51	Coverage	Precision	Coverage	
52	Coverage	Abstraction	Coverage	
53	Precision	Precision	Precision	
54	Precision	Abstraction	Precision	
55	Abstraction	Abstraction	Abstraction	
56	Equal	Match Mismatch	Match Mismatch (Mapping)	
57	Match Mismatch	Equal	Match Mismatch (Versioning)	
58	Disjoint	Match Mismatch	Match Mismatch (Versioning)	
59	Match Mismatch	Disjoint	Match Mismatch (Mapping)	