



Tiago José Ministro Costa Santos

Licenciado em Ciências da Engenharia Electrotécnica e de
Computadores

Negotiation environment to support enterprise interoperability sustainability

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

Co-orientador: Carlos Eduardo Dias Coutinho, Investigador, FCT-UNL

Júri:

Presidente: Doutor João Francisco Alves Martins
Vogais: Doutor Ricardo Luís Rosa Jardim Gonçalves
Doutor João Pedro Mendonça de Assunção da Silva
Doutor Carlos Eduardo Dias Coutinho



FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Março 2013

Copyright

Negotiation environment to support enterprise interoperability sustainability© Tiago José Ministro Costa Santos

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,
girlfriend and friends.

ACKNOWLEDGMENTS

I would like to demonstrate my acknowledgement to all people that demonstrated their support during my entire course and in particular, the realization of this dissertation.

First of all, I want to thank my family, principally my parents that gave all their efforts and never gave up on me during my entire path in order to obtain the Master's degree. All these years have been accompanied by a great support from them. To my brother that helped me to overcome some obstacle during these years.

To Rita, my girlfriend, that has been there every time that I needed, in good and bad times. Her support during these years, but mainly in this dissertation, was the most important source of motivation and for that I would like to thank her so much.

To my advisor Dr. Ricardo Gonçalves that gave me the opportunity to work with him and for all the support that he gave me towards the successful completion of this work.

To Dr. Carlos Coutinho that helped me with some important points in this dissertation and for the precious help in the scientific validation of the dissertation.

To all my friends and colleagues, which were always present during my entire path, which in one way or another gave their support, where some of them helped me a lot during the course.

Finally, I wish acknowledge the support of the European Commission through the funding of the FP7 ENSEMBLE, UNITE, MSEE and IMAGINE projects.

ABSTRACT

Specialized and diversified global markets are facing a competitiveness that keeps pushing enterprises to abandon their traditional product centrisms, where basically it is enough to concentrate their efforts in very narrow specialization fields and change their methods of work relying on networks of other providers that are able to fulfill their needs towards the development of complete solutions. These new methods of work, regarding the rapid change in markets and business organizations, requires new interoperability demands and complexity levels, from connection and syntax-oriented exchanges to semantic and model-oriented knowledge, which becomes very difficult for enterprises to cope with the pace of change. This dissertation proposes the implementation of a framework, based on agents and rules, to achieve solid and stable integration of solutions, via the use of a strong and formal negotiation mechanism, which will be the basis for increasing the enterprise interoperability in the supply chain for the development of solutions.

KEYWORDS

The keywords for this dissertation are: Interoperability, Sustainable Interoperability, Negotiation, Multi-Agent System and Rules Engine.

RESUMO

Os mercados globais especializados e diversificados enfrentam uma competitividade que obriga as empresas a abandonar os seus tradicionais métodos centrados no produto, onde basicamente é suficiente concentrar os esforços em áreas de especialização muito precisas, para métodos de trabalho que dependem de redes compostas por outros provedores que são capazes de satisfazer as suas necessidades para o desenvolvimento de soluções completas. Estes novos métodos de trabalho, em relação à rápida mudança nos mercados e nas organizações empresariais, requerem novas exigências de interoperabilidade e de níveis de complexidade que vão desde mudanças nos serviços orientados à ligação e à sintaxe até aos serviços de conhecimento orientados a modelos, o que para as empresas se torna muito difícil de acompanhar devido ao ritmo das mudanças. Esta dissertação propõe a implementação de uma estrutura baseada em agentes e regras com o intuito de atingir a sólida e estável integração de soluções, através da utilização de um forte e formal mecanismo de negociação, que será a base para o aumento da interoperabilidade entre empresas no desenvolvimento de novas soluções.

PALAVRAS-CHAVE

As palavras-chave para esta dissertação são: Interoperabilidade, Sustentabilidade da Interoperabilidade, Negociação, Sistema de Multi Agentes e Motor de Regras.

TABLE OF ACRONYMS

ATHENA	Advanced Technologies for interoperability of Heterogeneous Enterprise Networks
ACC	Agent Communication Channel
ACL	Agent Communication Language
API	Application Program Interface
BC	Backward-Chaining
C ⁴ IF	Connection, Communication, Consolidation, Collaboration Interoperability Framework
CAS	Complex Adaptive Systems
CIM	Computation Independent Model
CORBA	Common Object Request Broker Architecture
CS	Coordination Services
CWM	Common Warehouse Meta Model
DF	Directory Facilitator
DSMLs	Domain-Specific Modeling Languages
EIF	European Interoperability Framework
EPS	European Public Services
ESA-CDF	European Space Agency's Concurrent Design Facility
ESB	Enterprise Service Bus
ETSI	European Telecommunication Standards Institute
EU	Europe Union
FC	Forward-Chaining
FIPA	Foundation for Intelligent Physical Agents
GUI	Graphical User Interface
IaaS	Infrastructure as a service
ICL	Interagent Communication Language
ICT	Information and Communication Technology
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IS	Information Systems
J2EE	Java 2 Platform, Enterprise Edition
J2ME	Java 2 Platform, Micro Edition

J2SE	Java 2 Platform, Standard Edition
JADE	Java Agent Development Framework
LISI	Levels of Information System Interoperability
MAS	Multi-Agent System
MAS	Agent Management System
MDA	Model-Driven Architecture
MDD	Model-Driven Development
MDE	Model-Driven Engineering
MDI	Model-Driven Interoperability
MOF	Meta Object Facility
NEGOSEIO	NEGOTiations for achieving and maintaining a Sustainable Enterprise Interoperability
OAA	Open Agent Architecture
OMG	Object Management Group
PIM	Platform Independent Model
PSM	Platform Specific Model
QoS	Quality of Service
SaaS	Software as a Service
SEI	Sustainable Enterprise Interoperability
SMEs	Subject Matter Experts
SOA	Service Oriented Architecture
SQuaRE	Software product Quality Requirements and Evaluation
SUT	System Under Test
SWOT	Strengths, Weaknesses, Opportunities, and Threats
TTCN-2	Tree and Tabular Combined Notation
TTCN-3	Test and Test Control Notation
UML	Unified Modeling Language
VO	Virtual Organization
XMI	XML Metadata Interchange
XML	eXtensible Markup Language

TABLE OF CONTENTS

Acknowledgments.....	VII
Abstract	IX
Resumo	XI
Table of Acronyms	XIII
Table of Contents.....	XV
Table of Figures.....	XIX
List of Tables	XXI
1. Introduction	1
1.1. Research Framework and Motivation	2
1.2. Research Method	2
1.3. Research Problem and Questions.....	4
1.4. Hypothesis.....	4
1.5. Dissertation Outline	5
2. Enterprise Systems Interoperability	7
2.1. The Interoperability Problem	7
2.2. Networked Organizations.....	8
2.3. Interoperability Concept	9
2.4. Interoperability Typologies	10
2.4.1. ATHENA Interoperability Framework.....	10
2.4.2. Connection, Communication, Consolidation, Collaboration Interoperability Framework (C ⁴ IF).....	11
2.4.3. European Interoperability Framework.....	12
2.4.4. Interoperability Classification Framework.....	14
2.4.5. Interoperability Practices Pyramid	15
2.4.6. Levels of Information System Interoperability (LISI)	16
2.4.7. Outlook in the Interoperability Typologies	17
2.5. Model-Driven Interoperability	18
2.5.1. Model-Driven Engineering.....	18
2.5.2. Model-Driven Architecture	19

3. Enterprise Interoperability Sustainability	21
3.1. Sustainable Interoperability	21
3.1.1. Harmonization Breaking	21
3.1.2. Collaboration Networks and Complex Adaptive Systems	22
3.1.3. Heuristic Framework for Network Stability Maintenance	22
3.2. NEGOSEIO – A collaborative framework for SEI	24
3.2.1. The NEGOSEIO methodology	24
3.2.2. The NEGOSEIO architecture	26
3.2.3. Negotiation	27
4. Multi-Agent System and Rules Engine to support Negotiation on SEI	29
4.1. MAS overview	29
4.1.1. MAS as a technology for a negotiation SEI environment	30
4.1.2. Choice of a MAS technology	30
4.2. Rules Engine overview	34
4.2.1. Rules Engine as a technology for a negotiation SEI environment	36
4.2.2. Choice of a Rules Engine technology	36
4.3. Negotiation on SEI Framework and Architecture	38
4.4. System Controller	42
4.4.1. Application Overview	42
4.4.2. Application Specifications	43
4.4.3. Application Workflow	45
4.4.4. Learning Methodology	47
4.5. Trigger Agent	48
4.5.1. Application Overview	48
4.5.2. Application Specifications	49
4.5.3. Application Workflow	50
4.6. Usability Cases	52
5. Proof-of-concept Implementation	55
5.1. Application Scenarios	55
5.1.1. First Scenario – Block negotiation method	56
5.1.2. Second Scenario – Split negotiation method	57

5.2.	Implementation Steps	58
5.2.1.	Step 0 – Environment setup	59
5.2.2.	Step 1 – Negotiation.....	59
5.2.3.	Step 3 - Knowledge	62
6.	Testing and Hypothesis Validation.....	65
6.1.	Testing Methodologies.....	65
6.1.1.	iSurf Functional and Non-Functional Evaluation Methodology.....	66
6.1.2.	Tree and Tabular Combined Notation – Test Notation Standard	68
6.1.3.	Adopted Test Methodology.....	70
6.2.	Requirements and Functionalities	70
6.3.	Testing	72
6.3.1.	Step 0 – Environment setup	72
6.3.2.	Step 1 – Negotiation.....	76
6.3.3.	Step 3 – Knowledge	80
6.3.4.	Performance comparison	82
6.4.	Hypothesis Validation	87
6.5.	Scientific Validation.....	87
7.	Final Considerations and Future Work.....	89
7.1.	Future Work.....	89
8.	References	91

TABLE OF FIGURES

Figure 1-1 - Classical Research Method phases (based on (Camarinha-Matos 2010)).....	3
Figure 2-1 - Interoperability on all layers of an enterprise (Chen & Doumeingts 2003).....	8
Figure 2-2 - ATHENA MDI Framework (ATHENA 2010)	11
Figure 2-3 - The C ⁴ IF (Peristeras & Tarabanis 2006)	12
Figure 2-4 – The EIF (ISA 2010)	13
Figure 2-5 - Interoperability Classification Framework (Panetto 2007).....	15
Figure 2-6 - Interoperability Practices Pyramid (Jardim-Gonçalves et al. 2010)	16
Figure 2-7 - LISI interoperability maturity model (C4ISR 1998).....	17
Figure 2-8 - The levels of MDA approach (Petzmann et al. 2007)	19
Figure 3-1 - Sustainable Interoperability Framework (Agostinho & Jardim-Gonçalves 2009)	23
Figure 3-2 - Increasing the SEI through negotiations (Cretan et al. 2012)	24
Figure 3-3 - NEGOSEIO methodology (Cretan et al. 2012).....	25
Figure 3-4 - NEGOSEIO framework architecture (Cretan et al. 2012)	26
Figure 4-1- The JADE architecture (Bellifemine et al. 2003).....	33
Figure 4-2 - Traditional Rules Engine architecture (D. I. Liu et al. 2010).....	36
Figure 4-3 - Global vision of the proposed Prototype environment	39
Figure 4-4 - System Controller architecture	40
Figure 4-5 - Trigger Agent architecture	41
Figure 4-6 - System Controller use case diagram	43
Figure 4-7 - System Controller class diagram	44
Figure 4-8 - System Controller sequence diagram	46
Figure 4-9 - Questionnaire made by negotiation initiator	47
Figure 4-10 - Example of a rule	48
Figure 4-11 - Trigger Agent use case diagram.....	49
Figure 4-12 - Trigger Agent class diagram.....	50
Figure 4-13 - Trigger Agent sequence diagram.....	52
Figure 4-14 - Block negotiation method scenario	53
Figure 4-15 - Split negotiation method scenario.....	53
Figure 5-1 - Block negotiation method scenario detailed.....	56
Figure 5-2 - Split negotiation method scenario detailed.....	58
Figure 5-3 - Environment agents setup.....	59
Figure 5-4 - Block negotiation method sequence	61
Figure 5-5 - Split negotiation method sequence	61
Figure 5-6 - Negotiation rule example.....	62
Figure 5-7 - System Controller knowledge management process.....	63
Figure 5-8 - Knowledge results rules example	63
Figure 5-9 - Knowledge decision rules example.....	64
Figure 5-10 - Knowledge suggestion rules example.....	64

Figure 6-1 - GUI of System Controller when the application starts	73
Figure 6-2 - System Controller initiated successfully.....	73
Figure 6-3 - Trigger Agent connected to System Controller.....	74
Figure 6-4 - System Controller with two Trigger Agents connected	74
Figure 6-5 - System Controller receives the change notification from TriggerAgent-C	76
Figure 6-6 - TriggerAgent-A receives the proposal made by the TriggerAgent-C.....	77
Figure 6-7 - Proposal questionnaire created by TriggerAgent-C.....	77
Figure 6-8 - System Controller receives a reject decision from TriggerAgent-A for Proposal 1..	78
Figure 6-9 - System Controller after receiving all proposal decisions.....	79
Figure 6-10 - Proposal creation from TriggerAgent-N.....	80
Figure 6-11 - TriggerAgent-X response to the proposal made by TriggerAgent-N.....	81
Figure 6-12 - Proposal received by TriggerAgent-X with a suggestion by the System Controller	81
Figure 6-13 - Time spent in the system change vs. the complexity of the changes without negotiation.....	84
Figure 6-14 - Complexity of the system change over time with negotiation.....	85
Figure 6-15 - Interoperability complexity over time with and without negotiation.....	85
Figure 6-16 - Systems re-establishment time vs. interoperability complexity, with and without negotiation.....	86

LIST OF TABLES

Table 2-1 - Compatibility levels (adapted from (IEC TC65 2002)).....	9
Table 4-1 - Agent technologies comparison.....	32
Table 4-2 - Rules Engine technologies comparison	38
Table 6-1 - Simplified example of a TTCN table test	69
Table 6-2 - System Controller initialization functional test	75
Table 6-3 - Second Trigger Agent connection functional test	75
Table 6-4 - System Controller disconnects an Trigger Agent functional test	76
Table 6-5 - Negotiation flow functional test	79
Table 6-6 - System Controller knowledge process functional test.....	82

1. INTRODUCTION

As the current world's economy navigates through serious difficulties in, basically, all markets, the involved enterprises are struggling and fighting to remain healthy and competitive. This fight in some cases is aimed to survive among other enterprises, since that in all markets there are more and more enterprises "closing their doors". To counteract these economy instabilities, enterprises must do some continuously adaptations in their current methods of work, such as, search, face and act. These constant adaptations will allow a better response to new business and collaborative opportunities. In order to be capable of such responsiveness and because most of enterprises might not be able to provide some requested competencies, they will have to collaborate with their peers, and to make this happen, enterprises should be capable of forming Virtual Organizations (VO) to reach some agile and survival mechanisms to confront the current markets turbulence (Oliveira & Camarinha-Matos 2012).

The creation of new VOs is not the only concern that enterprises should face, they must also be aware that in order to collaborate with other enterprises, their systems and applications need to be interoperable, in other words, they should be capable of changing all types of defined information without any constraint, within and across enterprises. The interoperability between the involved enterprises also means that their systems and applications must be adaptable to different network environments (Ray & Jones 2006), (Jardim-Goncalves et al. 2007). Since that the environments are constantly changing and evolving, enterprises need to find a solution to maintain the interoperability with their partners, suppliers and customers when these changes occur.

All the concerns about the reliability in the data exchanges on the emergence of a Future Internet that are being addressed by the advances in the Information and Communication Technology (ICT) go beyond the current concerns of being able to interconnect and establish a data flow without errors in data exchanges. These new concerns about interoperability are related to deeper knowledge, semantics, models and business flows (Cretan et al. 2012) leading to the concept of Sustainable Enterprise Interoperability (SEI), where enterprises can create sustainable environments with cooperation networks in order to maintain their interoperability even when the environments are constantly changing and evolving (Jardim-Gonçalves et al. 2010).

Similarly to the behavior of personal relationships, the SEI only can be achieved if interoperability is not static which also is the key to make the SEI a valuable option to grant the enterprises interoperability (Coutinho et al. 2012), since that the sustainable interoperability add some extra time on total time spent on the communications between the enterprises comparing to the semantic interoperability (Jardim-Gonçalves et al. 2010). The environment evolution often leads to system changes that will break the interoperability between the already established parties. After the interoperability breaks, the parties need some time to adjust their systems in

order to re-establish the interoperability. The re-establish time, called “downtime” grows with the interoperability complexity, leading to large out-of-business time where all parties loose (Coutinho et al. 2012).

In (Coutinho et al. 2012) is proposed a Collaborative Negotiation Framework contributing to the improvement of the Enterprise Interoperability which offers new mechanisms to support negotiation towards interoperability in distributed environments. These negotiation mechanisms will allow enterprises negotiating their interoperability between each other with the proposed interoperable players of business-to-business interactions.

The motivation of this thesis is centralized on the Collaborative Negotiation Framework added to the SEI, which using the negotiation factor on the SEI environment as a possible solution for the problem on the time spent in the SEI when is necessary to adjust some system in order to re-establish the interoperability in the environment.

1.1. Research Framework and Motivation

As time passes, the meaning of interoperability also evolves, since that it is no longer associated only to the messages exchanging between two or more systems. In the present days the term interoperability must rely on knowledge and share of the involved business models and semantics which allows richer and stronger interoperability between parties making much more harder to break it and quicker to regain it (Coutinho et al. 2012).

So, this dissertation aims to contribute with an implementation of a negotiation SEI environment that will allow to the involved enterprises, negotiate their interoperability strategies in order to fortify the interoperability relations in the environment. The proposed environment will also allow better responses to the interoperability harmonization breaks that occur in the relations between enterprises which will be translated in shorter enterprise downtimes.

1.2. Research Method

This dissertation uses a research method based on the classical research method proposed in (Camarinha-Matos 2010) which is composed by seven phases and is represented in Figure 1-1. Each phase is composed by several tasks and as Figure 1-1 shows, the method starts with the problem finding and ends with the publication results and transfer to industry right after the results interpretation. This method also allows starting over again if the results are unsatisfactory as Figure 1-1 illustrates with the arrows on the left side of the picture.

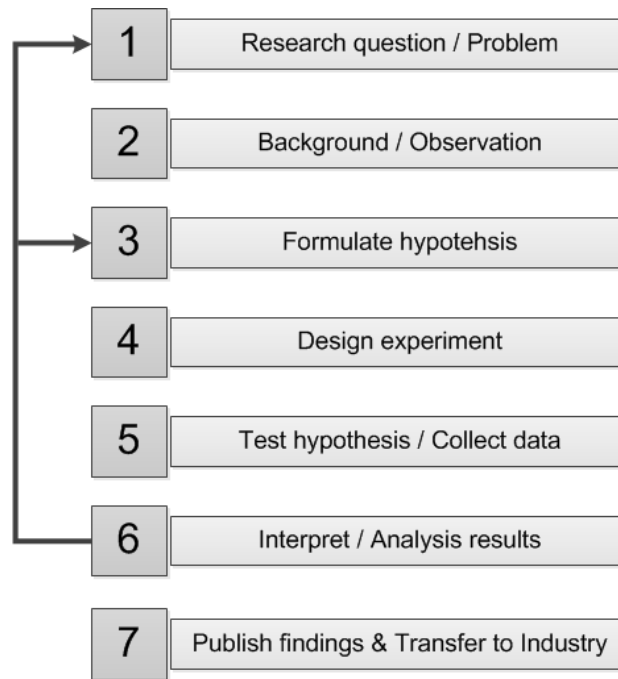


Figure 1-1 - Classical Research Method phases (based on (Camarinha-Matos 2010))

Each phase of the research method will now be explained more detailed:

1. *Research question / Problem*: This first step is the most important in this research method and it will define the area of interest of the research. The questions made in this step must be capable of being confirmed or refused. These questions are presented in the Section 1.3.
2. *Background / Observation*: This step is aimed to study the state of the art by reviewing some literature, previous done projects and informal discussions which will help to discover and distinguish the previous related work from what it will be done in the this research. This task may create new ideas for the research and because of that, this research model allows iteration between this step and the first one. The background task is made in the Sections 2 and 3.
3. *Formulate hypothesis*: In this step the scientific hypothesis should be made, what will bring clarity, specificity and focus to the research. The hypothesis should be simple, specific, conceptually clear and capable of verification. This research will present the hypothesis in the Section 1.4.
4. *Design experiment*: This step includes planning in detail all the experimental phase, often this step includes the design of a prototype or the system architecture. The Sections 4 and 5 will present the design experiments for this research.

5. *Test hypothesis / Collect data*: Here is where the pilot tests are done allowing the first architecture evaluation by testing and simulating different scenarios. These tests will be presented in the Section 6.
6. *Interpret / Analysis results*: This phase will use the data collected in the previous phase to perform an analysis of the results and do some discussion regarding the literature, the research objectives and the research questions. If the results are satisfactory is possible to consider the next steps making some recommendations for further research, but if the results are unsatisfactory, as referred before, here it is possible to return to first step and try a different approach. These tasks are made in the section 6.
7. *Publish findings and Transfer to Industry*: The final step is where it gives to know all the work done, because, as mention by Camarinha-Matos in (Camarinha-Matos 2010), a research result is not a contribution to the field if no one knows about it or can use it. So, when positive results are achieved is important to share these results to the scientific community, like in scientific papers, conferences and Journals. This step is presented in the sub-section 6.5.

1.3. Research Problem and Questions

Following the research method presented in the previous section, some questions will be presented in order to define the course of this thesis.

- In a SEI environment, introducing the interoperability negotiation will help the environment to reach better and stable interoperability states?
 - The system downtime due to harmonization breaks, it will be greater than if the environment does not have negotiation?
 - The environment can benefit from the interoperability negotiation?
- The networked enterprise environment can benefit from the interoperability negotiation?

1.4. Hypothesis

- If the proposed framework is capable of manage the negotiation and knowledge in the networked enterprise environment, then it is possible to make the environment more stable and more efficient to the harmonization breaks towards a robust SEI.

1.5. Dissertation Outline

This section will describe the context of this dissertation, explaining the main goal of all sections. In the sections 2, 3 and 4 are presented the topics that are the background of this thesis, where it is explained the Enterprise Systems Interoperability in the section 2, covering the interoperability subject, making a brief description of some approaches to classify the interoperability layers and also a brief explanation of the Model-Driven Interoperability (MDI) method. In the section 3 it will be explained more detailed the Enterprise Interoperability Sustainability, regarding the important points that this thesis will be focused.

In the section 4 is made a description about the Multi-Agent System and the Rules Engine that will support the proof-of-concept implemented in this thesis. The section will also define the framework and the architecture that will be the base for the proof-of-concept. The proof-of-concept implementation will be presented in the section 5, describing the environment creation, the negotiation and the knowledge steps. Section 6 is where the proof-of-concept is validated and where the tests over the system will be described.

Finally, in the section 0 it will be presented the thesis conclusions and the future work topics.

2. ENTERPRISE SYSTEMS INTEROPERABILITY

In a not very distant past, enterprises and organizations really needed to put their focus on being able to interconnect and to establish a data flow between their partners where the information changed should not contain any errors. Today, the enterprises and organizations main focus is much more complex because they need to be made interoperable in some different ways, such as both in terms of their business processes, their applications or IT systems and even in terms of their human resources, in order to face the current business challenge (F. Vernadat 2003) and (F. Vernadat 2004).

The term interoperability was defined by IEEE (Geraci et al. 1991) as the ability of two or more systems or components to exchange information and to use the information that has been exchanged. These means that, today, the new concerns about interoperability relate to deeper knowledge, semantics, models and business flows (Cretan et al. 2012) which will force both enterprises and organizations to stay connected in a network in order to succeed.

According to the European interoperability framework (Ruggaber 2006), interoperability can be considered in three aspects:

- Technical aspect that represents the data and message exchange;
- Semantic aspect that represents the meaning of the information and service shared;
- Organizational aspect that represents the business units, process and people interactions across organization borders.

2.1. The Interoperability Problem

Nowadays, the great trend in the global market is the continually collaboration between the enterprises and organization during the entire product life cycle where constant changes, both in inter and intra organizational environment, will continues in the future. With the aim to overcome these constant changes, organizations should not only have the necessary flexibility to react to these changes in markets and trading partners but also they have to deal with the internal changes from both technical and organizational point of view (Chen & Doumeingts 2003).

Thinking now in the enterprise applications, another problem stands out since that the software code, once written and implemented, turns very hardly to be modified and in many cases, the software previously developed was not designed to be interoperable with other applications. A great example of this problem is that, today, although many systems and applications speak through XML language, their data models and schemas are often a bit

different. Another obstacle in the software code is the lack of standards, for example, to control the business process flows across multiple systems (Chen & Doumeingts 2003).

These problems tend to getting worse in extended enterprises and networked organizations where the collaboration is the key to achieve further benefits. It is very important to understand the socioeconomic influences that surround the enterprises in order to deal with these interoperability problems. Also, with the same importance, it is necessary to understand the general set-up of organizations in the present and also in the future business networks realizing their influence on interoperability issues (Ruggaber 2006).

Since that interoperability is not only a concern in the software and IT technologies, it is necessary to change the focus to the communication and transactions between different organizations which must be based on shared business references. These references that are shared between the organizations must be based on business standards and norms in order to facilitate the interaction among organizations (Chen & Doumeingts 2003).

In the Figure 2-1 is illustrated an example of a conceptual model of the interaction between two enterprises where interoperability must be achieved on all layers of an enterprise, in order to reach a meaningful interoperation between enterprises. This concept includes some extra points in each layer and is it is also possible use some semantic descriptions to achieve the necessary mutual understanding between enterprises that want to collaborate. In the Business layer are included the business environment and the business processes. The Knowledge layer contains the organizational roles, skills and competencies of employees and the knowledge assets. In the last layer, the ICT layer hosts the applications, data and communication components (Chen & Doumeingts 2003).

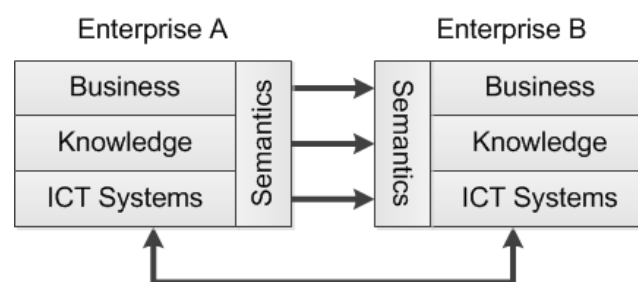


Figure 2-1 - Interoperability on all layers of an enterprise (Chen & Doumeingts 2003)

2.2. Networked Organizations

As stated before, enterprises and organizations have better changes to survive in the current economy if they are connected to a network of organizations. So, networked organizations are characterized by having a distributed control, inter-organizational business process crossing the enterprise boundaries, various producer-consumer supply chains and shared information and knowledge. These networked organizations have some important challenges such as the operation optimization via co-decision, co-ordination and even

negotiation mechanisms. The main advantages of the networked organizations are the flexibility and the dynamics that their structures offer, which allows a better control of the network, since that it is possible to add or remove new nodes to the network in order to face the economic turbulence and provides the a better agility to implement new business strategies (F. B. Vernadat 2007).

2.3. Interoperability Concept

Vernadat (F. B. Vernadat 1996) defines interoperability as the ability to communicate with their pier systems and access the functionality of the pier systems, but from the software engineering point of view, the term interoperability is defined by two or more software systems that are capable of co-operate and easily work together in a simple way, i.e. without a particular interfacing effort. The concept of interoperability was defined by (IEC TC65 2002) as a certain degree of compatibility, which can be seen in the Table 2-1, as “The application data, their semantic and application related functionality of each device is so defined that, should any device be replaced with a similar one of different manufacturer, all distributed applications involving the replaced device will continue to operate as before the replacement, but with possible different dynamic responses”.

Table 2-1 - Compatibility levels (adapted from (IEC TC65 2002))

		Compatibility levels					
		Incompatible	Coexistent	Interconnectable	Interworkable	Interoperable	Interchangeable
System Feature	Dynamic Behavior						x
	Application Functionality					x	x
	Parameter Semantics					x	x
	Data Types				x	x	x
	Data Access			x	x	x	x
	Communication Interface			x	x	x	x
	Communication Protocol		x	x	x	x	x

Merging the definition of interoperability made in (IEC TC65 2002) and the definition of enterprise application made in (Chen & Doumeingts 2003) it is possible to presume that the interoperability is achieved only when the interaction between two systems can, at least, be granted in three levels, namely: *data*, *resource* and *business process* where the semantics are defined in a business context (Chen & Doumeingts 2003).

2.4. Interoperability Typologies

As time elapses, more and more authors present new solutions to help achieve interoperability. Sometimes the interoperability types are called levels because normally the interoperability types follow a scale of advancement, where the higher a type is placed in the scale, the more advanced the achieved interoperability is considered. In order to reach an upper level of interoperability advancement, all the previous levels have to be successfully addressed, which means that certain features of an upper interoperability type may become available without fully addressing all the lower interoperability levels (Peristeras & Tarabanis 2006).

In the next sub-sections are presented and explained some well-known interoperability typologies.

2.4.1. ATHENA Interoperability Framework

In order to achieve real and meaningful interoperation between enterprises, ATHENA Interoperability Framework was created with a holistic perspective on interoperability. This framework is represented in the Figure 2-2 and was built on the vision “Enterprises are able to flexibly develop and execute interoperable applications based on model-driven development approaches to service-oriented and adaptive software solutions” and integrates principles of model-driven development, service-oriented architectures and adaptive architectures. The ATHENA framework is structured in three main integration areas (ATHENA 2010), that are described below:

- *Conceptual Integration* that is 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.

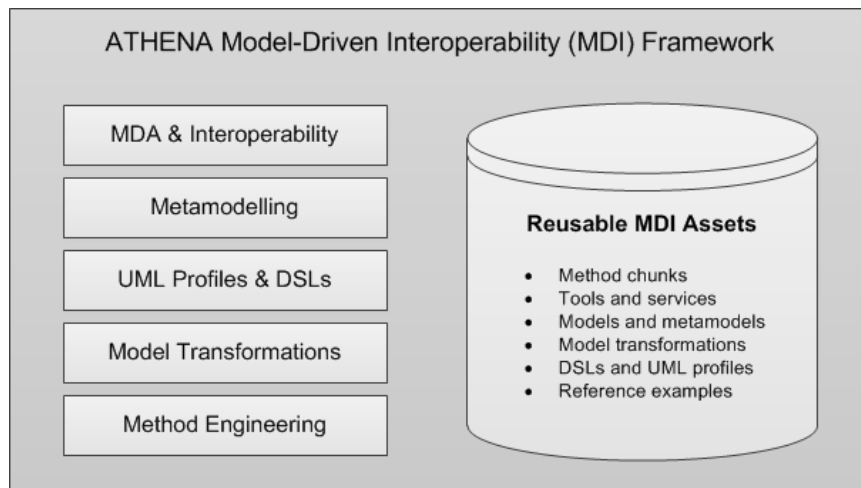


Figure 2-2 - ATHENA MDI Framework (ATHENA 2010)

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

The C⁴ Interoperability Framework was presented in (Peristeras & Tarabanis 2006) and has been developed focusing on the ways that the Information Systems (IS) communicate, modeling this communication as a discourse. Basically, this framework tries to transfer basic linguistic concepts to the domain of IF communication, in order to build their interoperability typology.

The C⁴IF defines four interoperability types that are explained below and are represented in the Figure 2-3.

- *Connection* refers to the ability of IS to exchange signals. To succeed in this, a physical contact/connection should be established between two (or more) systems;
- *Communication* refers to the ability of IS to exchange data. To succeed in this, a predefined data format and/or schema need to be accepted by the interlocutors. The focus of this type is on the data content and can be considered at least two levels of communications. The first level, the exchange is based on a commonly accepted data and in the second level, the exchange includes data;
- *Consolidation* refers to the ability of IS 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.

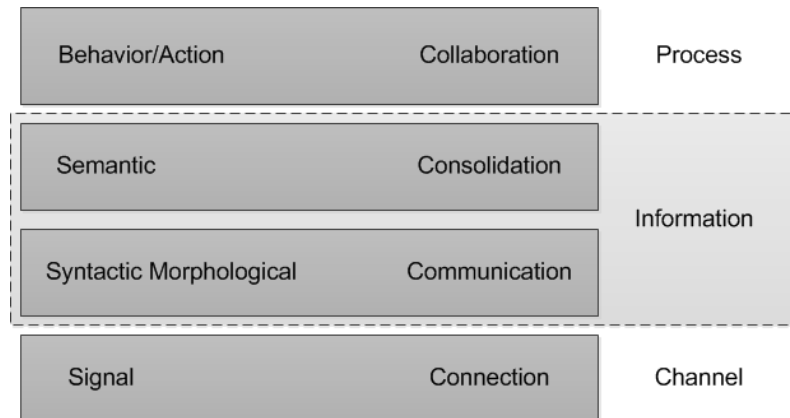


Figure 2-3 - The C⁴IF (Peristeras & Tarabanis 2006)

The four interoperability types that were presented above are organized in three demarcated areas, as illustrated on the Figure 2-3 and these areas are explained below:

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

2.4.3. European Interoperability Framework

The European Interoperability Framework (EIF) presented in (ISA 2010) addresses interoperability in the very specific context of providing European Public Services (EPS), although the provision of EPS almost always involves exchanging data between ICT systems. For EIF, EPS means “a cross-border public sector service supplied by public administrations, either to one another or to European businesses and citizens”.

The EIF presents four levels of interoperability where each one deserves special attention when a new EPS is established. These four interoperability levels are illustrated in the Figure 2-4 and are explained below.

- *Political Context* represents the establishment of a new EPS in the result of direct or indirect action at political level. In order to be effective, efforts should be done to facilitate cooperation among public administrations where all stakeholders involved must share visions, agree on objectives and align priorities;

- *Legal Interoperability* is where the legal validity of exchanged information between Member States to provide EPS must be maintained across borders and data protection legislation in both originating and receiving countries must be respected;
- *Organisational Interoperability* is concerned how organizations, such as public administrations in different Member States, cooperate to achieve their mutually agreed goals;
- *Semantic Interoperability* enables organizations to process information from external sources in a meaningful manner and ensures that the precise meaning of exchanges information is understood and preserved throughout exchanges between parties. In order to reach semantic interoperability at European level, it is necessary at least agreed processes and methodologies for developing semantic interoperability assets and agreement by sector-specific and cross-sectoral communities on the use of semantic interoperability assets at EU level;
- *Technical Interoperability* covers the technical aspects of linking information systems which includes aspects such as interface specifications, interconnections services, data integration services, data presentation and exchange, etc. Technical interoperability should be ensured, whenever possible, via the use of formalized specifications.

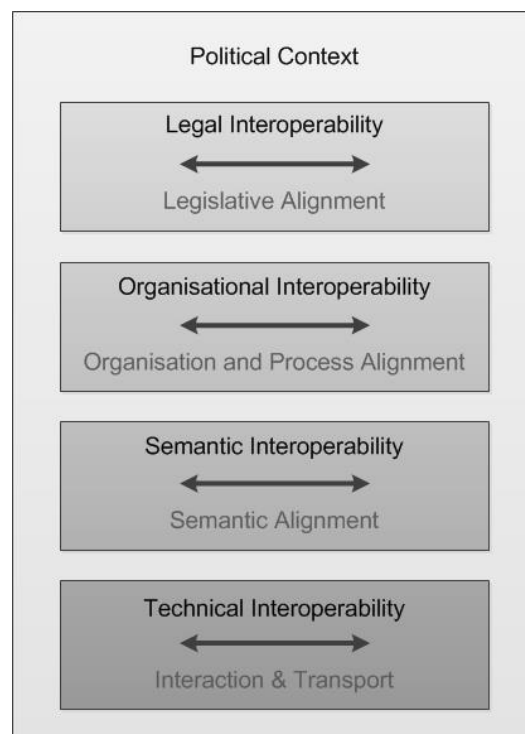


Figure 2-4 – The EIF (ISA 2010)

2.4.4. Interoperability Classification Framework

The Interoperability Classification Framework was proposed in (Panetto 2007) and is composed by six kinds of interoperability solutions that are illustrated in Figure 2-5 and are explained below.

- *Synchronic interoperability* are issues where applications exchange models defined by compatible languages (the same syntax) but with different semantics, in a synchronous way;
- *Model-driven interoperability* is focus mainly, but not only, on technologies (or standards) to solve model syntactic transformations;
- *Semantic-driven interoperability* is focus only on developments where the semantic alignment is the main issue;
- *Vertical interoperability* is 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 interoperability* is when applications interoperability problems may occur when exchanging models at the same abstraction level (CIM, PIM or PSM). As in the vertical interoperability, the exchange process from one application to another, also involves models transformations (syntactic) and semantic alignment (also called concept mapping);
- *Diachronic interoperability* 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.

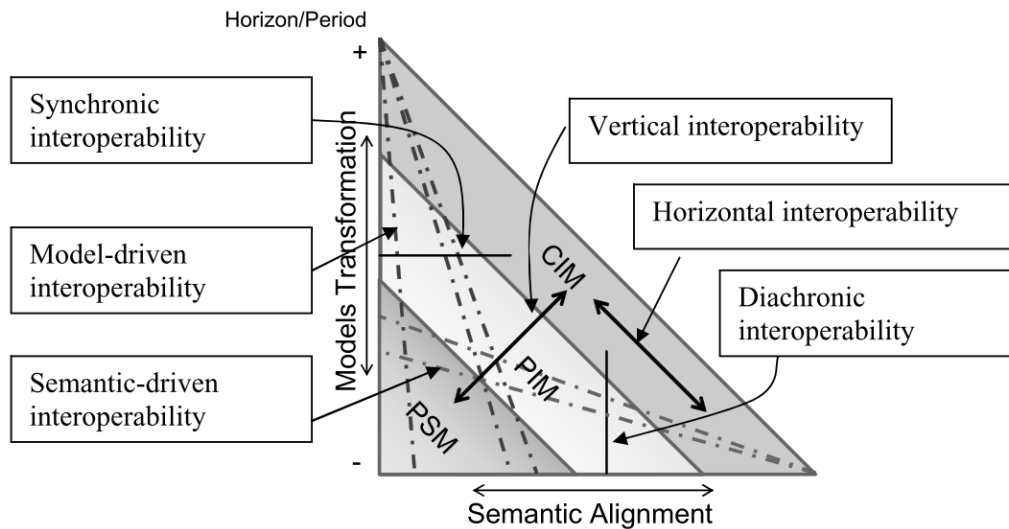


Figure 2-5 - Interoperability Classification Framework (Panetto 2007)

2.4.5. Interoperability Practices Pyramid

The Interoperability Practices Pyramid was presented in (Jardim-Gonçalves et al. 2010) and includes five layers of interoperability types. These layers are represented in Figure 2-6 and explained below.

- *Slack interoperability* is when there is no previous understanding between the sender and the receiver on all communication sets. This layer uses a rudimentary communication methodologies where the time spent on the communication is increased with the time spent on clarifications, responses and human interventions;
- *Unregulated interoperability* is when organizations are focused on peer-to-peer relationships and each organization uses its own data format and business rules. Also each organization handles as many mappings as the number of business partners;
- *Standard-based interoperability* is when the exchanged information is based on common models using standards as the reference format for that information exchange;
- *Semantic interoperability* is defined by two kinds of knowledge: tacit knowledge, that people carry in their minds, providing context for people, places, ideas and experiences; and explicit knowledge that has been or can be articulated, codified, and stored in certain media;
- *Sustainable interoperability* is composed by some capabilities, such as discovery, learning, adaptability, transient analysis and notifications. All these capabilities work together aiming of improving the quality of service by contributing to a more

robust interoperability, avoiding excessive consumption of resources when the dynamicity of systems and networks causes harmonization breaking.

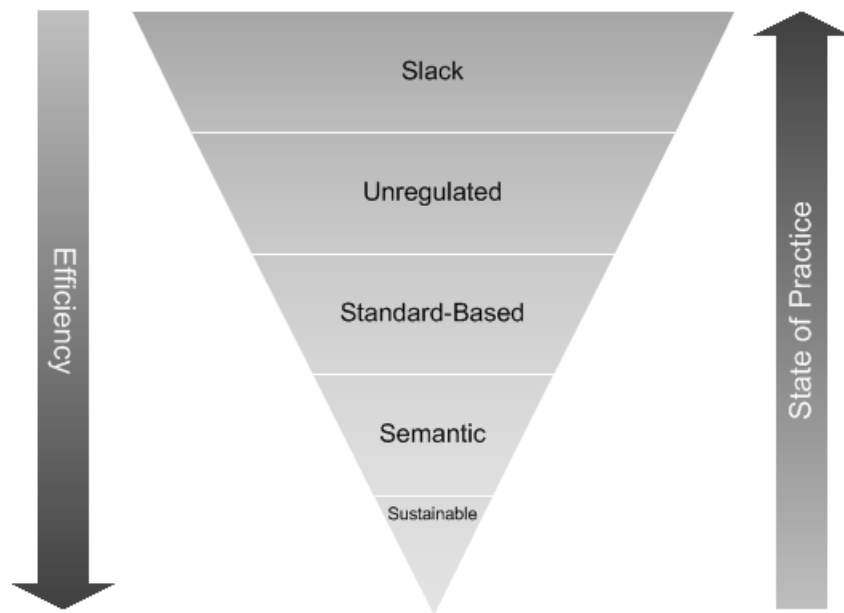


Figure 2-6 - Interoperability Practices Pyramid (Jardim-Gonçalves et al. 2010)

2.4.6. Levels of Information System Interoperability (LISI)

Levels of Information System Interoperability (LISI) was created and presented on (C4ISR 1998) and considers five increasing levels of sophistication with respect to exchanging and sharing information and services through the system's life cycle. These levels are illustrated in Figure 2-7 and are described below.

- *Level 0 – Isolated Interoperability in a Manual Environment* – This level embraces a wide range of isolated or stand-alone systems and is not allowed, nor are available, direct electronic connections. So the only interface between these systems is by manual re-keying or via extractable, common media;
- *Level 1 – Connected Interoperability in a Peer-to-Peer Environment* – In this level, systems are capable of being linked electronically and providing some form of simple electronic exchange. Generally these systems exchange homogeneous data types, such as voice, simple “text” e-mail, or fixed graphic files because have a limited capacity;
- *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 increase on the complexity of the media exchanges with the use of formal data models (logical and physical);

- *Level 3 – Domain-Based Interoperability in an Integrated Environment* – These systems are capable of being connected via wide area networks which allow multiple users to access data. It is present a domain-based data model that is understood, accepted and implemented across a functional area or group of organizations that comprises a domain.
- *Level 4 – Enterprise-Based Interoperability in a Universal Environment* – In this level, systems are capable of operating using distributed global information space across multiple domains, which allows the simultaneous access and interaction of multiple users to complex data. All data and applications are fully shared and can be distributed throughout this space to support information fusion.

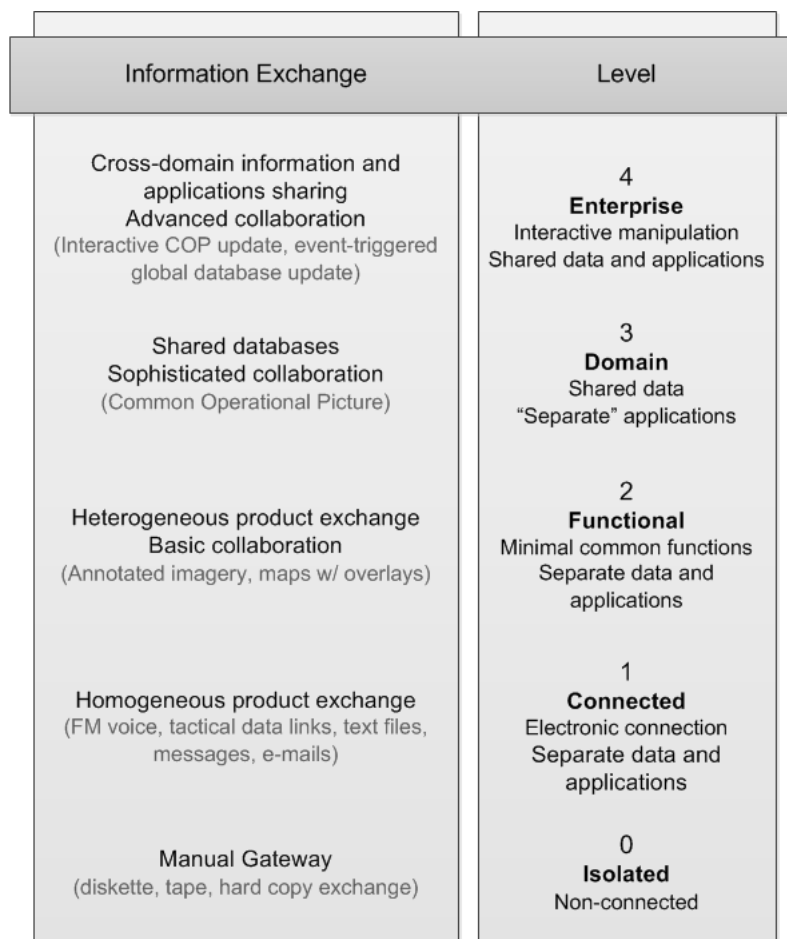


Figure 2-7 - LSI interoperability maturity model (C4ISR 1998)

2.4.7. Outlook in the Interoperability Typologies

With some important interoperability typologies already presented, where all of them have the purpose to evaluate the interoperability status inside an organization, a specific systems or a network, this thesis will focus on the Interoperability Practices Pyramid that (Jardim-Gonçalves et al. 2010) presented in 2.4.5 section, more precisely in the Sustainable Interoperability layer of

this pyramid, because it represents a clear advancement on the state of art on the interoperability among enterprises and organizations.

2.5. Model-Driven Interoperability

As presented before, enterprises and organizations face many challenges related to the lack of interoperability which they need to grant between their applications and software in order to achieve seamless business across organizational boundaries. Today Model-Driven Development (MDD) and in particular OMG's Model-Driven Architecture (MDA¹) (Miller & Mukerji 2003) is emerging, providing tools to develop modern enterprise applications and software systems. Compared to the earlier non-modeling approaches, MDD paradigm provides a better way of addressing and solving interoperability issues, however this is not an easy task. In order to facilitate this task, several projects were created providing guidance on how MDD should be applied to address interoperability. Some of those created project were presented in the previous section (Elvesæter et al. 2006).

2.5.1. Model-Driven Engineering

Model-Driven Engineering (MDE) also called MDD is a software-engineering approach consisting of the application of models and model technologies to raise the level of abstraction at which developers create and evolve software. This approach tries to both simplify (making easier) and formalize (standardizing) the various activities and tasks that comprise the software lifecycle (Hailpern & Tarr 2006), (ATHENA 2010).

The developing of MDE technologies that combine the following aspects is a promising approach to address platform complexity (Schmidt 2006).

- *Domain-specific modeling languages* whose types systems formalize the applications structure, behavior and requirements within particular domains, such as, for example, warehouse management and middleware platforms;
- *Transformation engines and generators* that analyze certain aspects of models and then synthesize various types of artifacts, such as, for example, simulations inputs, XML deployment descriptions and alternative model representation.

Using the MDE technologies it is possible to tailor the Domain-Specific Modeling Languages (DSMLs) to precisely match the domains semantic and syntax, instead of general-purpose notations that rarely express application domain concepts and design intent. Another advantage of the MDE is the utilization of graphic elements that relate directly to a familiar domain which allows the system engineers and software architects ensure that software systems meet user needs. Moreover, MDE tools impose domain-specific constraints and

¹ Model-Driven Architecture and MDA are registered trademarks of the Object Management Group (OMG)

perform model checking that can detect and prevent many errors early in the life cycle and it is often much easier to develop, debug and evolve the applications created with MDE tools since that today's platforms have much richer functionality and QoS than those in the past years.

2.5.2. Model-Driven Architecture

MDA was introduced in 2001 by the OMG as an approach for the specification of software systems based on a model transformation concept. One of the principal goals of the MDA approach is to separate software design from architecture and realization technologies facilitating that design and architecture can alter independently increasing the possibilities of automation in software development. The other important goals of MDA are portability, interoperability and reusability. Basically, to reach these goals, MDA is focused only on standardized techniques, like the Unified Modeling Language (UML), the Meta Object Facility (MOF), the XML Metadata Interchange (XMI) and the Common Warehouse Meta Model (CWM) (Petzmann et al. 2007), (ATHENA 2010), (Truyen 2006).

Since technology is constantly evolving and new platforms and technologies are constantly emerging, MDA allows a rapid development of new specifications that leverage them, and streamlines the process of their integration. This makes MDA a comprehensive and structured solution for application interoperability and portability into the future (Truyen 2006).

In order to solve this situation, MDA contains in its core an approach to design IT system architectures taking into consideration heterogeneous systems to be discovered in different level of models. This approach tries to describe how to perform a transformation on these models, step by step, from an independent system level to platform models. MDA defines three different types of models that can perhaps more accurately be described as layers of abstraction, since within each of these three layers, a set of models can be created, each one corresponding to a more focused viewpoint of the system (user interface, information, engineering, architecture, etc.). These models or layers of abstractions are explained below and are illustrated in the Figure 2-8 (Petzmann et al. 2007), (Truyen 2006).

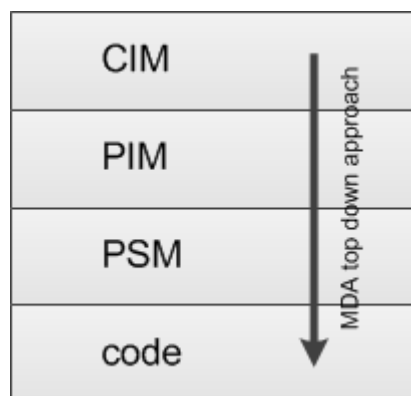


Figure 2-8 - The levels of MDA approach (Petzmann et al. 2007)

- *Computation Independent Model (CIM)* is where the environment and situation in which the system will be used from a business point of view are described. A CIM is also referred as a business or domain model because it uses a familiar vocabulary to the Subject Matter Experts (SMEs). Here is presented exactly what the system is expected to do, hiding all related technology information in order to remain independent of how that system will be implemented;
- *Platform Independent Model (PIM)* is where the view of a system from the platform independent viewpoint is designed. The main goal of this layer is producing models, which can be transformed in an arbitrary system platform. A PIM is independent enough to enable its mapping to one or more platforms through the definition of a set of services in a way that abstracts out technical details;
- *Platform Specific Model (PSM)* is a view of specific platform. PSM merges the specification of the PIM with the specific details of a particular system;
- *Code* is the code produced (in the broader sense) that can be run on specific platforms since MDA is a software engineering approach.

3. ENTERPRISE INTEROPERABILITY SUSTAINABILITY

In the previous section some details about interoperability was described and explained in order to clarify some terms and some methodologies that are used in the systems interoperability. In this section it will be explained in more detail the already started subject that is focus on the sustainable interoperability, proposed in (Jardim-Gonçalves et al. 2010), and presented in the section 2.4.5. Since that the enterprise interoperability sustainability is a state of the art topic, with some research already done, it will be the base of this dissertation.

3.1. Sustainable Interoperability

In the global market, all the companies and networks which they are part of have a certain behavior with some special characteristics similar to the characteristics of Complex Adaptive Systems (CAS). These characteristics are the trend that the companies and their networks to follow a dynamic and evolutionary behavior. Also, they have some properties in common such as the heterogeneous agents, interaction, autonomy, ability to learn, self-organization, melting zone and coevolution. The main goal of all organizations is to adapt themselves to the market demands and the availability of new requirements and applications, which in some cases, it is just necessary introducing some corrections to the existing ones. However, these adaptations require models and semantic changes which bring complexity resulting in harmonization breaking. All these problems introduce a new dimension to interoperability research, the sustainable interoperability, more precisely, the Sustainable Enterprise Interoperability (Agostinho & Jardim-Gonçalves 2009), (Jardim-Gonçalves et al. 2010), (Coutinho et al. 2012).

3.1.1. Harmonization Breaking

The authors of (Agostinho & Jardim-Gonçalves 2009) designate harmonization breaking, as the interoperability behavior equivalent to symmetry breaking from classical sciences. This comparison is made because in the classical sciences, like physics, certain phenomena can be described in exactly same way even if experiments are carried out under different observational circumstances. This means that the laws describing the phenomena display similar results for similar inputs, i.e. symmetric behavior. Also, experiments have proven that small fluctuations acting on a system may cause it to cross a critical point and evidence an expected behavior, which is called symmetric breaking. In the collaboration networks side when the interoperability is established, the set of organizations within a network demonstrate stability, exchanging e-messages following established laws. Therefore, networks display symmetry, so, if just one of the network members adapts to a new requirement, the harmony is broken, and the network begins experiencing interoperability problems, like in the classical sciences (Agostinho & Jardim-Gonçalves 2009).

3.1.2. Collaboration Networks and Complex Adaptive Systems

Cybernetic systems operate at the level of basic processes that are relatively undistributed and are limited in scope to the boundaries of a system and by the perspective of management, as well as the systems thinking that is an approach to problem solving when substantial changes occurs in processes leading to disruptions of an overall system. However, neither can deal with major environmental changes of collaborative networks, because real dynamic systems are too complex to manage in a traditional manner (Agostinho & Jardim-Gonçalves 2009).

CAS studies the ultimate interdisciplinary science, focus on how microstate events, whether particles, molecules, human agents or firms, self-organize into emergent aggregate structure. Speaking in a computer way, CAS is focus on the interplay between a system and its environment and the co-evolutions of both. Models of CAS can be used to determine how different patterns of local interactions and organization adaptive behavior impact the overall network behavior and performance. Therefore, CAS can be used to analyze how intervention strategies on the network evolution, namely attempts to shape local interaction patterns and mappings, affect the network interoperability sustainability (Agostinho & Jardim-Gonçalves 2009).

3.1.3. Heuristic Framework for Network Stability Maintenance

It is clearly, with some available literature to prove (Wycisk et al. 2008), that the CAS results in non-linear behavior when changes happen in the systems. This non-linear behavior can result in butterfly events spiraling into positive and negative extremes. In order to avoid this behavior the Integration Intelligence Layer was created in the framework making that the context awareness is demanded in support of intelligence. Also, Monitoring and Decision Support Systems must be considered in the creation of the framework that implements sustainable interoperability in cooperation networks. The result is the Sustainable Interoperability Framework (SIF) that is illustrated in the Figure 3-1 with all layers explained below (Agostinho & Jardim-Gonçalves 2009):

- *Monitoring System* addresses multiple stages, from capturing information to its analysis, and is structured into specific components in order to meet a set of requirements. So, mainly this layer is responsible for detecting the harmonization breaking explained in the section 3.1.1 and analyzing it in order to discovery what causes the anomaly;
 - *Discovery* capabilities in order to detect when new system is added, or updated in the network, thus creating harmonization breaking;
- *Integration Intelligence Layer* is responsible for the system learning that occurs after the Monitoring System detecting the harmonization breaking. Also, when

this happen, it should calculate the required adaptation in the system nodes. The adaptation of the system and the optimization of the maintenance process is made through dynamic model morphisms, using knowledge representation technologies applied to the model management domain;

- *Learning and Adaptability* capabilities in order to learn when changes occur and to help adapt the system facing the new changes;
- *Decision Support System* acts when changes at the internal or interfaces structures of the organization's information systems lead to unexpected situations. When this happen, SIF must consider some kind of decision support, allowing the manager or any other decision responsible to take the final word regarding whether or not to execute the mapping proposed in the adaptation;
 - *Transient Analysis* capabilities to understand how a network will suffer from the transient period;
- *Communication Layer* is responsible to the re-adaptation of a network node and for the communications in the entire business network in such a way that it causes minimal disruption to the other members of the network;
 - *Notification* capabilities in order to inform in what way should the network nodes react leading the entire network to a new interoperable state.

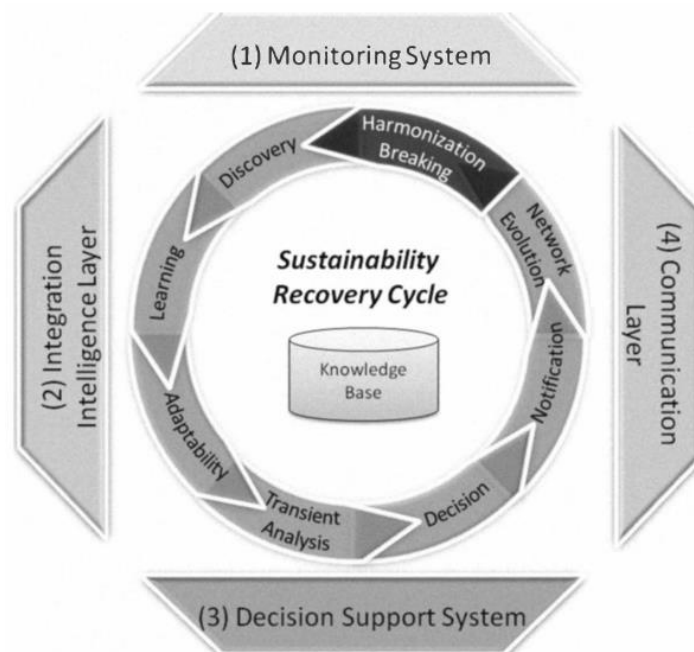


Figure 3-1 - Sustainable Interoperability Framework (Agostinho & Jardim-Gonçalves 2009)

3.2. NEGOSEIO – A collaborative framework for SEI

In (Cretan et al. 2012) and (Coutinho et al. 2012) was proposed a collaborative negotiation framework for improving interoperability in systems and applications by negotiating the enterprise interoperability changes with the proposed interoperable players of business-to-business interactions. Mainly, this framework was proposed aiming the time required to regain interoperability between systems when a harmonization breaking occurs in a previous established party. This out-of-business “downtime” is higher as the complexity of the interoperability, leading to large periods of no operation where all parties loose (Coutinho et al. 2012).

The NEGOSEIO is a framework for NEGotiations for achieving and maintaining a Sustainable Enterprise Interoperability and purpose some solutions to solve some problems of achieving a SEI scenario. These solutions include the creation of negotiation mechanisms that analyze and reduce the impact of the interoperability changes that may occur in the systems, not by avoiding them, but by compromising these changes with the need of maintaining interoperability and with the need of reducing effort and downtime. In order to reach an ideal solution for all member of the interoperability party, the resulting decision may vary from implementing the changes, rejecting the changes, performing partial changes, delaying the changes, selecting new strategies or even determine that interoperability is not possible, desirable or worthy. In the Figure 3-2 is described current and proposed scenarios (Cretan et al. 2012).

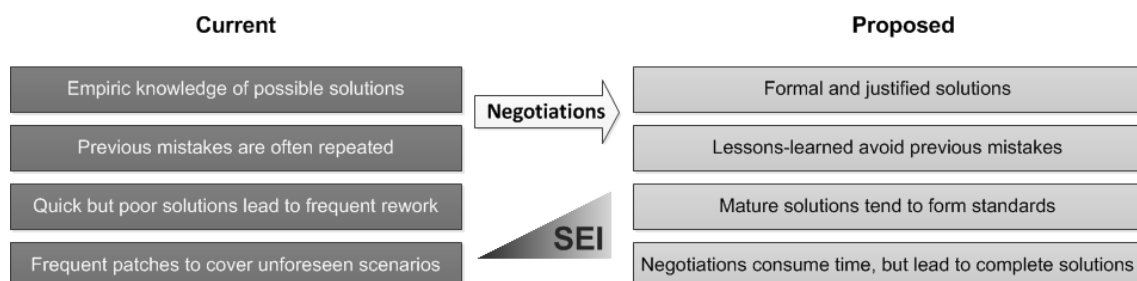


Figure 3-2 - Increasing the SEI through negotiations (Cretan et al. 2012)

3.2.1. The NEGOSEIO methodology

The NEGOSEIO framework is composed by a methodology with several steps and a set of services to accomplish a SEI. The first step of this methodology is responsible to submit a set of assessments to all involved enterprises in order to acquire the enterprises knowledge and their requirements towards interoperability. This shared knowledge must be divided in a “public” part and a “private” part, so that the “private” part of knowledge needs to be kept safe from other enterprises aiming the competitiveness of the enterprise, while the “public” knowledge will be used by the interoperating parties to reach a desirable interoperable state (Cretan et al. 2012).

The second step of the NEGOSEIO methodology is to model the captured business knowledge into MDAs and MDI, where the MDA will responsible to define the solutions and architecture foundations and the MDI will allow a flexible horizontal transformation of data towards interoperability on all MDA layers, defining and understanding of how the interoperability between the enterprises may be defined on the MDA abstraction, from business layers to the computerized layer (Cretan et al. 2012).

The third step of this methodology is the implementation of PSM in the shape of flexible services organized in Service Oriented Architectures (SOAs). This will allow some flexibility and adaptability, where services in a SOA may be improved, updated, adapted and combined in order to build more complex services. Since SOA is independent of the underlying technology, makes it suitable to work together with de MDA and the MDI paradigms. Other feature of SOA is the capability of encapsulate information, which is an asset for the separation in the “public” and “private” knowledge (Cretan et al. 2012).

Distributed computing is a complement to the service flexibility in the implemented solutions. The cloud-computing concept is a very flexible way to deal with scalability, redundancy and security, in terms, not only, of service deployment, but also, of the entire architecture deployment (Cretan et al. 2012), (Coutinho et al. 2012).

The final step is the introduction of a negotiation mechanism. Negotiations start in the MDA/MDI definitions, where the involved parties can propose their interoperability, exposing their own “public” Strengths, Weaknesses, Opportunities, and Threats (SWOT) analysis and will determine if the parties are suited to collaborate among them. When negotiations between already established interoperable parties are about interoperability changes that are proposed by some enterprise that belongs to the party, the negotiation outcomes may vary from, e.g., accepting the change, rejecting it, adopting new consensus solutions or even ending interoperability, as illustrated in Figure 3-3 (Cretan et al. 2012), (Coutinho et al. 2012).

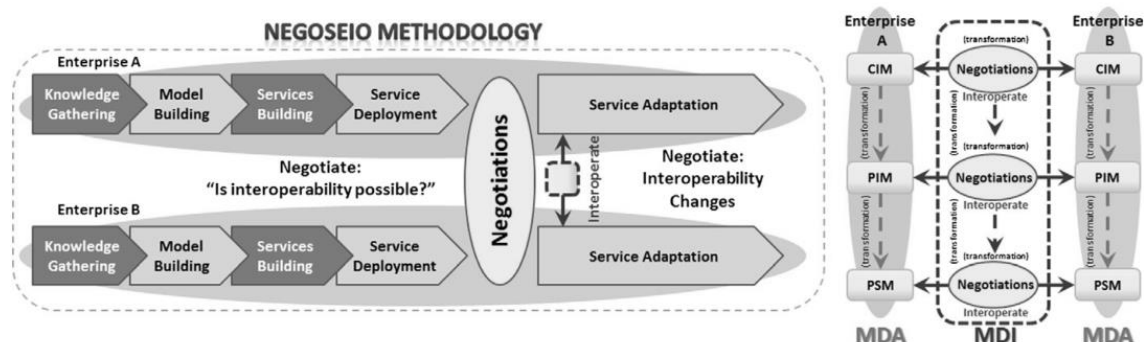


Figure 3-3 - NEGOSEIO methodology (Cretan et al. 2012)

3.2.2. The NEGOSEIO architecture

The NEGOSEIO framework is composed by a multi-levelled set of services that are deployed over a Cloud “Software as a Service” (SaaS) platform adopting the MDA and MDI paradigms, implementing a level Middleware services to handle the heterogeneity issues on the basic interoperability level, e.g., authentication, permissions, communications, syntax, session and data. These middleware services are divided in levels and in the top of the these levels stands the Coordination Services (CS), which is responsible to perform activities, e.g., management of data transactions, agent-based change detection, semantic interpretation, dynamic discovery of services, ontology harmonization and implementation of the business model and negotiation rules. The final level is the Negotiation Manager that interacts with the enterprises applications in the client level, allowing the negotiation actions, e.g., creation of proposals, invitation of new partners and decision about accepting and rejecting proposals (Cretan et al. 2012).

In the Figure 3-4 is illustrated the NEGOSEIO framework architecture and as shown, the cloud-based services that handle and encapsulate the interoperable environment communicate to the Negotiation Manager of each enterprise through web-services and accessed by an Enterprise Service Bus (ESB), which grants the protection of the private knowledge and the separation and disclosure of the public information. In (Cretan et al. 2012) the architecture was applied to the European Space Agency’s Concurrent Design Facility (ESA-CDF), but in this thesis it will be presents generally, which allow it to be applied to a lot of different scenarios, mainly because the architecture was designed to be flexible, using rules engine to implement the dynamic negotiation flows and rules, a SOA to implement the services that support the interoperability and the cloud management to support flexibility and scalability. Moreover, modeling data using standard reference models, data can be persisted on a cloud-based (Infrastructure as a service (IaaS)) infrastructure (Cretan et al. 2012), (Coutinho et al. 2012).

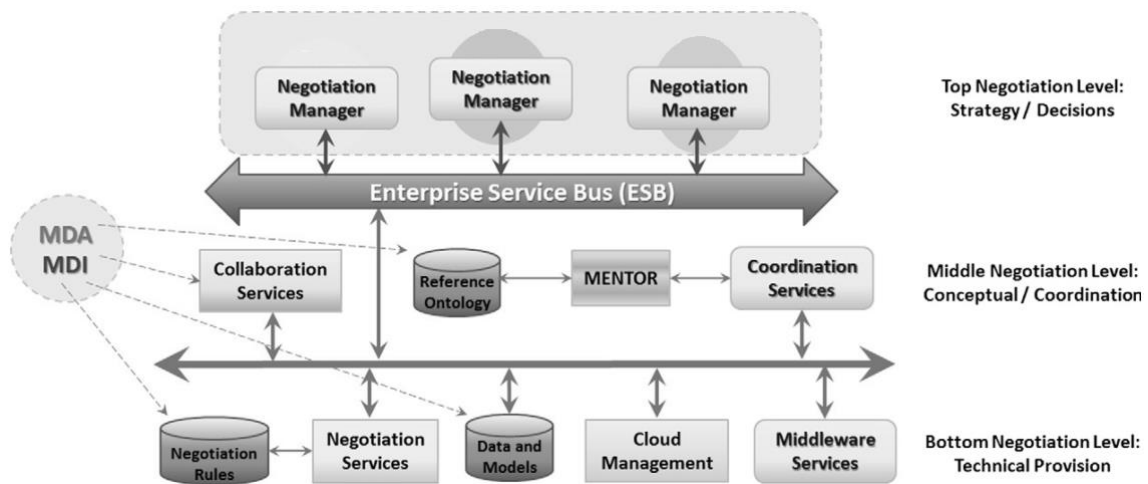


Figure 3-4 - NEGOSEIO framework architecture (Cretan et al. 2012)

3.2.3. Negotiation

Basically, negotiation on the NEGOSEIO framework is divided in three main steps, the Initialization, where it is defined what has to be negotiated and how. The second step is the Refinement and here is where the negotiation participants exchange proposals on the negotiation object aiming to satisfy their constraints. The final step is the Closure and is where the negotiation is concluded (Cretan et al. 2012).

NEGOSEIO allows that different negotiation scenarios can be modeled from a simple case of selection of possible partners and a direct outsourcing of a job, to more complex scenarios of concurrent negotiations with multiple partners to outsource a non-divided job, or even concurrent negotiations with the possibility to dynamically split the job during the negotiation. In order to handle these different scenarios, NEGOSEIO offers several services that are able to evaluate the received proposals and are able to reply with new proposals constructed based on their particular coordination constraints (Cretan et al. 2012).

4. MULTI-AGENT SYSTEM AND RULES ENGINE TO SUPPORT NEGOTIATION ON SEI

As presented in the previous sections, the introduction of negotiation on a SEI scenario requires some additional factors that can affect the development of the entire SEI environment. In (Cretan et al. 2012) the authors already have referred the utilization of agents and rules engine, in the development process of the network, to perform some tasks in the framework. Since that dissertation is about an implementation of a negotiation in a SEI environment, it will be used the same concepts of agents and rules engine, but at least the rules engine technology it will be different from the NEGOSEIO framework, which will bring some extra challenges to development of this proof-of-concept, but this will be described on the next section.

From the list of capabilities that the SEI environment must have, some may be highlighted for their importance, such as the discovery, the learning and the negotiation. Thinking in terms of the prototype implementation, these three capabilities may be assigned to different development technologies. So, for the discovery capability, will be needed an tool that is able to detect changes in the systems, and for that the proposed solution is the use of a Multi-Agent System (MAS), i.e., the use of agent technology, mainly because they can be autonomous, reactive and pro-active. Agents will also be used to do all the communications, since each entity on the envisioned environment contains an agent. The learning capability will be ensured by the rules engine technology, because they can make a clearly separation between the business data and the business logic, which is perfect as knowledge represents the business data. The negotiation will be divided between the agents and the rules engines because the communication part of the negotiation will be implemented in the agents side and the negotiation flow will be implemented in the rules engine side, since the negotiation flow can be represented as a state machine that has the state transitions declared in the form of rules.

In the next sub-sections will be explained more deeply these two technologies and how can they be developed in order to reach a SEI environment with negotiation. It will be explaining also, the architecture of this prototype.

4.1. MAS overview

We can define agents as autonomous entities located in some environment where they are capable of flexible behaviors with several abilities that make them responsive, pro-active and social-able (R. Jennings et al. 1998). Our objective is to create a multi-agent platform where all agents in the platform can communicate with each other. To do this, we will analyze the Java-based agent platforms technologies in order to choose only one that fit in the criteria explained hereafter.

A multi-agent platform or a MAS is an environment where various agents working in collaboration with a series of assigned tasks aiming to reach the overall goal of the system. MAS are autonomous because they operate without human intervention, they are social-able because they interact with other agents in the system through some kind of agent communication language, they are reactive because they perceive and react to their environment and finally, they are proactive because they can take initiatives through the pre-defined behaviors (Leeton & Kulworawanichpong 2012). So, when we refer to agents we mean interactive and autonomous systems where they have the ability to make decisions for themselves and execute actions that are pre-defined (Alonso 2002).

In software development, there are two factors for handling the systems complexity: modularity and abstraction. In order to achieve these factors we use agents or more precisely MAS because they represent a powerful tool to add modularity to systems. They can turn a complex software system into a society of cooperating autonomous problem solvers (Massawe et al. 2010).

4.1.1. MAS as a technology for a negotiation SEI environment

One of the many advantages of the MAS technology according to Alonso (Alonso 2002) is that an agent-based approach can often provide an effective solution when the domain involves a number of distinct problem solving entities which are physically or logically distributed, which means that is possible to develop a distributed multi-agent application through MAS, making it the base of this proof-of-concept.

Beyond all the control in the system, MAS is capable of other things. The communication between the agents is another thing that MAS can do easily (Chmiel et al. 2004). In this case, since, each enterprise in the system will be represented by an agent, all the communications in the environment will pass through the agents, making them the communications actors on the system.

Agents can execute very well some programmed task, which adding to the fact that they can be autonomous and they can operate without human intervention, they are the best choice to control a list of interoperability points of an enterprise that is in the SEI environment. This control involves a constant observation of a list of interoperability points, which will fire a trigger when some change is made to the list that can compromise the interoperability between other systems. This trigger will notify the responsible authority that deals with these occurrences.

4.1.2. Choice of a MAS technology

Since agent technology offers a wide range of particular applications is necessary to choose one of them to use in this prototype. The selection criteria were the following aspects:

- Costs to obtain the full version of the platform;

- Documentation available to have enough support on installation and developing;
- Preference on the use of Java programming language;
- Ease on developing and understanding the programming code;
- Must have an easy way to do the agent communication.

The analyzed applications were:

- Java Agent Development Framework (JADE) (JADE Agent Framework);
- JACK Agent Framework (JACK Agent Framework);
- Open Agent Architecture (OAA) (Open Agent Architecture);
- Jason Agent Framework (Jason Agent Framework).

Analyzing these technologies one by one, over the selection criteria, JACK can be excluded, as it is a technology that though having a cool interface with the user through their graphical agent development tool and being lightweight, it is a technology that is paid. And worse, it has its own programming language, which is not easy to integrate with the Java language. OAA also can be excluded from the list of possible winners because despite that is a technology that has a well-defined communication language – the Interagent Communication Language (ICL) – and has no costs associated, the last version was released five years ago and is not very well documented. From the last finalists, Jason is the technology that is excluded due to the fact that it uses JADE in order to run a multi-agent system distributed over a network, and it is better to use one single technology to do the job. Despite this, Jason is a great technology with great documentation available, it's free to use and can be programmed in AgentSpeak language or in Java.

Finally, the JADE that from the selected technologies is the most used in research (like (R. Jennings et al. 1998) and (Leeton & Kulworawanichpong 2012)), in commercial applications, and also in industrial applications, entertainment applications and medical applications (Nguyen et al. 2009) and has a very active user and developer community over the entire internet. JADE stood out from the other technologies because has various positive points, such as being free to use, allowing very simple implementations on MAS, using the Foundation for Intelligent Physical Agents (FIPA) specification for inter agent communication, provides a Graphical User Interface (GUI) to control the agents in the system and as has it been said previously, has a lot of documentation available over the internet and in the JADE website (JADE Agent Framework). This comparison can be viewed in a simple way in the Table 4-1.

Other features that JADE technology can offer are listed below (Bellifemine et al. 2003):

- Pro-activity, since JADE agents can control their own thread of executions;

- Versatility, because JADE provides a set of Application Program Interfaces (APIs) that are independent from the network or the Java version;
- Ease of use, since JADE APIs reduce the time spent in the development of the agents comparing with the utilization of the Java traditional packages.

Table 4-1 - Agent technologies comparison

		Criteria aspects				
		Costs	Documentation	Java oriented	Programming language	Agent communication
Technology	JADE	Free	Very Good	Yes	Java	FIPA
	JACK	Paid	Good	No	Own	FIPA
	OAA	Free	Poor	Yes	Java/Others	ICL
	Jason	Free	Very Good	Yes	Own/Java	Speech-act based

4.1.2.1. JADE

As presented above, JADE uses the FIPA specifications for inter-agent communications, more precisely it uses the Agent Communication Language (ACL) to perform their communications. ACL is not only a high level and message-oriented agent communication language it is also a logical layer of communications protocols. This protocol defines the type and the meaning of the messages changed between the agents (X. Liu et al. 2011). JADE also includes all libraries required to develop agent applications and the runtime environment in order to accommodate the agents. Each instance of JADE runtime that contains agents is called Container and the group of all containers is called Platform, as we can see in JADE architecture illustrated in Figure 4-1 (Bellifemine et al. 2003).

In (Bellifemine et al. 2003) the author described JADE as an enabling technology, a middleware for the development and run-time execution of peer-to-peer applications which are based on the agents paradigm and which can seamless work and interoperate both in wired and wireless environment. JADE is fully developed in Java and is based on the following driving principles:

- *Interoperability* – JADE is compliant with the FIPA specifications. As a consequence, JADE agents can interoperate with other agents, provided that they comply with the same standard;

- *Uniformity and portability* – JADE provides a homogeneous set of APIs that are independent from the underlying network and Java version. More in details, the JADE run-time provides the same APIs both for the Java 2 Platform, Enterprise Edition (J2EE), Java 2 Platform, Standard Edition (J2SE) and Java 2 Platform, Micro Edition (J2ME) environment. In theory, application developers could decide the Java run-time environment at deploy-time.
- *Easy to use* – The complexity of the middleware is hidden behind a simple and intuitive set of APIs.
- *Pay-as-you-go philosophy* – Programmers do not need to use all the features provided by the middleware. Features that are not used do not require programmers to know anything about them, neither adds any computational overhead.

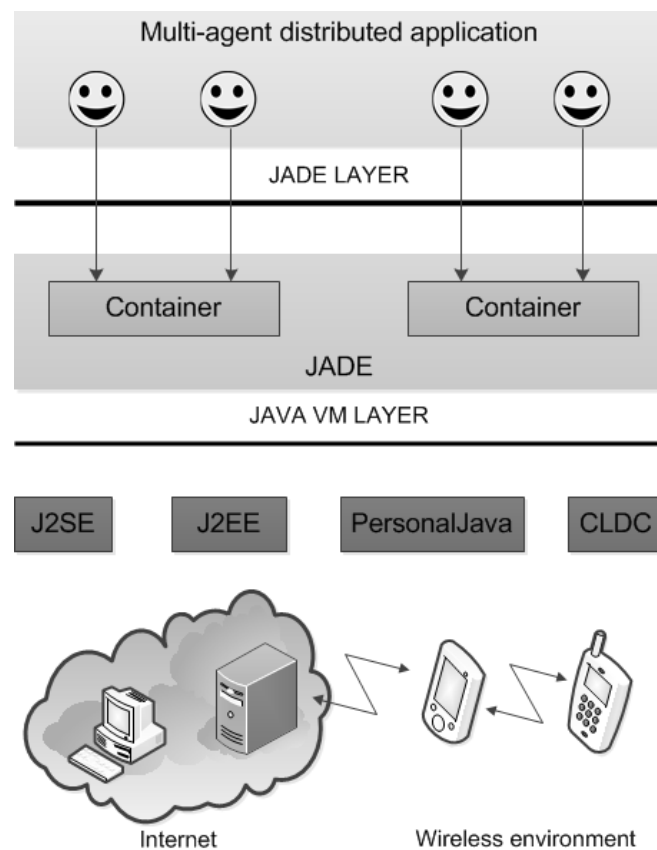


Figure 4-1- The JADE architecture (Bellifemine et al. 2003)

4.1.2.2. FIPA

FIPA (Foundation for Intelligent Physical Agents) is an Institute of Electrical and Electronics Engineers (IEEE) Computer Society standards organization that promotes agent-based technology and the interoperability of its standards with other technologies. FIPA just not promote a technology for a single application but promote a set of general technologies for

different application areas where developers can integrate to make complex systems with a high degree of interoperability.

FIPA is responsible to define 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, then they identify the roles of some key agents necessary for managing the platform and to finish they describe the agent management content language and ontology. An agent platform needs to have three mandatory roles, such as:

- The *Agent Management System* (AMS) that is responsible to exerts supervisory control over access to/use of the platform. This agent is also responsible to for maintaining a directory of resident agents and for handling their life cycle;
- The *Agent Communication Channel* (ACC) that provides the path for basic contact between agents inside and outside the platform. This agent is the default communication method, which offers a reliable, orderly and accurate message routing service;
- The *Directory Facilitator* (DF) which provides a yellow page service to the platform.

Other specification is the utilization of the ACL by the agents to exchange messages which is a language describing message encoding and semantics, but it does not mandate specific mechanisms for message transportation.

4.2. Rules Engine overview

With the current software development methodologies, in order to fit the complexity and frequent changes of software requirement, too much time and money are being spent. Since then, when a new problem comes out, it's necessary to develop new things. With the utilization of rules engines, we can save some time and money because the reasoning of the Java-based rules engine technology is to make a clear separation between the business logic and the business data, i.e., separate the application code (the code that generally is not modified) from the logic code (the code that can be modified to change the logic of the application) (Gang Zhang et al. 2010). In a more practically way, the basic task of a rule engine is to compare the submitted data (the facts) with the business rules (the rules) in the engine and when the engine found a match, according to the logic of the rule, the rule engine can execute any specified operation (D. I. Liu et al. 2010), this task is made by an inference engine that is a part of the rule engine and that will be explained later.

The rules engine technology offers various advantages on using it on the today's applications. These advantages are listed below (JBoss Drools team 2012):

- Declarative programming, i.e. rules engines allow that we say “What to do” and not “How to do it”;
- Logic and data separations as we explained above;
- Speed and scalability, since the pattern matching algorithms like the Rete or Leaps provide very efficient ways of matching the rules;
- Centralization of knowledge, by using rules we can create a repository of knowledge which can be executable;
- Understandable rules, i.e. it’s possible to write rules that are very close to natural language.

Most of all rules engines have a structure with three modules, which are the Production Memory, where the business rules (rules) are stored, the Working Memory, where the business data (facts) are stored, and the most important, the Inference Engine, where a pattern matcher exists that compares the data of the rules and the facts and adds to the agenda the rules that satisfy the facts. The agenda will manage the execution sequence of the rules which are chosen by the pattern matcher and executed by the executions engine. In Figure 4-2 is illustrated the traditional rule engine architecture like was explained before. This architecture is used in the most used rules engine technologies (D. I. Liu et al. 2010). Most of them also use an enhanced implementation of the Rete algorithm to do an efficient pattern matching over the business rules.

The Rete algorithm is located in the pattern matcher sub module and is an efficient pattern matching algorithm. Rete will give an efficient method to implement the matching state from the three states present when the facts are being asserted by the rules engine which are matching, selecting and implementing, since the Rete is pattern matching algorithm. Inference engine in the most of the cases also has two modes of reasoning, the Forward-Chaining (FC) and the Backward-Chaining (BC). FC is “data-driven” and uses the rules to deduce the result from the initial facts and the BC is “goal-driven” and searches in the facts that satisfying the hypothesis (D. I. Liu et al. 2010).

With all the reasons presented above, it is clearly that the rules engine technology became a very important tool in the software development market, not only in the commercial field but also in research, as we can see in (Bayegan & Moslehi 2011), (Xu & Xie 2008) and (Yin et al. 2012). As proof of this we can see in (Kim et al. 2010) that are being done some research over the rules engines in the mobile market, which is a market in continuously expansion.

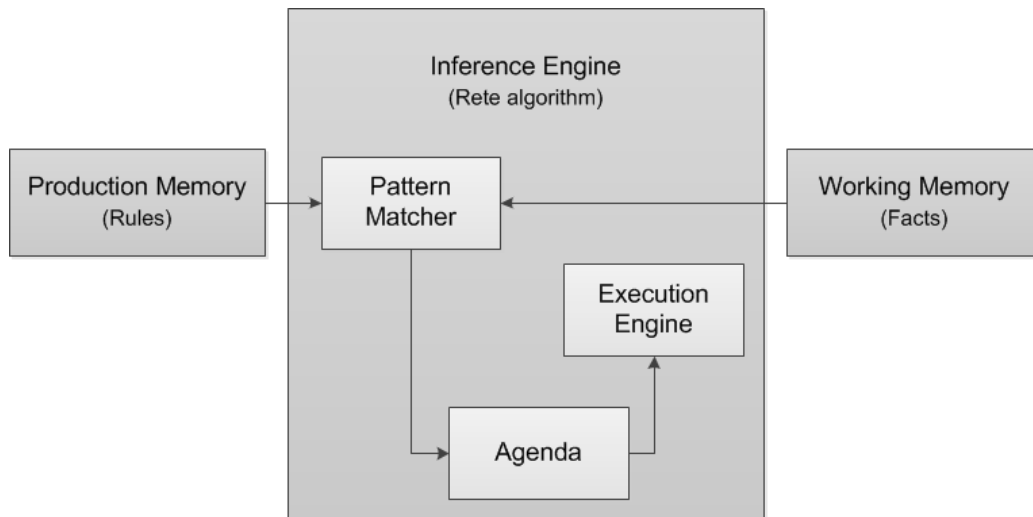


Figure 4-2 - Traditional Rules Engine architecture (D. I. Liu et al. 2010)

4.2.1. Rules Engine as a technology for a negotiation SEI environment

With all features that were presented above, it is clearly that the rules engine technology allows a lot of useful characteristics that can be an asset, not only, to introduce the negotiation factor into the SEI environment, but also, it can perform some tasks in the SEI environment. Basically, these tasks are the negotiation flow and the knowledge manager, once the negotiation flow corresponds to the negotiation part and the knowledge manager to the SEI environment.

In more practically terms, rules engine can learn from negotiations that occurs in the SEI environment once this technology allows a great control of the rules in the production memory, inclusive, allows adding more rules to the production memory in runtime, which means that if we represent the knowledge in form of rules we will be able to save this knowledge in runtime while the negotiations are running, without stopping the entire system.

Rules engine can also be used to control the negotiations flow as a state machine controller, i.e. if the negotiation transitions were represented in rules and the inputs were represented in facts. So, it is possible to separate the flow from the application core code, which will allow a better control over the flow (state machine).

4.2.2. Choice of a Rules Engine technology

Like as the agent technology, rules engine also have a lot of applications over the web and because of that in this section it will be made an analysis to some of those applications. The following selection criteria were used in order to select only one application capable of satisfying all criteria:

- Cost to obtain the full version of the engine;

- The available documentation in order to have enough support, both in terms of installation and in terms of developing;
- Ease of understanding the rule language;
- Ease of java integration;
- Performance of the engine.

The rules engines applications to be analyzed were:

- JESS (JESS Rule Engine);
- OpenRules (OpenRules Rule Engine);
- Drools (Drools Rule Engine).

Analyzing the previous criteria over the selected technologies, it is possible to conclude that despite all of them are good enough to use in this prototype, only Drools fitted perfectly in the used selection criteria. OpenRules, although being a rules engine that deals well with Microsoft Excel files (which is a great feature because anyone can create or edit this files and is a free product for academic use), the interaction between Excel files and Java is a bit complicated, when compared with a Drools rule file that can easily be created or edited, even in runtime. About JESS, is a technology widely used in research and have some positive points, like the fact that it is free to use for academic usage. Like Drools, JESS uses an enhanced Rete algorithm and the rule language is actually not very complicated, but the last version of JESS was released four years ago, which probably means that some features may be obsolete. Comparing with Drools, JESS has much less documentation available.

Hence the chosen technology to perform the pretended tasks was Drools, which comparing with other rules engines technologies that were been analyzed has some advantages, such as the different possibilities to store rules. In this case, it will be used the Drools specific rule language, which is very accessible and very easy to understand. The existing documentation is extensive, and it is an open source project, an asset when we are developing a prototype with no costs associated. Drools offers various products to connect with the rules engine, like a business rule manager for example, but it is only necessary the module that contains the rules engine to benefit from the advantages of Drools. The available documentation can be found in the Drools website (Drools Rule Engine) as well as an issue tracker that can help anyone found any bug in the Drools platform.

Table 4-2 - Rules Engine technologies comparison

		Criteria aspects			
		Costs	Documentation	Rule language difficulty	Java integration
Technology	JESS	Free	Good	Easy	Good
	OpenRules	Paid/Free	Very Good	Very Easy	Not so Good
	Drools	Free	Very Good	Very Easy	Good

4.2.2.1. Drools

Drools (Drools Rule Engine) provides a unified and integrated platform for Rules, Workflow and Event Processing divided in some projects, like the Drools Guvnor that is a business rules manager, the Drools Expert that is the rule engine, the Drools Fusion that is an event processing/temporal reasoning, and some more. For this dissertation it is necessary only one module of Drools, the Drools Expert that is a declarative, rule based and coding environment, which allows to put the focus on "what it is you want to do" and not the "how to do this".

Drools offers some features that are important assets for this prototype, like the utilization of the Rete algorithm that supports FC and BC. Drools provides an Eclipse-based IDE to simplify the utilization of this technology, since the Eclipse Integrated Development Environment (IDE) is a great tool to develop in Java. The efficient integration of Drool with Eclipse, allows the developer running a Drools application in Debug mode, passing through the rule file, viewing step by step what is being executed. Drools uses a simple native language to represent the rules and it is very easy to integrate with the Java which is a great asset when creating and editing new rules is very important on this prototype.

4.3. Negotiation on SEI Framework and Architecture

The proposed prototype in this thesis is focused on proving that the interoperability between two or more systems is much more controllable and stable when negotiation is used to reach an interoperability state between the interoperability participants, ensuring that the participants will spent less time to modify their systems to handle the changes proposed by some participant system. This approach, in general, should produce better results than having systems where one of them performs changes unilaterally, leading the other ones to adjust its

system in order to continue with previous interoperability. This often carries a lot of time and produces poor and immature solutions. With the proposed solution, if a system wants to change something, it will trigger a negotiation, where the involved systems can reach a solution that is probably better for all the systems instead of only one.

The proposed prototype consists in a distributed system composed by a central application and various clients connected to it, forming a basic negotiation SEI environment. Figure 4-3 shows an illustration that transmits the “big picture” of the environment, where is represented the System Controller (the central application) connected to the Trigger Agents (the clients). Note that although only three Trigger Agents are represented in Figure 4-3, the environment supports many more Trigger Agents connected to each System Controller. The names of the applications were chosen by the tasks that each application performs, therefore, the trigger agent is the application responsible for the detection of changes in the system, and the System Controller is the application that controls the entire system, governing the connected agents and the negotiation flow.

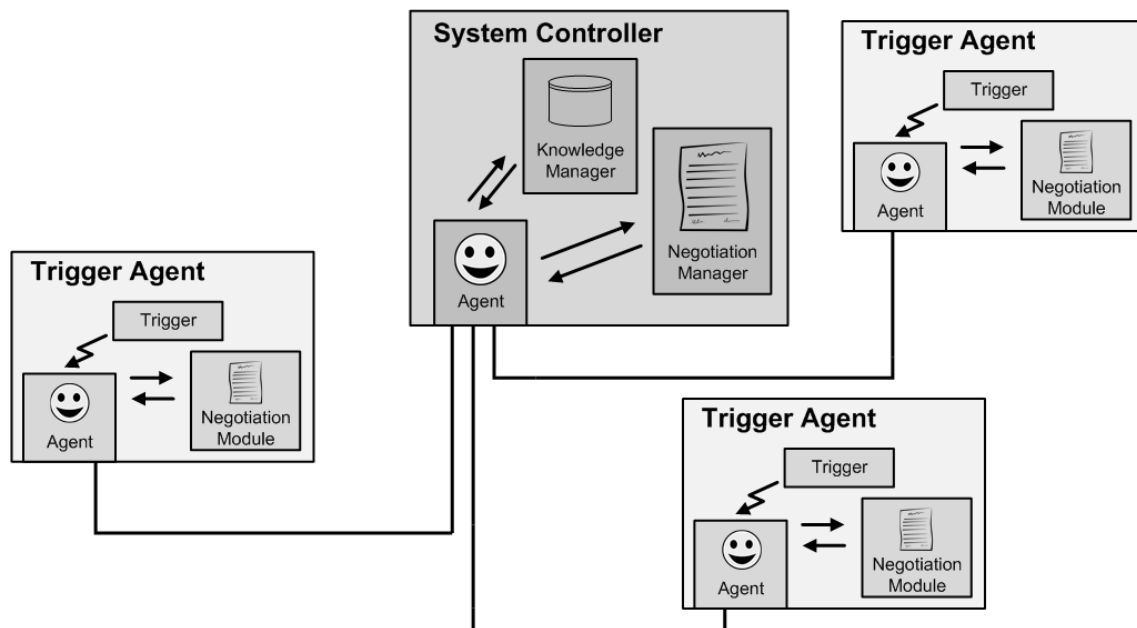


Figure 4-3 - Global vision of the proposed Prototype environment

As illustrated in Figure 4-3 this prototype is divided in two applications, the System Controller and the Trigger Agent. Basically these two applications have complete different behaviors, which are described below:

- *System Controller* is the central point in proposed environment and is responsible to control all interactions between all clients connected to him through the MAS. This application is also responsible to control the negotiations in the environment, not only the negotiation flows, but also to save some negotiation knowledge in order to be capable of helping the clients in the negotiations decisions. These

features will be performed by the rules engine situated in the knowledge manager and in the negotiation manager.

- *Trigger Agent* completes the MAS, as it is connected to the System Controller by its agent. This application, besides performing all communications with the System Controller, also has a task to fire a trigger when a change occurs in the Trigger Agent system. When this happens, the trigger notifies the System Controller that a negotiation round should start to handle the change. The negotiation module is responsible only to notify the user that is necessary to make and action over the current negotiation round.

In order to implement the framework that was described until now, it was created an architecture for each application that performs the entire system. These architectures were created having in mind the main objectives of this dissertation, that are the communication between the involved enterprises, negotiation to reach interoperability, knowledge to help the negotiation and to grant the interoperability sustainable and a GUI to make possible the human intervention and interaction. So, in the Figure 4-4 is shown the architecture of the System Controller that is divided in five important blocks. These blocks are described below.

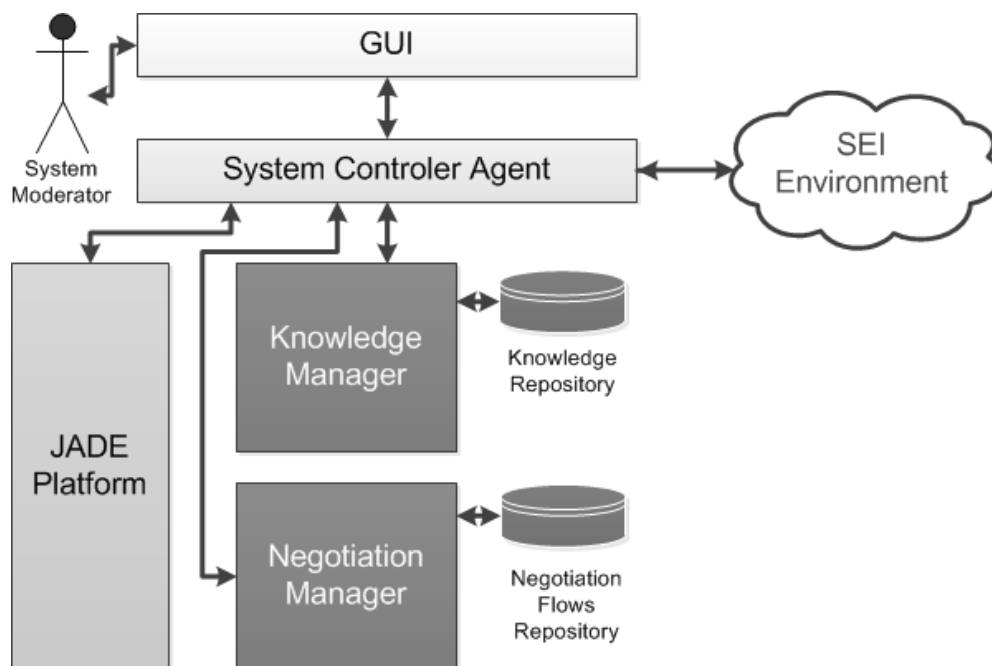


Figure 4-4 - System Controller architecture

- *GUI* is the block responsible to make a bridge between the internal parts of the System Controller to the outside world. i.e., makes the interaction with the human that in this case will be a System Moderator that will be responsible to act on System Controller if necessary;

- *System Controller Agent* is the block that grants the communication with the rest of the environment through the FIPA inter agent communication. This block is also responsible to make decisions and act when it's not necessary human intervention, which make it the principal block on the application due to the fact that is it that controls all operations in the System Controller application;
- *JADE Platform* is where the MAS is created and is where all agents in the systems are connected;
- *Knowledge Manager* is where all the knowledge that is saved in the Knowledge Repository, is managed, i.e., in the course of negotiation, a lot of data about the involved enterprises are exchange, this block is responsible to collect that data and change it into knowledge. The knowledge saved in the Knowledge Repository must be used in order to help System Controller Agent to make suggestions about each negotiation;
- *Negotiation Manager* is where the negotiation flows are controlled. This block maintains the negotiation flows saved in the Negotiation Flows Repository in form of rules, in order to control different type of negotiations between the involved enterprises.

The architecture of the Trigger Agent is shown in the Figure 4-5 and the descriptions of each block are also described below.

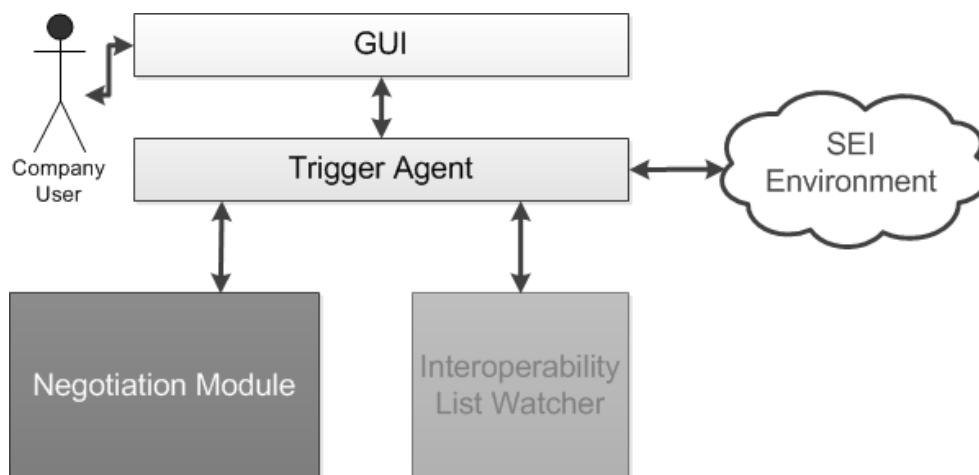


Figure 4-5 - Trigger Agent architecture

- *GUI*, as in the System Controller, is the block responsible to make a bridge between the internal parts of the Trigger Agent to the outside world. i.e., makes the interaction with the human that in this case will be a Company User that will be responsible to act on Trigger Agent if necessary;

- *Trigger Agent*, as the name suggests, is the block that make the communication with the System Controller and more important, is responsible to control the entire operation of the Trigger Agent application;
- *Negotiation Module* is the block responsible for the treatment of the negotiation messages that the System Controller sends to the Trigger Agent, e.g., when the agent receives an negotiation message, the Negotiation Module takes care of the message and analysis it. In some cases it requires the human intervention for the negotiation continues, and in these cases, the GUI needs to be notified through the Trigger Agent;
- *Interoperability List Watcher* is responsible to make a continue observation on the enterprise interoperability list of points that are important to that enterprise to be interoperable to other enterprises. When some change occurs, i.e., when this block detects a possible harmonization break, it is responsible to notify the Trigger Agent in order this notification reach the System Controller. The functionality of this block is a bit complicated, so it not be focused on this dissertation and it will be used a simulation of it.

4.4. System Controller

In the previous sub-section it was introduced the architecture of System Controller and it was explained the functions of each block. In this sub-section it will be explained more detailed all the components of this application and how they interact with each other. Since that this application will be the central application on the Negotiation SEI environment, it will have a more important job, in the environment, that the Trigger Agent application. Being the central application on the environment, all interactions between the connected enterprises will pass through it.

The proposed Negotiation SEI environment is based on negotiations and knowledge, which means that the System Controller will be able to control these factors. The negotiations as described in the previous sub-section will be controlled by the Negotiation Manager that will control them through the rules engine, since that this technology is capable of separate the application code from the logic code (Gang Zhang et al. 2010), what would be an asset if the negotiation flows are saved as logic code in form of rules. Once more, all the knowledge of the system will be saved as rules, since this technology allows a full control of the saved rules in runtime (JBoss Drools team 2012).

4.4.1. Application Overview

The Figure 4-6 illustrates the System Controller UML Use Case diagram that represents the relationships between the actors and the use cases within the system (OMG 1999). In this

particular case, the proposed application system it will have only one actor and will be a person responsible for the environment, capable of “moderate” the negotiations and control the connections of Trigger Agents.

The use cases of the application system are the actions that the actor can perform in the application, and in this case, these actions are mostly, just to let the actor see what is happening in the environment. Looking at the use cases available in the Figure 4-6, it's possible to analyze that the actor will not need to act very often due to the fact that the most use cases are for view information about the state of the entire environment and not to perform action on the environment. That means that the System Controller will bring some automation to the system, making that the System Controller agent, one of the blocks of the System Controller architecture (Figure 4-4), has very important tasks to ensure the proper operation of the environment.

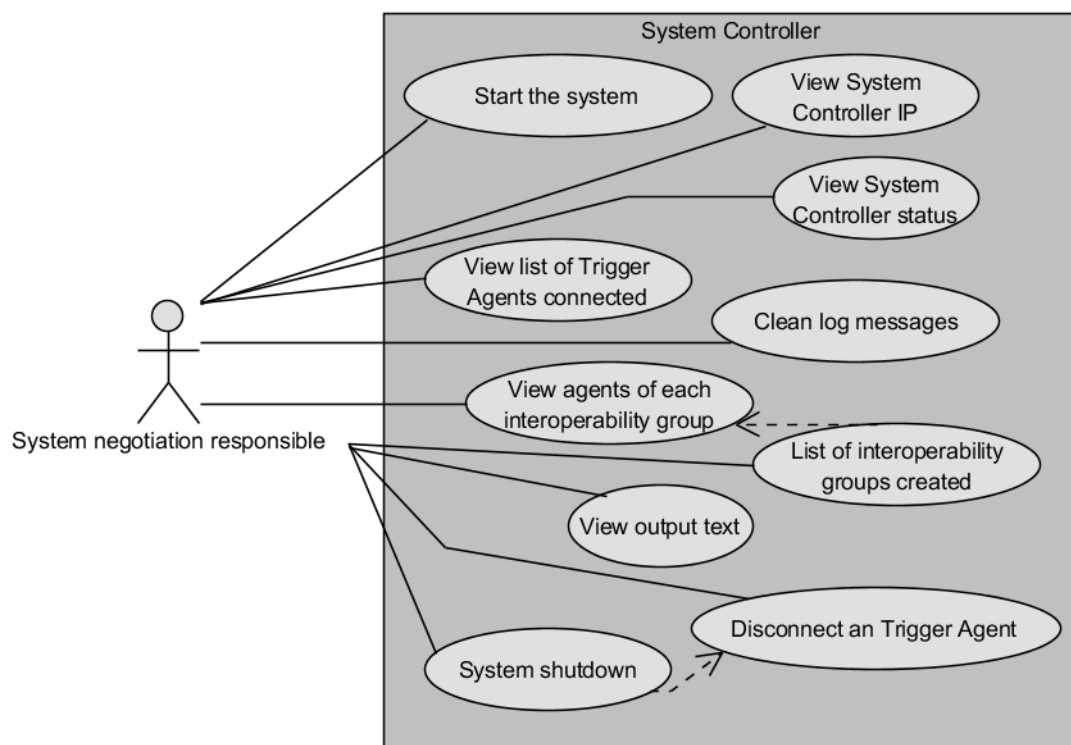


Figure 4-6 - System Controller use case diagram

4.4.2. Application Specifications

Now that the options available to the actor were explained in the previous sub-sections through the UML class diagram, here it will be explain more deeply the application specifications and how the application is constituted. In the Figure 4-7 is illustrated the UML class diagram of the System Controller which represents the static structure of the application, in particular, the things that exist (such as classes and types), their internal structure, and their relationships to other things (OMG 1999).

Basically, as shown in the Figure 4-7, the class diagram is divided in four classes and each class represent each block illustrated in the Figure 4-4. Below is explained the content of each class and how they interact.

- *System Controller GUI* is the class that interacts with the actor, providing essential functions to allow the interaction between the actor and the System Controller. Mostly, the functions provided in this class are directly related to the use cases shown in the Figure 4-6.
- *Knowledge* is the class responsible to interact with the knowledge that is saved during the negotiations. Besides the functions to control the knowledge (“addRule” and “executeDroolsKnowledge”) that interacts directly with the Drools rules engine, this class also offers a function to determine the level of suggestion that the System Controller is able to offer to each Trigger Agent during the negotiations.
- *Negotiation* is the class that keeps all the negotiation information, including the interaction with Drools rules engine to control the negotiation flows. The functions available in this class are mainly to set or to get some variable. This class also contains two types of constants which are the types of negotiation messages and the negotiation status.
- *System Controller Agent* is the core of the System Controller and is the class that represents the agent. Basically, is in this class where all the information and actions pass through since this class is where all other ones are connected.

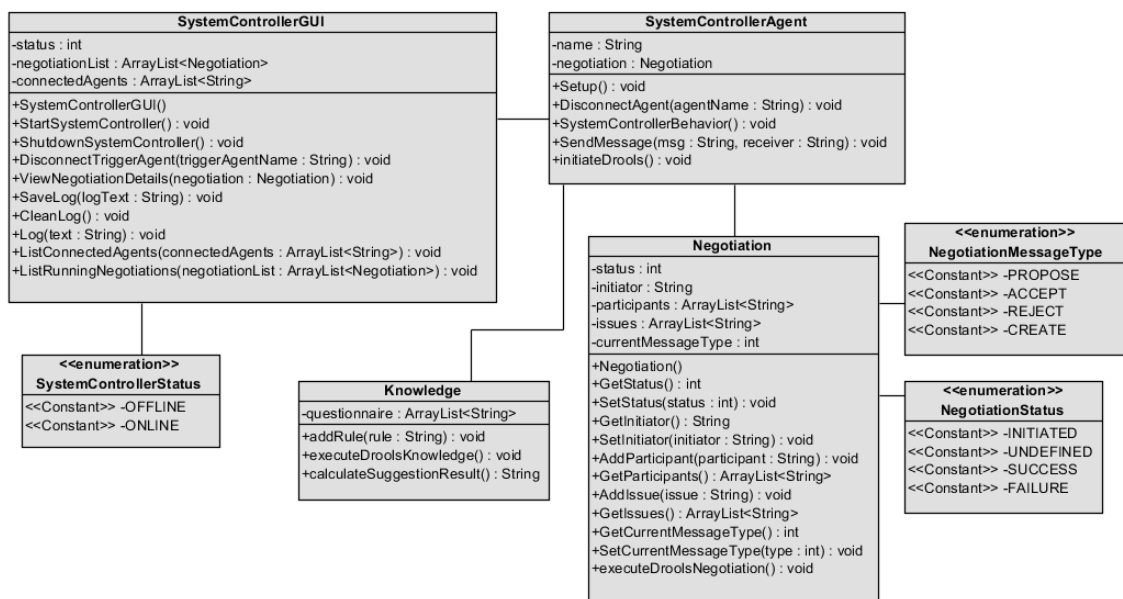


Figure 4-7 - System Controller class diagram

The class diagram presented above was designed in order to include all System Controller specifications, but when the design passes to the implementation some changes on the class diagram may occur, because there are some things that probably require an adjustment when implemented. But despite the changes that possible will be occur, the main structure of System Controller will be the structure presented in the Figure 4-7.

4.4.3. Application Workflow

In the previous sub-sections it were presented the actions that the actor will be available and the classes structure that the System Controller will have, now it will be presented the temporal information of the System Controller through the UML sequence diagram that shows the interactions arranged in time sequence and in particular, it shows the objects participating in the interaction and the sequence of messages exchanged (OMG 1999).

In the Figure 4-8 is illustrated the sequence diagram of System Controller, where is represented the actions to complete a negotiation flow, from the beginning to the end with the interactions between the System Controller (Knowledge Manager, Negotiation Manager and System Controller Agent) and the Environment. Note that in this diagram the interactions between the actor and the System Controller GUI were not included because the focus of this diagram is the internal sequence of actions.

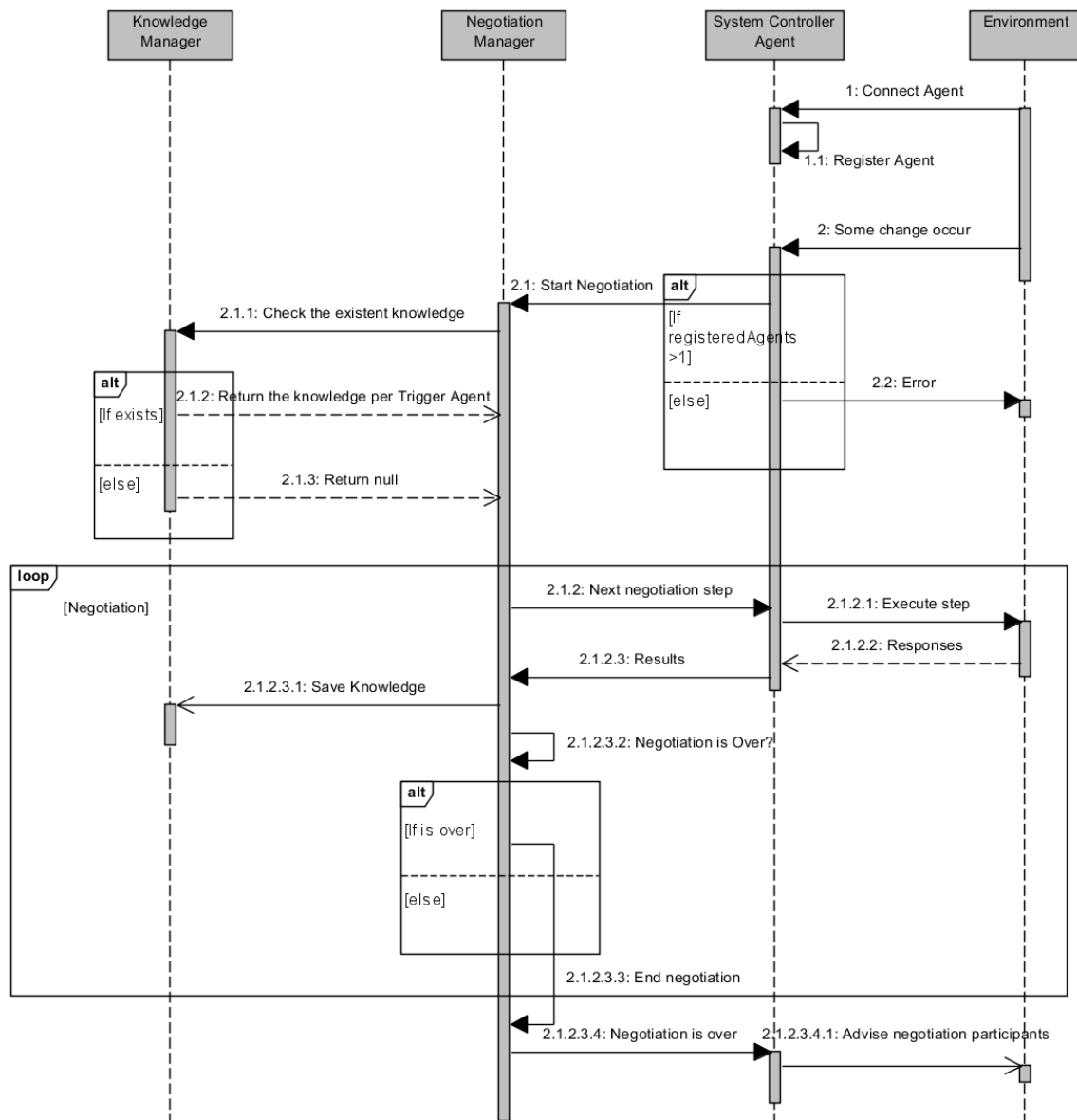


Figure 4-8 - System Controller sequence diagram

The diagram begins with a connectivity message, of a Trigger Agent, from the Environment, which came from a fresh started Trigger Agent. After that, supposing that all involved enterprises are interoperable, forming an interoperability group, some Trigger Agent detects a change in his system and fire a trigger that notify the System Controller. In this step is where the negotiation will begin, only if there are two or more Trigger Agents already registered in the System Controller, because the negotiation to reach an interoperability state only make sense if is made between two or more entities. So, when exist two or more Trigger Agents connected, the System Controller Agent will notify the Negotiation Manager to start the negotiation flow. When the negotiation flow starts, the first important move is to ask the Knowledge Manager if there is some saved knowledge about previous negotiation and if it can be used in that negotiation. After this, the negotiation flow enters on a loop in order to be executed all negotiation steps, which involve all System Controller blocks. This part of the

diagram is a bit widespread because this flow depends on the negotiation flow, defined on the rule files accessed by the Negotiation Manager. To finish the negotiation, when the Negotiation Manager detects that the negotiation is over, the negotiation loop is stopped and the System Controller Agent is notified about it and notify all negotiation participants about the negotiation results.

4.4.4. Learning Methodology

After knowing how the classes in the System Controller interact between each other, in this sub-section it will be explain how the System Controller works with the knowledge. This knowledge is controlled by the Knowledge Manager and as it has been said before, the knowledge is saved in form of rules, more precisely, in Drools rules files.

The knowledge is acquired from negotiation decisions made by the Trigger Agents to the formularies (questionnaires) created in the negotiation initiator and is stored in the System Controller. These formularies in the prototype are static, i.e. the negotiation initiator system is limited to the pre-created formulary that is illustrated in the Figure 4-9, but efforts are being made in this research to change them to dynamic surveys. Each questionnaire is intended to all negotiation participants when a new negotiation round starts and all the answers made by the participants are saved in the System Controller's knowledge base where each answer is saved in form of a rule. These rules are constituted by key elements, like the name of the agent that made that answer, the elements that constitute the questionnaire, the answers to each question and the decision taken by the Trigger Agent.

Questionnaire

What has changed on your system (SWOT) ?

☒ Characteristic 1
☐ Characteristic 2
☐ Characteristic 3
☐ Characteristic 4

Advantages:

☐ More performance
☐ More availability
☐ Energy efficiency
☐ Costs reduction

Drawbacks:

☐ Less performance
☐ Less availability
☐ High energy consumption
☐ More costs

OK Cancel

Figure 4-9 - Questionnaire made by negotiation initiator

An example of a rule can be viewed in the Figure 4-10 and in this rule it is possible to see that when the negotiation initiator creates a questionnaire that has the value “Characteristic 1” selected (true value), the response of the TriggerAgent-2 to this questionnaire will be to reject, i.e., in a previous negotiation the enterprise with the “TriggerAgent-2” associated decided to reject when only the “characteristic 1” was selected. This means that when another negotiation is started, with only the value “characteristic 1” selected, the agent “TriggerAgent-2” maybe want to reject again. Of course that this suggestion is not direct from this rule, because the knowledge base can contain more rules of this type but with accept result which may influence on the suggestion certain percentage.

```
rule "Rule-Example-Reject"
when
    $map : java.util.HashMap(
        this["Energy efficiency"] == false,
        this["High energy consumption"] == false,
        this["Costs reduction"] == false,
        this["More costs"] == false,
        this["Less availability"] == false,
        this["More availability"] == false,
        this["Less performance"] == false,
        this["More performance"] == false,
        this["Characteristic 1"] == true,
        this["Characteristic 2"] == false,
        this["Characteristic 3"] == false,
        this["Characteristic 4"] == false,
        this["Agent"] == "TriggerAgent-2",
    )
then
    resBean.addReject();
end
```

Figure 4-10 - Example of a rule

4.5. Trigger Agent

The Trigger Agent application is a client application that will be responsible to connect the enterprise to the SEI environment. This application as explained in the Section 4.3 it will be responsible to perform all communication between the System Controller and also, it will be responsible to perform a continuous watch over the enterprise interoperability list of points that are important to that enterprise to be interoperable to other enterprises and when some change occurs it will be fired a trigger that will notify the System Controller that will occurs a harmonization break.

4.5.1. Application Overview

Figure 4-11 illustrates the Trigger Agent use case diagram, as described above, represents the relationships between the actors and the use cases within the system. Compared to the use case diagram of the System Controller (Figure 4-6) this has more use cases due to the fact that this application has much more interaction with the actor, compared to the System Controller. The basic use cases are the connect/disconnect Trigger Agent from the

System Controller and the view options that allows the actor to view some important information. The more important use cases are the one that simulates a change in the interoperability list and create new proposal, which will affect the SEI environment.

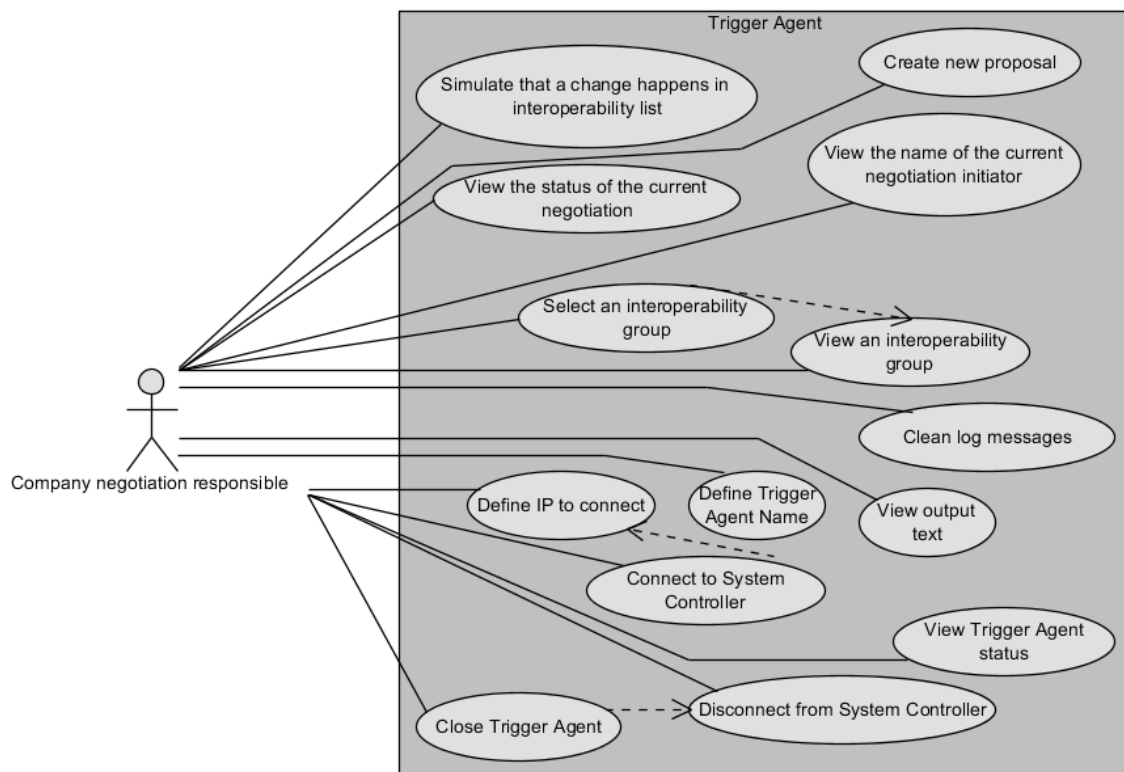


Figure 4-11 - Trigger Agent use case diagram

The use case for the simulation of a change in the enterprise system, is available for the actor only because that in this proof-of-concept the Interoperability List Watcher will not be implemented, so it is necessary a simulation button in order to allow the actor simulates a real change in the interoperability list. Of course that this feature is one of many that will be considered to implement on the future work, because it is an important feature that will bring more automation to the system.

4.5.2. Application Specifications

With all the use cases for the Trigger Agent presented, in this sub-section it will be presented the classes of the application through the UML class diagram, which according to OMG in (OMG 1999), represents the static structure of the application, in particular, the things that exist (such as classes and types), their internal structure, and their relationships to other things. The class diagram for the Trigger Agent application is shown in the Figure 4-12 and all the classes that form the Trigger Agent will be explained below.

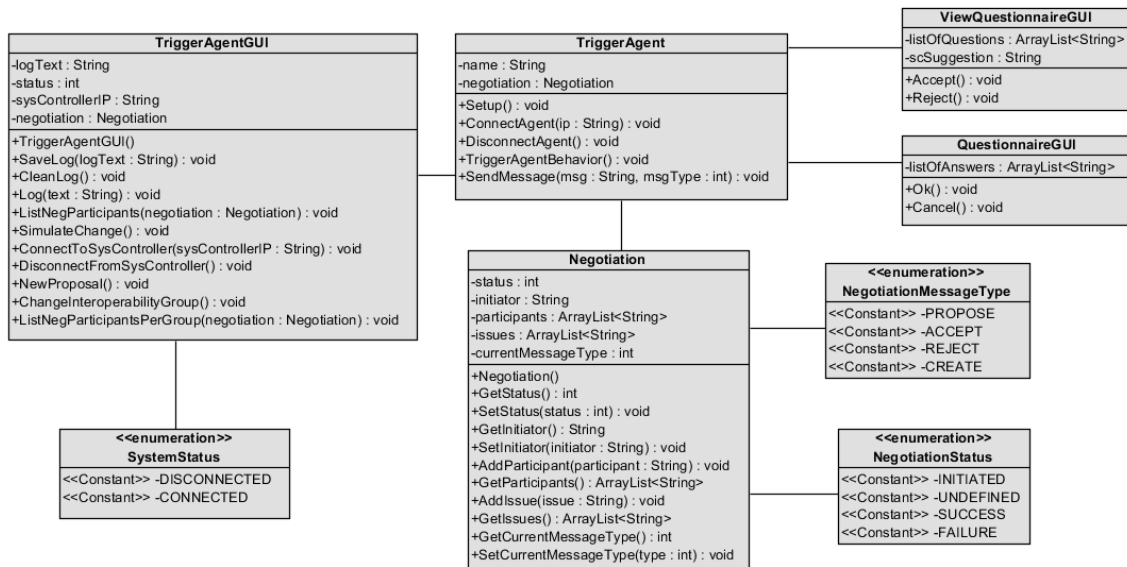


Figure 4-12 - Trigger Agent class diagram

- *Trigger Agent GUI* is the class responsible for the main interaction between the actor of the application and the Trigger Agent application. Basically this class implements all use cases defined in the Figure 4-11 and is the “face” of the application, since that is that class that will show the actor all the information that the System Controller sends to the Trigger Agent.
- *Negotiation* represents the Negotiation Module and is responsible to let the actor know all the information about the negotiations.
- *Trigger Agent* is where the agent is located and is the main class of the application since that is this class that will control all information that is exchanged between the Trigger Agent and the System Controller.
- *View Questionnaire GUI* and *Questionnaire GUI* are two simple GUIs whose functions are to show the questionnaire made by the negotiation initiator to the Trigger Agent and also, to show the actor the questionnaire to be filled when the changes are made by him.

As explained in the System Controller class diagram, it is possible that some changes may occur in this diagram due to the fact that when the development process starts, is always necessary some adjustments in the diagrams to grant that all features specified will work without any problem.

4.5.3. Application Workflow

Now that all options available to the actor and all the classes were presented, in this subsection it will be explained how the internal parts of Trigger Agents, i.e., classes, work together

through UML sequence diagram that shows the interactions arranged in time sequence and in particular, it shows the objects participating in the interaction and the sequence of messages exchanged (OMG 1999), as explained in the Section 4.4.3.

The UML sequence diagram of the Trigger Agent is illustrated in the Figure 4-13 and as in the System Controller, here it will be presented the sequence for a negotiation flow, since that is the main focus of the application.

The sequence starts with the Trigger Agent connection to the System Controller which is represented in the Environment in order to simplify the diagram. When the Trigger Agent is connected to System Controller, it could send/receive messages to/from System Controller, which means that in the second action in the diagram, Trigger Agent simulates that a change occur in his interoperability list and notify the System Controller about this event with a message. This trigger will make System Controller start a negotiation round and with that all involved Trigger Agents will receive a message informing the initiation of a new round of negotiations. When the Trigger Agent receives that message from System Controller it will enter on a negotiation loop and all negotiation messages that are received by Trigger Agent will be treated inside of the negotiation loop. The first action that Trigger Agent does to the received negotiation messages is to check if the message is to inform the end of negotiation round, in order to exit the loop and treat that type of message.

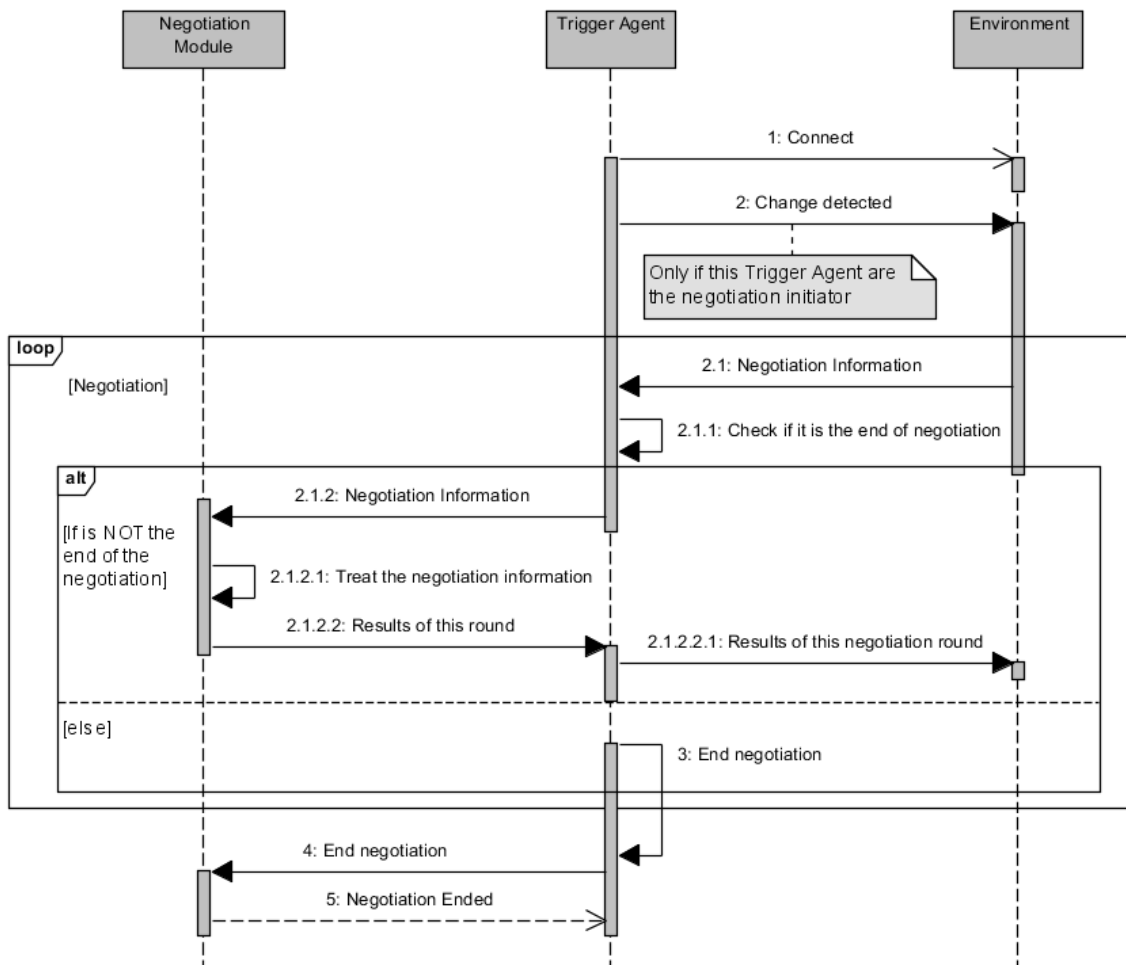


Figure 4-13 - Trigger Agent sequence diagram

4.6. Usability Cases

Earlier it was explained how the proposed prototype works through the UML diagrams, in this sub-section it will be presented some scenarios to be implemented on the system. These scenarios will differ in the negotiation flows and will be the most important scenarios that enterprises can face in their lifecycles, passing from the easiest scenario to a much more complicated scenario.

In the Figure 4-14 is illustrated the first usability case that the system will perform. This scenario is the negotiation method called Block. This negotiation method can be characterized by the specific number of the Trigger Agents (Enterprises) that will perform the necessary modification to bring interoperability to the system again. In this method, the job of regaining interoperability is given only to one enterprise. This means that during the negotiation, and after enterprises changing some proposals between them, only one these proposals will be accepted and is it that will be taken into account to perform the necessary modifications to the system.

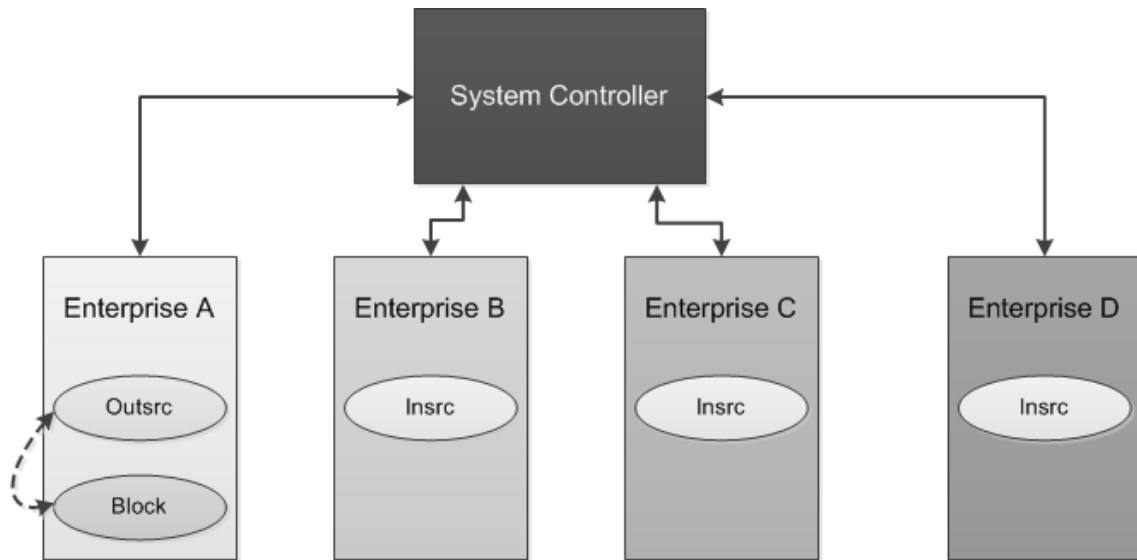


Figure 4-14 - Block negotiation method scenario

The second scenario is the negotiation method called Split and is illustrated in the Figure 4-15. This method, as the name suggests, is when the job of rearranging the system is done by different proposals, i.e., partners. This method shows a great advantage when the solution for regaining interoperability is a bit complicated, which can be simplified if the work is divided in some parts.

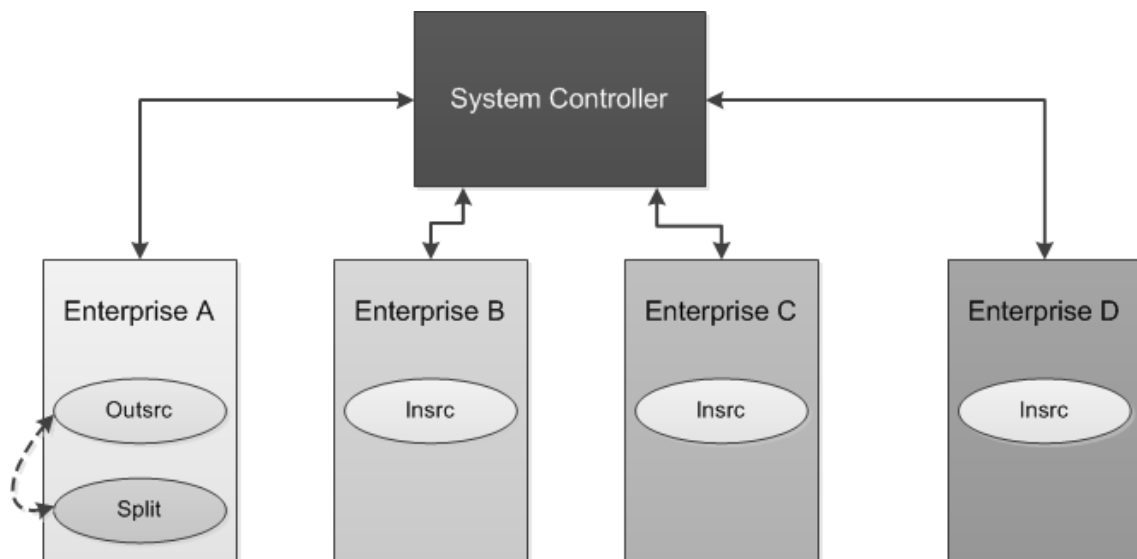


Figure 4-15 - Split negotiation method scenario

5. PROOF-OF-CONCEPT IMPLEMENTATION

The framework and architecture that are proposed in the sub-section 4.3 need to be validated in order to prove their viability which requires an implementation of what is presented in the sub-sections 4.4 and 4.5. Since that this dissertation is based on the SIF proposed in (Agostinho & Jardim-Gonçalves 2009) and in the NEGOSIO framework proposed in (Cretan et al. 2012), this proof-of-concept also be based on these frameworks, although it will be more simple as it was presented in their original proposed documents.

Some technologies were used in the development of this proof-of-concept such as the JADE (JADE Agent Framework) that was responsible for the agents implementation and Drools (Drools Rule Engine) that implements the rules engine, which were already introduced in the sub-sections 4.1 and 4.2 respectively.

5.1. Application Scenarios

In order to maintain a system sustainable interoperable with negotiation involved between the systems clients, were presented in the sub-section 4.3 an architecture that is capable of accomplish this task. To demonstrate the reliability of this framework, will be presented two application scenarios that will simulate real scenarios of enterprise negotiation methods to reach an interoperability state.

The different scenarios that will be presented next have the objective to simulate a real negotiation situation between various enterprises aiming to reach a stable state which all of them are interoperable in the system. So, these scenarios take into account that the system is composed by a System Controller already started and working and some enterprises connected to him, those are the Trigger Agents. Since that a negotiation only make sense when exists more than one actor, System Controller will only allow that the negotiation begins when are more than one Trigger Agent connected. The two negotiation method that will be presented as scenarios will be presented below:

- **Block**, this method is when a negotiation is initiated by some enterprise and the System Controller will choose only one presented proposal to be the one that will be used to implement the necessary changes in the system to allow regaining interoperability. The chosen proposal is associated with the enterprise that made the proposal and that enterprise will be responsible to implement that changes.
- **Split**, as the name suggests is when the System Controller can choose, from all received proposal, some proposals that will be responsible, together, to modify the system in order to bring the interoperability to the system. The number of the chosen proposals will be the split number, defined in the System Controller. This

means that the chosen proposals will make that the enterprises will divide the work between them.

Since the proposed framework is also based in knowledge, during the negotiation, both in the block or split method, System Controller will learn from all negotiations. This knowledge will help System Controller to make some decision suggestions to the negotiation participants, when new proposals are received and help them also with their previous decisions to the proposals. As knowledge is a thing that grows with time, in this case the System Controller knowledge will grow with the number of negotiations, which means that the help that System Controller can provide to the Trigger Agents will be more accurate as time passes and the number of negotiations grows.

In the two presented scenarios will be assumed that System Controller will have four Trigger Agents connected to him, representing four different enterprises, the enterprise A, B, C and D.

5.1.1. First Scenario – Block negotiation method

This scenario demonstrates how the proposed architecture works in the Block negotiation method. An overall picture of this method can be viewed in the Figure 5-1, where it is shown a possible negotiation round over the Block negotiation method. Basically, in the Figure 5-1, the Enterprise C starts the negotiation by simulating a change in their system and with that, they create a proposal to go with the change notification. After Enterprise A and D reject the first proposal, Enterprise A creates a new one. Enterprise B responds to the first and second proposals and next, Enterprise D and C accept the second proposal. In this state, where all proposals are answered, System Controller can evaluate the 2 proposals and select the winner proposal and notify the winner Enterprise, that in this case is the second proposal from Enterprise A.

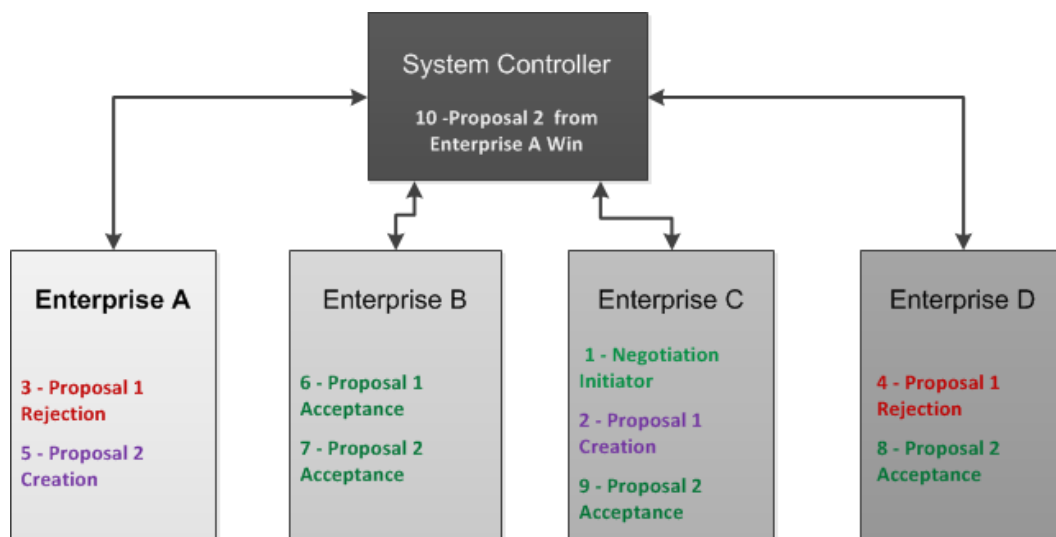


Figure 5-1 - Block negotiation method scenario detailed

In this particular case, System Controller has chosen the Proposal 2 to be the one that will be implemented in the system. This choice was made because among all received proposals, the Proposal 2 was the one that had one hundred percent of acceptance, which is greater than the Proposal 1 that had only thirty three percent of acceptance. The value of the acceptance percentage can be defined in the System Controller, which in a future version of the prototype, possible can be defined by all involved enterprises or even other method that is more appealing for the enterprises. For default, this value is defined at fifty percent which will make that in a negotiation round with several proposals, when none of them will pass the fifty percentage of acceptance, the negotiation will be failed, and possible it will be require passing to the second scenario, which will bring the Split negotiation method.

Not represented in the Figure 5-1 is the knowledge that System Controller acquires with the represented negotiation round. Of course, when the system starts does not exist any knowledge in the system, but after the first negotiation round, some knowledge was acquire and with that, the following negotiation rounds will have the help of that knowledge. This knowledge will make that the involved enterprises will answer faster and more accurately to the received proposals. Of course that these two characteristics becomes much stronger with the lapse of time and the number of negotiation rounds.

5.1.2. Second Scenario – Split negotiation method

The second scenario is the Split negotiation method that is represented in the Figure 5-2, where it is shown an example of a negotiation round over the Split method. In this case, the negotiation initiator is the Enterprise D which also creates the Proposal 1 that is rejected by the Enterprise C. Enterprise B created a new proposal and accepts the initial one. Enterprise A rejects the first proposal and accepts the second one which makes the Proposal 1 with thirty three percent of acceptance. After Enterprise C accepts the second proposal, Enterprise A creates a new proposal, which after the Enterprise D accepts the second proposal, is accepted by the Enterprise B and C and is rejected by the Enterprise D. With all answers to the proposals received by the System Controller, they are analyzed and since that exist tree proposals and the split number is two, means that the negotiation may end with success. The Proposal 1, as concluded before, reached an acceptance percentage of thirty three percent of acceptance which makes it excluded for the winners list, because as the Block negotiation method, in the Split method, the default value of fifty percent of acceptance also is important in order to accept or reject the proposals. With the first proposal excluded, remain the second and third ones. The second proposal was accepted by all enterprises which makes it with one hundred percent of acceptance. The last proposal was accepted by the Enterprise B and C and rejected by Enterprise D which makes it with sixty six percent of acceptance, which is greater than the acceptance percentage value. Thus, the Proposal 2 and 3 are in conditions to be accepted and as the split number is also two (which mean that is necessary to have two proposals to split the

work between them), the negotiation round reach the end with two winners, the Proposal 2 and 3 from Enterprise B and A respectively.

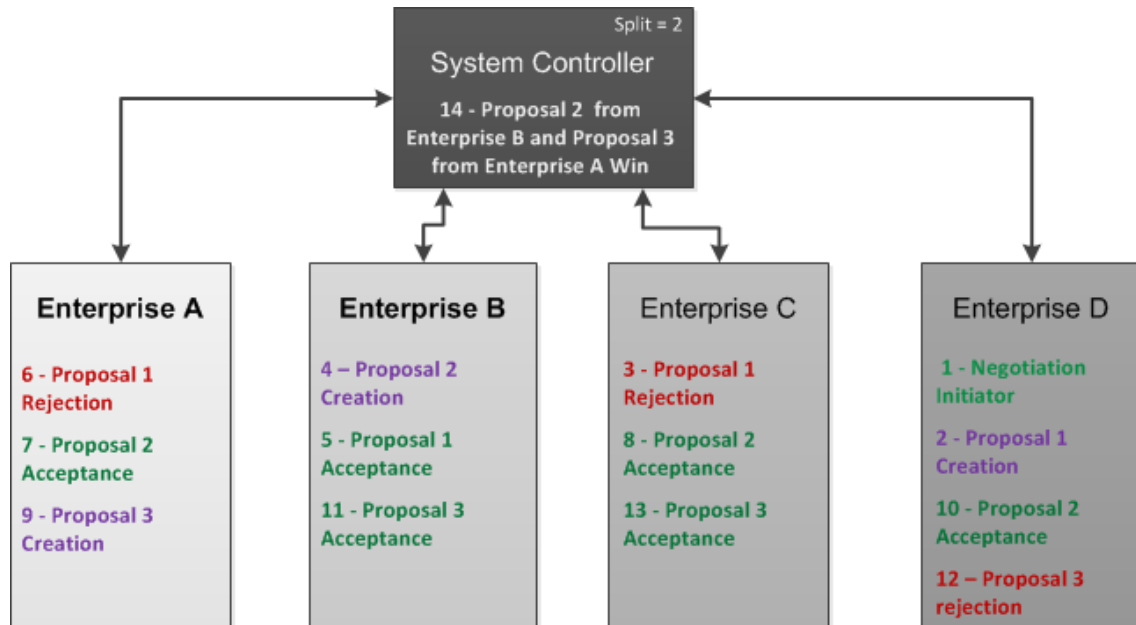


Figure 5-2 - Split negotiation method scenario detailed

As in the Block method and as explained before the conclusion of this negotiation round was dependent from the acceptance percentage value which makes that the proposals that had an acceptance percentage lower than the fixed value, are excluded from the winners list. In this method is added another selection variable, the split number which in the end, the negotiation is succeed only when the number of accepted proposals, that are those that have the acceptance value greater than the acceptance fixed value, are equal or greater than the split value. If are more accepted proposals than the split number, the best ones are chosen to equal the split number.

Once more, the knowledge is not represented in the Figure 5-2, but as in the Block negotiation method, it is present in all decisions made by the Trigger Agents.

5.2. Implementation Steps

The main objective of this proof-of-concept consists on the implementation of a negotiation SEI environment that is represented in the Figure 4-3 and as this architecture are divided in two different application, it is possible to see, more detailed, the architecture of the two applications that forms the pretended architecture, in the Figure 4-4 and in the Figure 4-5. The implementation of this proof-of-concept can be divided in three parts. The first step shows the implementation of the agents through the JADE platform and how these agents are distributed over the applications. The part two explain how the system controller controls the

negotiation flows through the Drools and the final part, the part three is explained how the System Controller manages the knowledge, also through the Drools.

5.2.1. Step 0 – Environment setup

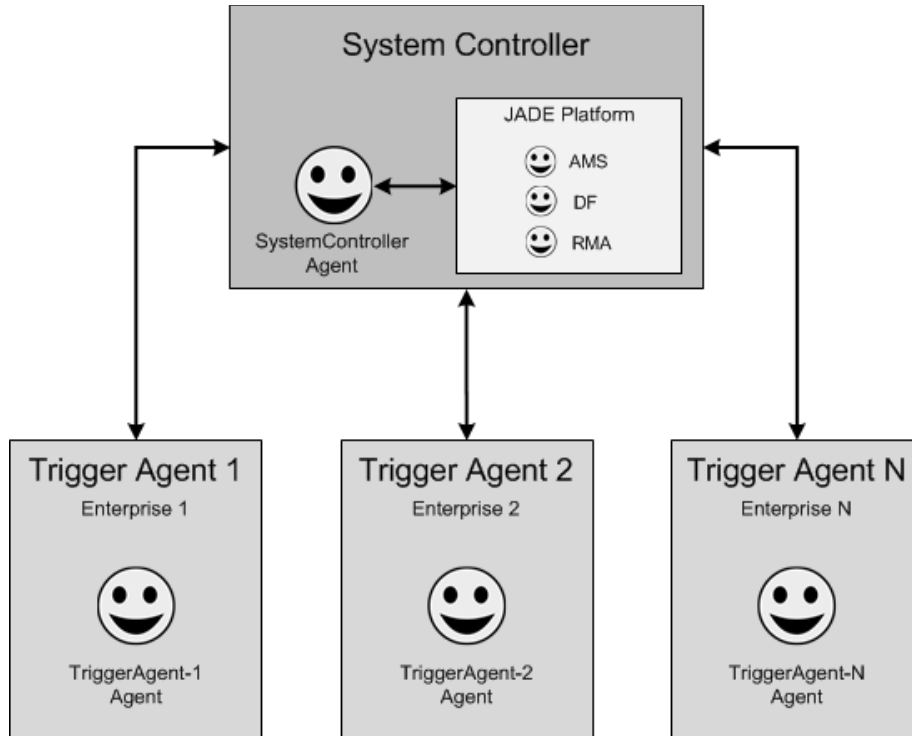


Figure 5-3 - Environment agents setup

In sub-section 4.3 was explained how the proposed framework works, now it will be explained the setup of the agents in the framework. In the Figure 5-3 is represented the agents that make up the system. Since that the number of Trigger Agents can be variable, in the Figure 5-3 they are represented from 1 to N which mean that the number N can be any number. So, the framework starts in the System Controller that initiate the JADE platform and after that starts the SystemController Agent that will be responsible to communicate to the rest of the environment. When the System Controller is initialized correctly, it can start to receiving connections from Trigger Agents. Actually, the connections from Trigger Agents are made to the JADE platform, but the SystemController Agent is notified about these events, in order to control the connected agents. When there are more than one Trigger Agents connected, System Controller is able to start a negotiation round with the connected Trigger Agents.

5.2.2. Step 1 – Negotiation

The most important work on this framework is done in this step, which is the negotiation of the interoperability over the environment. As already presented, the negotiation is controlled by a state machine developed in Drools rule engine. The sequences of the two negotiation methods are illustrated on the Figure 5-4 and Figure 5-5 and basically, the rules that define the

states of the state machine are these sequences. Since that in the sub-sections 5.1.1 and 5.1.2 these two methods were explained in a functionality view, in this sub-section it will be explained how they are implemented in the system.

In the two negotiation methods it is possible to view that the dotted arrows represent some external events to the state machine, and these events are explained below:

- **clone_create(...)** is the event that will create the negotiation round. This event is generated when a change notification is received from some Trigger Agent, meaning that is necessary to start a negotiation round in order to overcome the changes imposed by the initiator system;
- **clone_propose(...)** is the event generated when a proposal is received in the System Controller from some Trigger Agent during a negotiation round;
- **clone_reject(...)** is the event generated when a proposal does not meet the necessary attributes to be accepted, i.e., when a proposal has an acceptance percentage below the value defined.
- **clone_accept(...)** is the event generated when a proposal is good to be accepted, unlike the previous event, here is when the acceptance percentage is greater than the defined value.

When some event is generated, the Drools is executed in order to check if the current state meets the requirements to change to the next state. The state changes are not dependent only to the external event as it is shown in the Figure 5-4 and Figure 5-5, they depend also from the other state variables. These state variables are explained below:

- **name** is the variable that represents the name of the state;
- **status** represents the current status of the state, which in the Figure 5-4, the first state starts with the status start;
- **test_size** represents the size of the interoperability task fixed by the initiator enterprise and is used only in the Block method;
- **count** is the variable that represents the value of the split defined by the System Controller. This variable is used only in the Split method, i.e., in the Block method, count is equals to zero, because the regain interoperability job will not be splitted;
- **localr** is a representation particle of the negotiation method;
- **firsttr** is a representation particle of the outsrc, used by the negotiation participants;

- **extr** is a representation particle of the enterprise that has the accepted proposal.

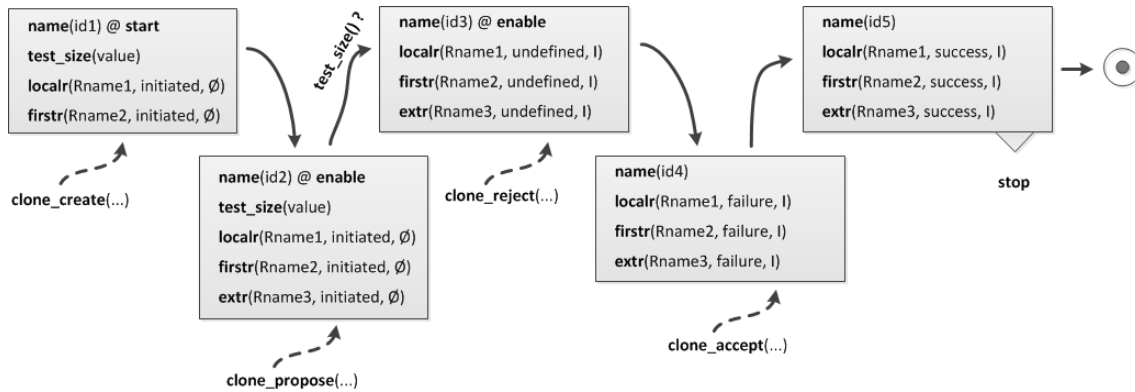


Figure 5-4 - Block negotiation method sequence

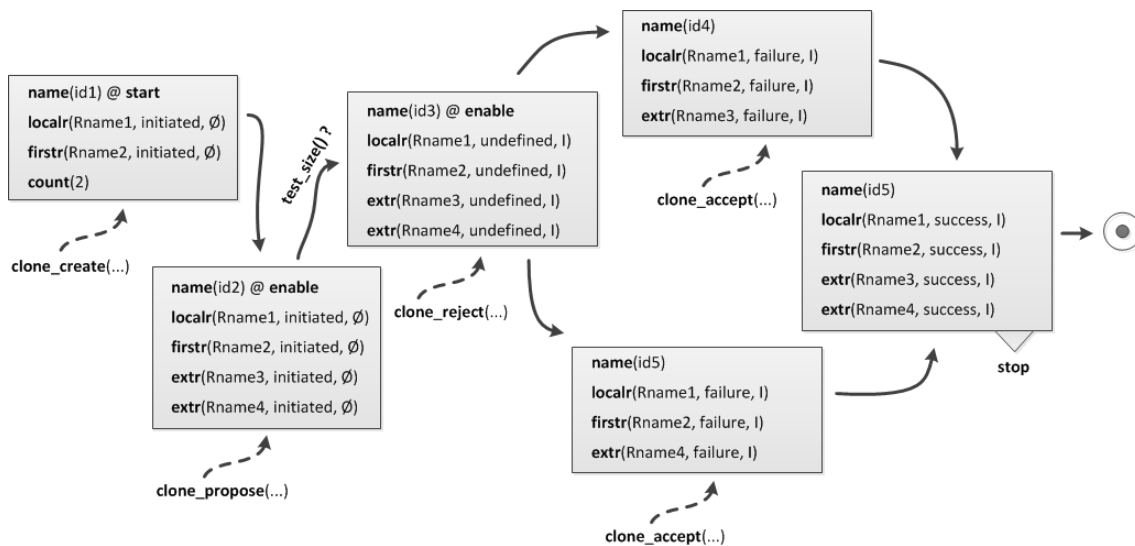


Figure 5-5 - Split negotiation method sequence

Since that the sequence of the two negotiation methods is presented, in the Figure 5-6 is illustrated an example of a negotiation rule. In this specific case, this rule is the first rule of the Block method, which will make that the state changes from #0 to #1. The rule is written in Drool rule language which is very easy to understand. So, below is explained the four numbered rectangles in the Figure 5-6.

1. This rectangle is where the conditions are declared, which in this case will represent the conditions that the current state needs to have in order to change the state, which will make the rectangles 2, 3 and 4 to be executed. As shown in the first rectangle, the current state need to have the “name = #0”, the “status = start”, the System Controller should have received a clone_create event and the count must be equal to zero, which represent the Block method;
2. When the conditions in the rectangle 1 are verified the lines below the then clause are executed. In the second rectangle is defined the base characteristics of the next

state which are the “name = #1”, the “status = freeze”, the “localr(Rname, initiated, 0)”, the “firstR(Rname, initiated, 0)”, the “create” that correspond to the clone_create and the “test_size” which is equal to the initial one;

3. In the third rectangle, is created a negotiation atom that represents the previous states. This means that even when the state machine reach the last state, it is possible to know all the previous states;
4. The two lines in the fourth rectangle are only to allow a better control over the states and are not important.

```
rule "#0 to #1"

  salience 100

  when

s : State ( name == "#0",
                  status == State.START,
                  cloneCreate == true,
                  count == 0)


    1

  then

nextState.setName("#1");
      nextState.setStatus(State.FREEZE);
      nextState.setLocalRHashMap(s.getLocalRHashMap());
      nextState.setFirstRHashMap(s.getFirstRHashMap());
      nextState.createCreate(s.getMsgRname(), s.getMsgType());
      nextState.setTestSize(s.getTestSize());


    2

nextState.createAtom(s.getName().s.getStatus());


    3

nextState.setProposalsResultHashMap(s.getProposalsResultHashMap());
      nextState.setNewCheckRequired(true);


    4

  end
```

Figure 5-6 - Negotiation rule example

5.2.3. Step 3 - Knowledge

Other important step in the framework is the knowledge process that is controlled by the System Controller in form of rules and is managed with the Drools tool, also as in the negotiation flow. The knowledge management process is illustrated in the Figure 5-7 and basically it is characterized by save the information about the Trigger Agents decisions to a certain proposal, into the knowledgebase, in order to be used in future negotiations. Explaining more detailed the Figure 5-7, when a Trigger Agent makes a proposal, he need to fulfill the questionnaire, which contain various points that are important to their system that will be modified. When System Controller receives this information it will run Drools in order to get some precious information that can be helpful for Trigger Agents that are participating in the negotiation.

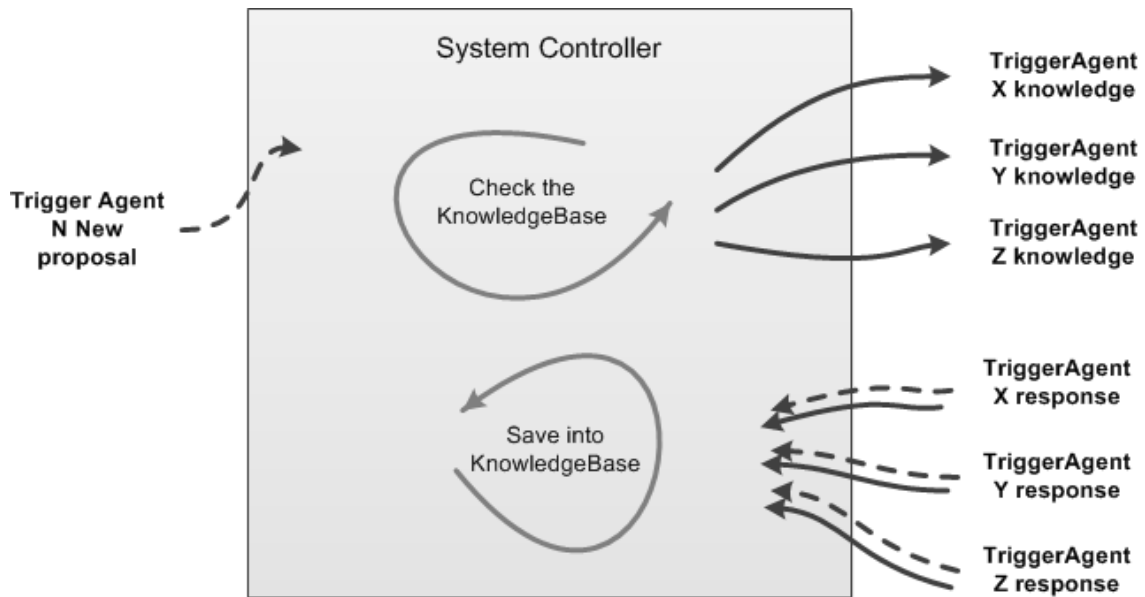


Figure 5-7 - System Controller knowledge management process

There are two types of knowledge that System Controller will look for. The first type is the qualification that each Trigger Agent makes to each point in the questionnaire made by the Trigger Agent that made the proposal. For example, if Trigger Agent N makes a proposal that contains a point “Less costs = true”, the participants in the moment of the decision can choose if the point “Less costs = true” is a positive or negative point. As System Controller will save this information, the next time that some Trigger Agent put the point “Less costs = true” in his questionnaire, System Controller will help the participants and will provide the last choice that they did, helping them with a possible qualification for that point. This type of rules can be viewed in the Figure 5-8.

```
rule "Check Performance"

  when

      issue : com.systemcontroller.negotiation.Issue (      description = "More performance",
                                                            answer = "true",
                                                            participant = "TriggerAgent-2" )

  then

      issue.setResult("positive");

  end
```

Figure 5-8 - Knowledge results rules example

The second type of saved knowledge by the System Controller is a bit more complex because each rule has much more information than the previous type of rules. This type of knowledge will save the decision of each Trigger Agent mapped with the points in the proposal questionnaire. This knowledge will help System Controller to determine a percentage of certainty that a Trigger Agent has to accept or reject a proposal. This percentage can be

calculated due to the fact that this type of rules can be repeated, which means that if System Controller has some rules for the same characteristics and the same Trigger Agent, the number of accepts and rejects will produce a percentage that will be the certainty of the decision. An example rule of this knowledge can be viewed in the Figure 5-9.

```
rule "example"

    when

        $map : java.util.HashMap (
            this["Characteristic1"] = "true",
            this["Characteristic2"] = "true",
            this["Characteristic3"] = "false",
            this["Characteristic4"] = "true",
            this["Agent"] = "TriggerAgent-2",
            this["decision"] = "accept" )

    then

        resBean.addAccept();

    end
```

Figure 5-9 - Knowledge decision rules example

In the Figure 5-9 example, when the System Controller receives a proposal with the illustrated characteristics, it will count the number of accepts and rejects and the result percentage will be sent to the TriggerAgent-2 in order to help him to make a decision. The suggestion that is sent to the destination Trigger Agent is calculated with the tree pre-defined rules that are illustrated in the Figure 5-10. The suggestion level will depend on the “resultPercentage” explained before and with the number of answered questions.

```
rule "Calculate Result (NOTENOUGHDATA)"
    when
        result : com.systemcontroller.negotiation.Result ( resultPercentage < 50 )
    then
        result.setFinalResult(com.systemcontroller.negotiation.Result.NOTENOUGHDATA);
    end

rule "Calculate Result (ENOUGHDATAREJECT)"
    when
        result : com.systemcontroller.negotiation.Result ( resultPercentage >= 50
                                                            numAnsweredPositives < numAnswered - 1 )
    then
        result.setFinalResult(com.systemcontroller.negotiation.Result.ENOUGHDATAREJECT);
    end

rule "Calculate Result (ENOUGHDATAACCEPT)"
    when
        result : com.systemcontroller.negotiation.Result ( resultPercentage >= 50
                                                            numAnsweredPositives >= numAnswered - 1 )
    then
        result.setFinalResult(com.systemcontroller.negotiation.Result.ENOUGHDATAACCEPT);
    end
```

Figure 5-10 - Knowledge suggestion rules example

6. TESTING AND HYPOTHESIS VALIDATION

Some definitions about testing can be found over the documentation available about tests. According to (Tretmans 2001) testing is the process of trying to find errors in a system implementation by means of experimentation. To (ISTQB 2011), a common perception of testing is that it only consists of running tests, i.e., executing the software, depending on the test type, testing can mean cause as many failures as possible so that defects in the software are identified and can be fixed or can mean the confirmation that the system works as expected. Basically these two definitions about testing will mean the same and the both definitions agree that the main goal 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, which means that it can only show the presence of errors, not their absence (Tretmans 2001).

Testing can have some objectives, like finding defects, gaining confidence about the level of quality, providing information for decision-making and preventing defects. Different viewpoints in testing take different objectives into account. For example, in development testing (e.g., component, integration and system testing), the main objective may be to cause as many failures as possible as described above. In acceptance testing, the main objective may be to confirm that the system works as expected, also as described above (ISTQB 2011).

In the next sub-sections it will be presented some testing methodologies that are available for software testing. Through these methodologies it will be chosen one that will be the best approach to apply to this particular proof-of-concept implementation. After some tests formalization it will be presented the acquired results based on the differences between different approaches of negotiating interoperability in the SEI environment. To conclude this section a scientific context validation will be presented.

6.1. Testing Methodologies

Many testing methodologies are well known and are available to use in the software projects. Many of these methodologies are abstract concepts like the white-box testing, the black-box testing, the grey-box testing, the unit testing, the conformance testing and so on. Particularly, in software testing, the methodology that distinguishes is the functional and the structural testing (ISTQB 2011), (White 1987), (Myers et al. 2004).

Structural testing, also referred to as white-box testing, is based on the internal structure of a computer program. The main goal of this methodology is to exercise thoroughly the program code, e.g., by executing each statement at least once, or by trying to execute all paths through the program code taking into account decisions, branches, loops, etc. These tests are

derived from the program code, since that the code is essential to perform a good structural test and for that, structural testing is most used in the early stages of program development (Tretmans 2001).

With functional testing the emphasis is on testing the externally observed functionality of a program based on its specification. Functional testing is also called black-box testing, where a system is treated as a black box, whose functionality is checked by observing it, i.e., no reference is made to the internal structure of the program. The aim of this methodology is to determine whether the right (with respect to the specification) product has been built. These tests are derived from the specification and consequently, the most important prerequisite is a precise, complete and clear specification. Functional testing is usually concentrated in the later stages of program development (Tretmans 2001).

Conformance testing is a kind of functional testing where an implementation of a protocol entity is solely tested for conformance with respect to the requirements given in its specification. The idea is that only systems with correctly implemented protocols can communicate successfully with peer entities. In practical, conformance testing tests the internal structure of an entity that usually is not accessible to the tester, which means that the computer system in which the entity under test is located need not be accessible, e.g., when testing is performed by an independent, accredited test laboratory, that has no access to the implementation details of an implementation (Tretmans 2001).

In the next subsections it will be presented two standards that implement the previous presented methods. These standards were defined and revised throughout the years based on the expertise of using them and their practical results.

6.1.1. iSurf Functional and Non-Functional Evaluation Methodology

The iSURF European Project is integrated in the European Community's Seventh Framework Programme and develops an intelligent collaborative supply chain planning network that realizes a knowledge-oriented inter-enterprise collaboration environment in which distributed intelligence of multiple trading partners are exploited in the planning and fulfillment of customer demand in the supply chain. The project provides interoperability solutions for achieving the semantic reconciliation of the planning and forecasting business documents exchanged between the companies according to different standards (Anon 2010).

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 25040) of the Software product Quality Requirements and Evaluation (SQuaRE) series of standards. This standard 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 will be the repeatability, reproducibility, impartiality and objectivity of all process (i-Surf 2009).

The principal standard steps for iSurf evaluation strategy are the following: prepare; establish; specify; design; execute; report. The iSurf project also defines in detail the procedures used to generate the evaluation criteria that were applied for the functional and non-functional characteristics, which are: functionality, reliability, usability, efficiency, maintainability and portability. The project identifies the following techniques that are applied for evaluation of the iSurf components and architecture: functional tests; unit tests; fault tolerance analysis; user interface analysis; execution time measurements; inspection of documentation and analysis of software installation procedures (i-Surf 2009).

This techniques and the iSurf evaluation criteria are modularized 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) and it identifies the evidence it needs. It also defines the elementary evaluation procedure and the format for reporting the measurements resulting from the application of the technique (i-Surf 2009).

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 (i-Surf 2009).

The modules described define a specific aspect of a software quality characteristic that is being measured. It specifies 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 (i-Surf 2009):

- It provides formal information about the evaluation module and gives an introduction to the evaluation technique described in the evaluation module;
- Defines the scope of applicability of the evaluation module;
- Specifies the input products required for the evaluation and defines the data to be collected and measures to be calculated;
- 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 (i-Surf 2009):

- 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 – Test Notation Standard

Test and Test Control Notation (TTCN-3) is the evolution of Tree and Tabular Combined Notation (TTCN-2) and is a standardized testing technology developed and maintained by the European Telecommunication Standards Institute (ETSI) and specifically designed for testing and certification. TTCN-3 shows a lot of new capabilities comparing to the old TTCN-2, but since that for the validation of this proof-of-concept is enough a basic and simple test methodology, the main focus will be over the TTCN-2 technology (TTCN-3).

TTCN is a flexible and powerful language applicable to the specification of all types of reactive system tests over a variety of communication interfaces. Typical areas of application are protocol testing (including mobile and Internet protocols), service testing (including supplementary services), module testing, testing of Common Object Request Broker Architecture (CORBA) based platforms, API testing, etc. TTCN is not restricted to conformance testing and can be used for many other kinds of testing including interoperability, robustness, regression, system and integration testing (ETSI ES 201 873-1 2012).

In TTCN, the tests behaviors are defined by a sequence of events that are represented as trees, containing branches of actions based on evaluation of the system output after one or more executed events. Each event has its own respective level of indentation and can be declared in two different types: action or question. Actions are preceded by an exclamation point before its description and are 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. To complete a TTCN test table, a verdict must be deliberate, which can be “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 (ETSI ES 201 873-1 2012).

In the Table 6-1 is described a simplified example of a phone call establishment evaluation. As shown in the Table 6-1 different verdicts result after a series of actions and evaluations. Below is explained textually the content of the table (TTCN-3).

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:		
Behavior	Constraints	Verdict
! Pick up headphone ? Dialing tone ! Dial number ? Calling tone ? Connected line ! Hung up headphone OTHERWISE ? Busy tone ! Hung up headphone OTHERWISE ? Dialing tone absent		Success Failure Inconclusive Failure Failure

- The user picks up the headphone;
- Tests if the dialing tone is present;
- If the dialing tone is present, then the user must dial the other phone's number. Otherwise, if the dialing tone is absent, the verdict is a “Failure” of the possibility of establishing a phone call;
- If there is a calling tone after dialing the number, the user may test if the line is in fact connected;
- 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;

- If the dialing 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;
- If none of the tones corresponds to calling or busy, then the verdict is set as “Failure” on establishing a phone call.

6.1.3. Adopted Test Methodology

The proof-of-concept developed in this thesis is not like a commercial product that is not supposed to be flawless and should be a complete solution. Unlike that, this proof-of-concept should be a working proof of feasibility of a full solution. So, this means that is not necessary a complex methodology for testing this proof-of-concept, since that is too expensive for such kind of implementation. With this and analyzing the two presented methodologies, it is clearly that the iSurf is much more complex than the TTCN, which means that it will be used a mix of these two methodologies to validate the proof-of-concept implementation.

Based on these two methodologies, a series of functional test cases and unit tests described by TTCN tables will be designed and applied to the various units of the implementation steps. Besides this, non-functional tests such as reliability, efficiency and portability were also addressed. The results taken by the execution of these tests were published in the sub-section 6.3.

6.2. Requirements and Functionalities

The requirements and the functionalities of the system that are defined during the design of the system are presented in this sub-section. All the requirements and functionalities presented here are responsible to define all the capabilities of the developed proof-of-concept and in order to evaluate the extent of the proof-of-concept implementation, it will be made a mapping between the requirements and functionalities of the system and what is implemented.

▪ Requirements

▫ **Each application should have a GUI**

The developed implementation has a GUI for each application, allowing a full interaction between the application and the user both in the System Controller as in the Trigger Agent;

▫ **Both System Controller and Trigger Agent should be able to run on different machines, i.e., with different IP addresses**

Since the developed system is a distributed system, controlled by JADE, it allows that all connected Trigger Agents were in different networks of the System Controller;

- **Every connected Trigger Agent should be able to start a negotiation round**

Since that exists two or more connected Trigger Agents to System Controller, every Trigger Agent can start a negotiation round when it detects a change in its interoperability list;

- **The Trigger Agent application should detect the changes on the interoperability list of his system and initiate a trigger to the System Controller indicating what has changed**

As explained earlier, this requirement is out of scope of this thesis, thus, it is used a simple button to simulate the changes in the interoperability list;

- **The System Controller should have a rule engine system in order to process all the negotiation functions**

The implemented System Controller uses the Drools rule engine, which allows the application to control the flows of the negotiations and control the knowledge of the system;

- **The communications between the applications should be made through the agents**

The utilization of JADE to implement a MAS allows the perfect communication between the applications through the agents;

- **The System Controller should support at least two Trigger Agents connected to him**

The System Controller allows much more Trigger Agents connected to him due to the utilization of JADE. The limit of connections depends on the JADE;

- **Each Trigger Agent should be able to connect only to one System Controller**

The Trigger Agent application only allows one connection, controlled in the application;

- **Functionalities**

- **The architecture of the system should be implemented using agents technology**

In the execution steps of this proof-of-concept it is possible to see that the architecture developed consists in a central agents that accepts connections from various client agents.

- **The system should have a mechanism based on a rules engine that control the knowledge of the system, helping clients in their decisions**

Also on the implementation steps, it is possible to see that the System Controller keeps the knowledge in a knowledge base, using Drools.

- **The system must be capable of control the negotiation flows through the rules predefined in a rules engine technology.**

The system controls the various types of negotiation methods through the Drools rules engine.

6.3. Testing

In order to address the functional and non-functional testing of the implemented proof-of-concept, in this sub-section it will be demonstrated all the steps presented in the sub-section 5.2 followed by the explanation that how this prototype works. As in the sub-section 5.2, the following sub-section will cover the four steps of the implementation, showing all the results of the tests made to each step. The functional tests will be presented through the Table 6-1 and the non-functional tests will be executed in order to test if the software performs the required functions under a given conditions for a given time interval. The last point that is important to test is a comparison between the performance of this proof-of-concept compared to the old method, i.e., without negotiation.

6.3.1. Step 0 – Environment setup

The environment setup can be tested with an example showed before, like in the Figure 5-3, where the System Controller initiate and then, four Trigger Agents connect to him. This is a basic test but it will test if the environment is performing the connections and disconnections correctly.

So, when the System Controller application starts, the user have the GUI showed in the Figure 6-1 available. As shown in the Figure 6-1, the status is "Offline" and only the Start button is enabled. When the user clicks in the Start button, the system will initialize, starting the JADE platform and the Drools rule engine. This operation may last a few seconds to be completed. When System Controller is initiated successfully, a message is shown and the other buttons became enabled, as shown in the Figure 6-2. In this process of initialization, System Controller beyond of starting JADE and Drools, it creates the SystemController agent and registers it in the JADE platform in order to be able to communicate with the connected Trigger Agents.

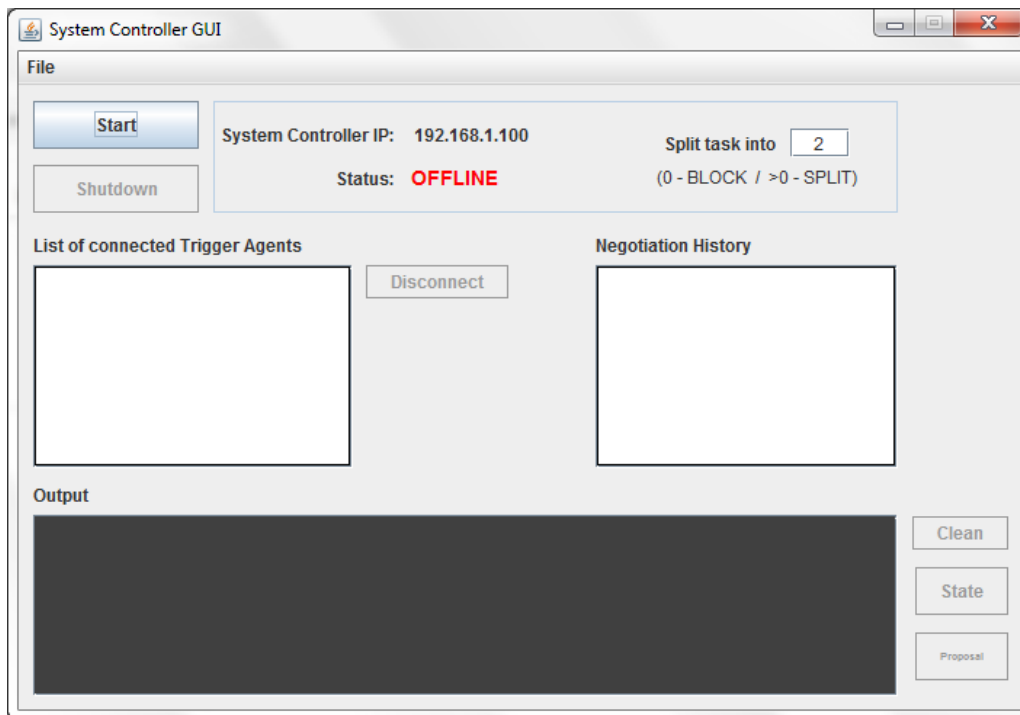


Figure 6-1 - GUI of System Controller when the application starts

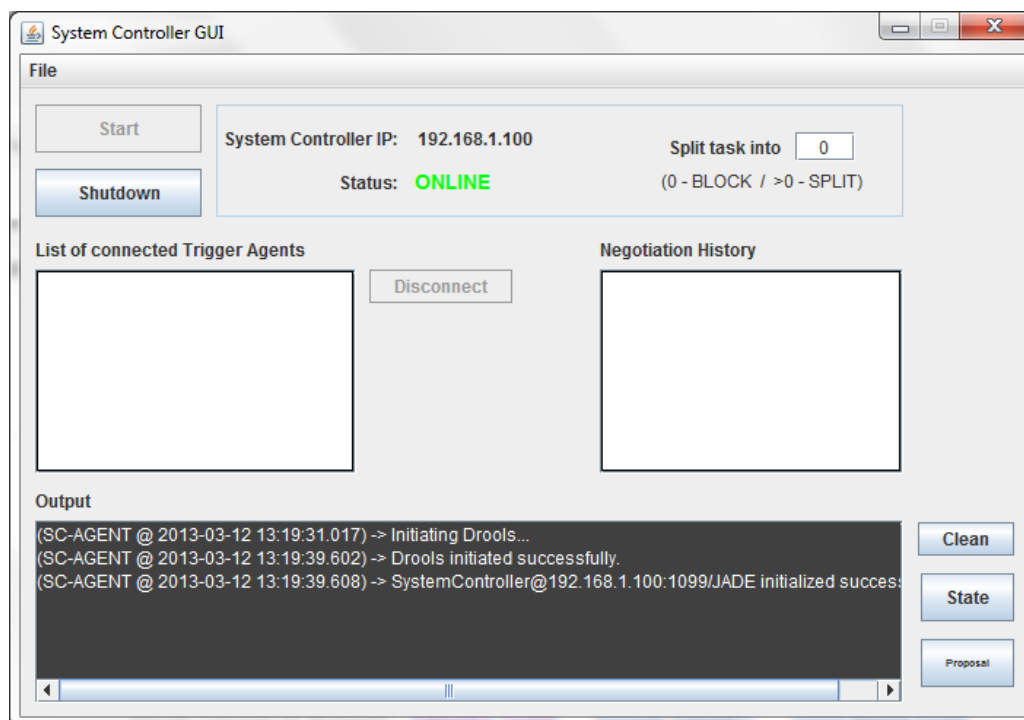


Figure 6-2 - System Controller initiated successfully

In the Trigger Agent side, when the user connects the Trigger Agent, the agent is registered in the JADE platform and the System Controller receives a notification of this event in order to add the connected agent to his list. The Figure 6-3 shows the Trigger Agent already connected to System Controller by its Internet Protocol (IP) address.

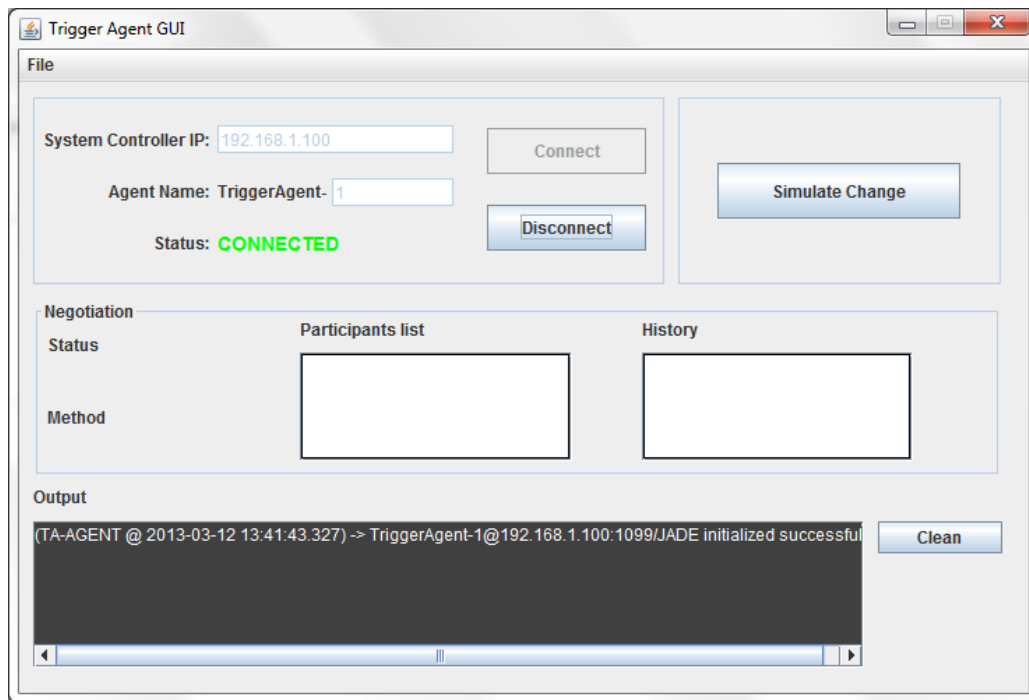


Figure 6-3 - Trigger Agent connected to System Controller

After the initialization, System Controller is now able to receive connections from the Trigger Agents and when it receives the second connection, Trigger Agents can start a negotiation round. In the Figure 6-4 it is possible to view this case, which shows in the last line of output field that the initial state was created, meaning that it can receive an interoperability change notification in order to start a new negotiation round.

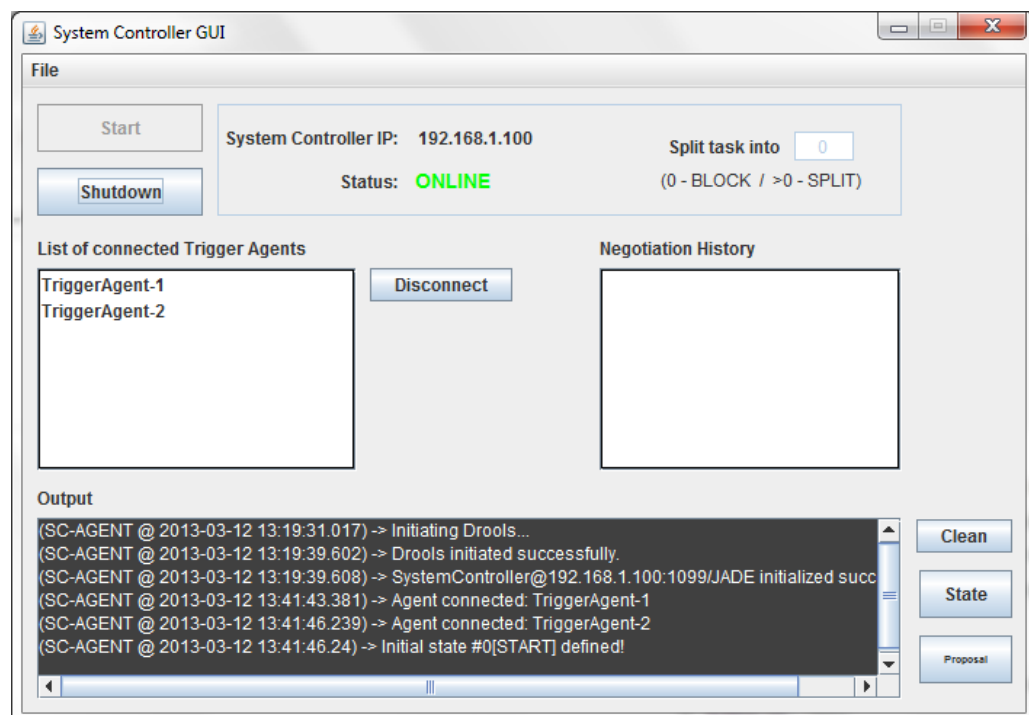


Figure 6-4 - System Controller with two Trigger Agents connected

Now the results of the functional tests will be presented, covering the initialization of the System Controller, the Trigger Agent connections and the Trigger Agent disconnections. Starting with the System Controller initialization, in the Table 6-2 is shown the results of this test, using the method explaining in the sub-section 6.1.3.

Table 6-2 - System Controller initialization functional test

Test Case		
Test Case: System Controller connection Group: Purpose: Check if the System Controller starts successfully Comments:		
Behavior	Constraints	Verdict
! Start System Controller		
? JADE starts		Success
? Drools starts		Success
? SystemController agent is registered in JADE		Success
OTHERWISE		Success
OTHERWISE		Success
OTHERWISE		Success

The second functional test case is represented in the Table 6-3 and correspond to the connection of a Trigger Agent, more precisely, it consists in the connection of the second Trigger Agents, which allow to test, not only the agent connection, but also the creation of the initial state which will allow that the negotiation round could start.

Table 6-3 - Second Trigger Agent connection functional test

Test Case		
Test Case: System Controller receives the second Trigger Agent connection Group: Purpose: Check if the Trigger Agent is successfully connected and the initial state is created Comments:		
Behavior	Constraints	Verdict
! Connect Trigger Agent		
? TriggerAgent agent is correctly initiated in JADE		Success
? System Controller receives a CONNECT a notification		Success
? SystemController creates the initial state		Success
OTHERWISE		Success
OTHERWISE		Success
OTHERWISE		Success

The last functional test in this step is a disconnection from the System Controller of an connected Trigger Agent and is represented in the Table 6-4.

Table 6-4 - System Controller disconnects an Trigger Agent functional test

Test Case		
Test Case:	System Controller disconnects an Trigger Agent	
Group:		
Purpose:	Check if the Trigger Agent is successfully disconnected	
Comments:		
Behavior	Constraints	Verdict
! Disconnect Trigger Agent from System Controller		Success
? TriggerAgent agent is correctly disconnected from JADE		Success
? TriggerAgent agent is correctly disconnected from System Controller		Success
OTHERWISE		Success
OTHERWISE		Success

6.3.2. Step 1 – Negotiation

In this step it will be tested the negotiation between the enterprises through their Trigger Agents. For this test will be used the Block negotiation method explained in the sub-section 5.1.1, in particular, it will be used the example of the Figure 5-1 with four Trigger Agents in the negotiation round.

First of all, when all the four Trigger Agents are connected to System Controller, the TriggerAgent-C will simulate a change in his interoperability list in order to start the negotiation round. When the TriggerAgent-C makes the simulation it will send the Proposal 1 to System Controller and all the other participants will receive the notification of a new proposal, as shown in the Figure 6-5, Figure 6-6 and Figure 6-7.

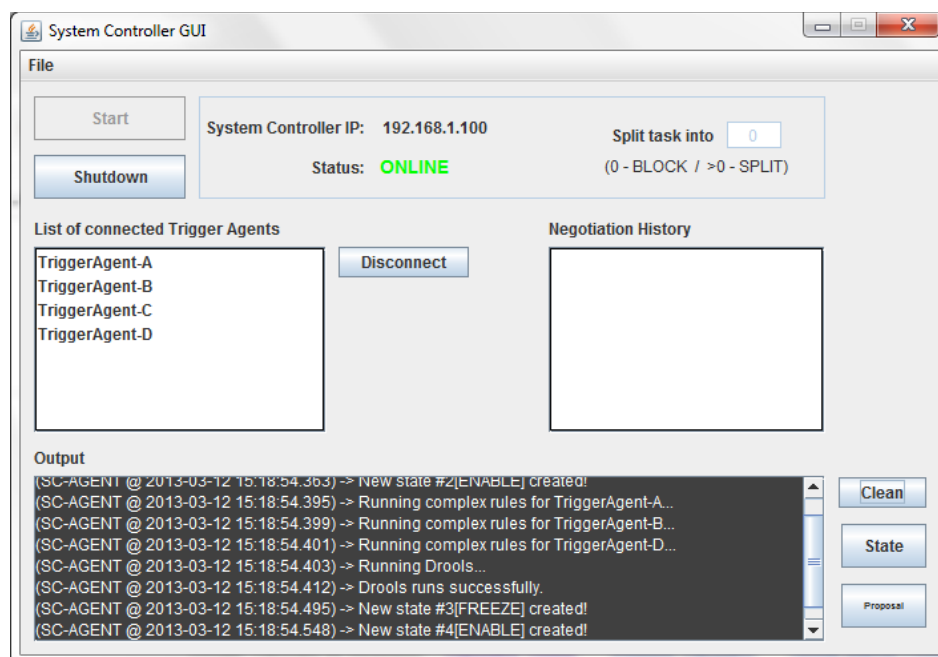


Figure 6-5 - System Controller receives the change notification from TriggerAgent-C

As illustrated in the Figure 6-5 the system controller changes its state to “#4[ENABLE]” which corresponds to the state that waits for the decisions from the other participants. In this state, the negotiation participants may create new proposals.

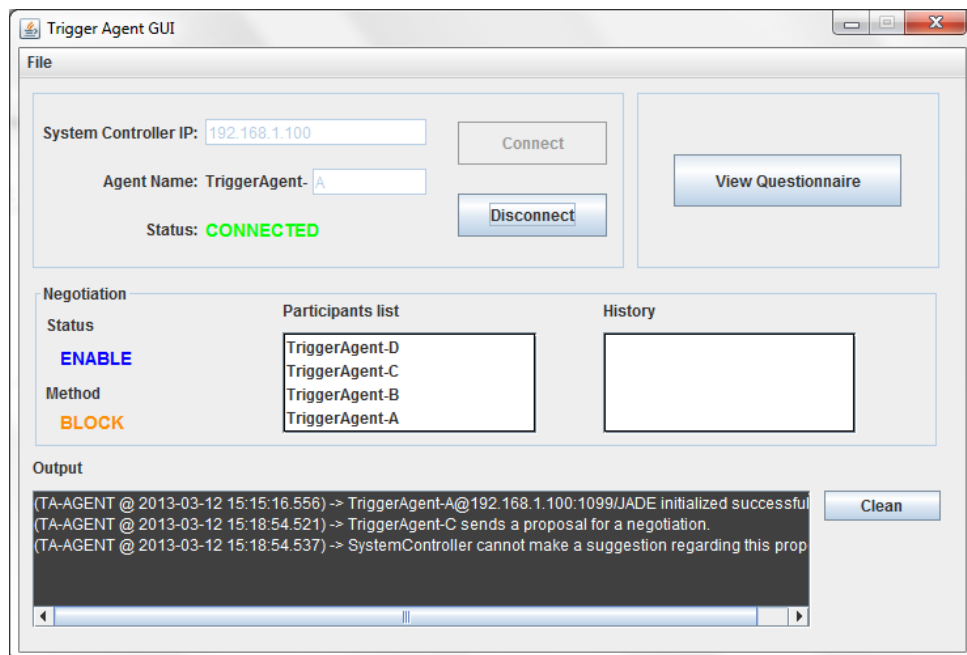


Figure 6-6 - TriggerAgent-A receives the proposal made by the TriggerAgent-C

In the Figure 6-6, TriggerAgent-A receives the proposal made by the TriggerAgent-C and with the GUI is updated to show the user some important negotiation information, like the negotiation status, the current negotiation method, and the negotiation participants. By clicking in the View Questionnaire button, a new window opens with the details of the proposal made by the TriggerAgent-C which is illustrated in the Figure 6-7.

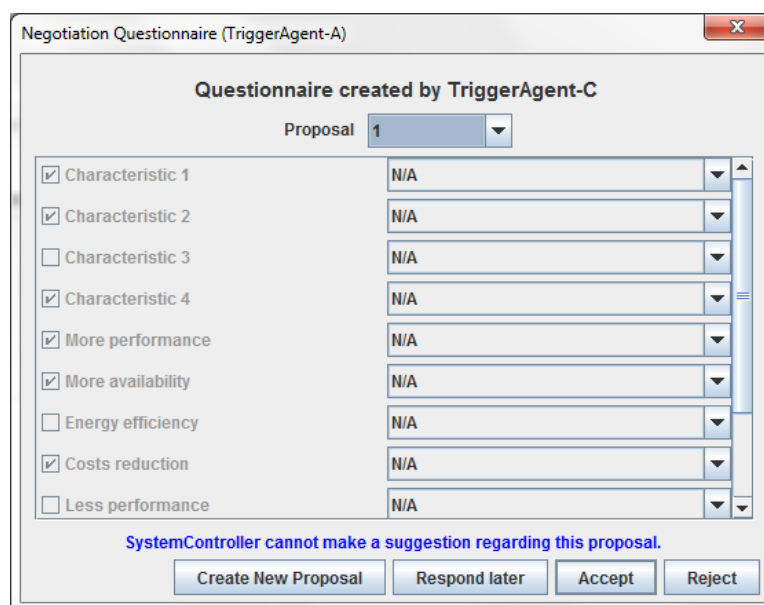


Figure 6-7 - Proposal questionnaire created by TriggerAgent-C

As shown in the Figure 6-7, TriggerAgent-A have some possibilities to respond to this proposal, but first, if there is more than one proposal to answer, they can be chosen in the proposal combo box. In this case, following the example in the Figure 5-1, TriggerAgent-A will reject this proposal and the System Controller will receive this decision as shown in the Figure 6-8.

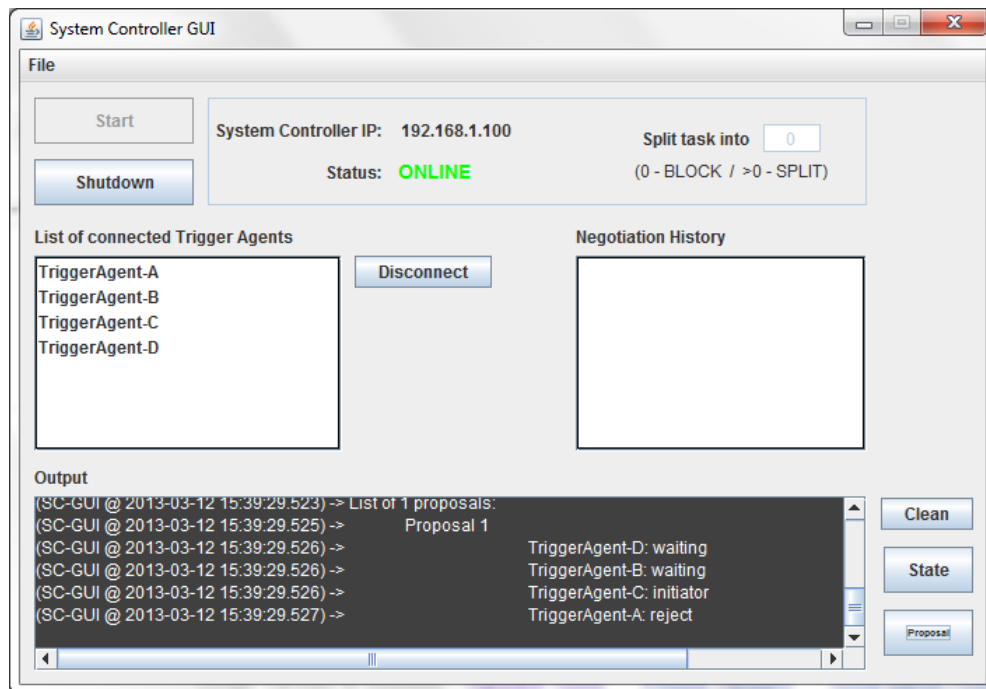


Figure 6-8 - System Controller receives a reject decision from TriggerAgent-A for Proposal 1

In the Figure 6-8 it is possible to see that the TriggerAgent-A rejected the Proposal 1 as explained before and the TriggerAgent-B and D have not yet answered to the Proposal 1.

This negotiation process will continue, regarding the example in the Figure 5-1. When all proposals are answered, System Controller will automatically change the state and it will evaluate the proposals in order to end up with a winner proposal, as shown in the Figure 6-9.

The Figure 6-9 shows the System Controller after receiving all proposals decisions from all negotiation participants and as illustrated, the winner proposal was the Proposal 2 as expected. The negotiation round reach the final state and with that, the System Controller prepare himself for a new negotiation round, restarting the states and put the previous negotiation available to be viewed in the Negotiation History list.

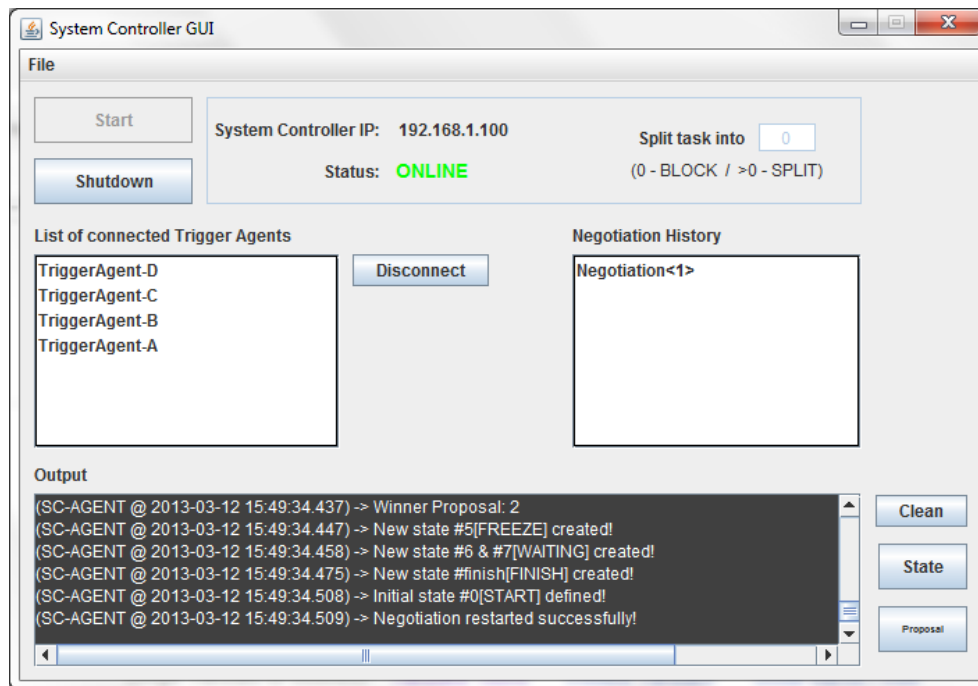


Figure 6-9 - System Controller after receiving all proposal decisions

In the Table 6-5 is presented the results of the functional tests done during a negotiation round with the some Trigger Agents involved. This test is generic, which make it a good test to apply over the Block or Split methods.

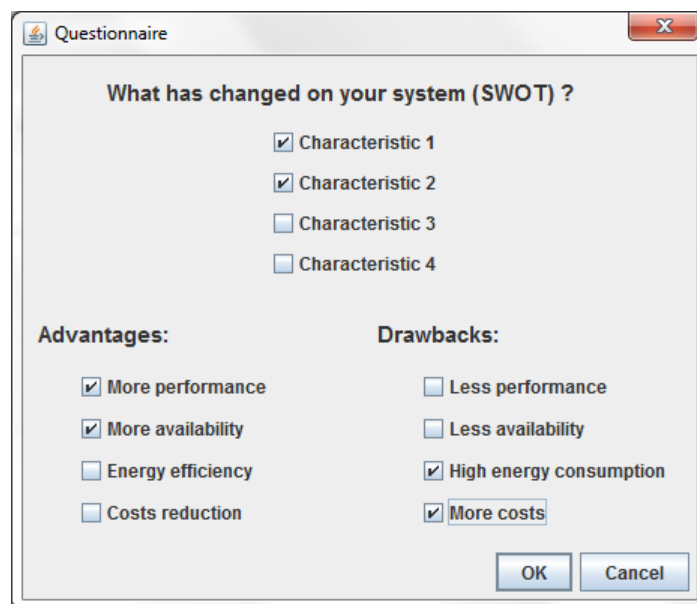
Table 6-5 - Negotiation flow functional test

Test Case		
Test Case: Negotiation flow Group: Purpose: Check if the negotiation flow is correct Comments: For the Block method is only necessary one proposal but for the Split method is necessary a number of proposals equal or greater than the split number defined in the System Controller		
Behavior	Constraints	Verdict
! Trigger Agent simulates a change in his interoperability list		
? System Controller receives the notification		Success
! System Controller notify all negotiation participants		
? Participants receive the notification		Success
! Participants respond to the proposal		
? System Controller receives all decisions		Success
! System Controller choose the winner(s) proposal		
OTHERWISE		Success
OTHERWISE		Success
OTHERWISE		Success

6.3.3. Step 3 – Knowledge

In the previous sub-section the main focus was over the negotiation flow and not over the knowledge, but as the knowledge is related with the negotiation, in the some illustrations was possible to see some points of knowledge that will be presented in this sub-section. As in the previous tests, this test also will be done regarding an example that in this case will be the example shown in the Figure 5-7, that once more it will be the System Controller with four Trigger Agents connected.

For this test, TriggerAgent-N from the Figure 5-7 will start to create a proposal that contain the point illustrated in the Figure 6-10.



Questionnaire

What has changed on your system (SWOT) ?

☒ Characteristic 1
☒ Characteristic 2
☐ Characteristic 3
☐ Characteristic 4

Advantages:

☒ More performance
☒ More availability
☐ Energy efficiency
☐ Costs reduction

Drawbacks:

☐ Less performance
☐ Less availability
☒ High energy consumption
☒ More costs

OK Cancel

Figure 6-10 - Proposal creation from TriggerAgent-N

When TriggerAgent-X receives this proposal, it will accept with the results defined for each point in the proposal that are represented in the Figure 6-11. In the Figure 6-11 it is possible to view that the System Controller failed to give an suggestion about the decision of the proposal because this proposal was the first proposal running in the System Controller, meaning that when the System Controller searched in the rules for a rule that can help in this proposal, there was no rules in the knowledgebase.

When the System Controller receives the decisions from the negotiation participants, it will save the knowledge into the knowledgebase, as explained in the sub-section 5.2.3.

Negotiation Questionnaire (TriggerAgent-X)

Questionnaire created by TriggerAgent-N

Proposal 1

<input checked="" type="checkbox"/> Characteristic 1	positive
<input checked="" type="checkbox"/> Characteristic 2	negative
<input type="checkbox"/> Characteristic 3	N/A
<input type="checkbox"/> Characteristic 4	N/A
<input checked="" type="checkbox"/> More performance	positive
<input checked="" type="checkbox"/> More availability	positive
<input type="checkbox"/> Energy efficiency	negative
<input type="checkbox"/> Costs reduction	negative
<input type="checkbox"/> Less performance	positive

SystemController cannot make a suggestion regarding this proposal.

Create New Proposal Respond later Accept Reject

Figure 6-11 - TriggerAgent-X response to the proposal made by TriggerAgent-N

When the previous negotiation round reach the end and the TriggerAgent-N starts other negotiation round by simulating a change in his interoperability list, System Controller will check once more if there are any rules to help the negotiation participants. In this case TriggerAgent-N will create a proposal equal to the one in the Figure 6-10 causing there any some valid knowledge to help the TriggerAgent-X in this negotiation round. So, when System Controller notifies the negotiation participants, TriggerAgent-X will receive the proposal as shown in the Figure 6-12.

Negotiation Questionnaire (TriggerAgent-X)

Questionnaire created by TriggerAgent-N

Proposal 1

<input checked="" type="checkbox"/> Characteristic 1	positive
<input checked="" type="checkbox"/> Characteristic 2	negative
<input type="checkbox"/> Characteristic 3	N/A
<input type="checkbox"/> Characteristic 4	N/A
<input checked="" type="checkbox"/> More performance	positive
<input checked="" type="checkbox"/> More availability	positive
<input type="checkbox"/> Energy efficiency	negative
<input type="checkbox"/> Costs reduction	negative
<input type="checkbox"/> Less performance	positive

SystemController suggests that we should ACCEPT this proposal with 100.0% certainty.

Create New Proposal Respond later Accept Reject

Figure 6-12 - Proposal received by TriggerAgent-X with a suggestion by the System Controller

In the Figure 6-12 is shown that as TriggerAgent-X accepted the first proposal, in this proposal, the System Controller, not only, makes a suggestion to accept the proposal with one hundred percent certainty, but also, fulfill automatically the results for each point in the proposal, which can be very useful for the TriggerAgent-X.

This example only reaches the second proposal, but if the negotiation continues, System Controller will enrich his knowledgebase and the next suggestions will certainly be more consistent.

In the Table 6-6 is represented the results of the functional test for the knowledge process that consists basically in the example explained before, where in the first proposal the System Controller does not have any knowledge about it, but in the second one, it is possible to help the negotiation participants through the knowledge acquired in the first proposal.

Table 6-6 - System Controller knowledge process functional test

Test Case		
Test Case: System Controller knowledge process Group: Purpose: Check if the System Controller is control well the knowledge Comments: This test case starts with no previous negotiation rounds		
Behavior	Constraints	Verdict
! Trigger Agent creates a proposal		
? System Controller does not have any knowledge about it		Success
! Participants respond to the proposal		
! System Controller saves the knowledge		
! Trigger Agent creates another equal proposal		
? System Controller have some knowledge about it		Success
! System Controller sends the suggestion to the participants		
OTHERWISE		Success
OTHERWISE		Success

6.3.4. Performance comparison

In this sub-section it will presented a method to do some tests in order to determine the time that a client needs, to change his system to continue interoperable with the system that made the interoperability change. For that, it was created a web-service to accommodate some changes, this web-service was created in both Java and C#, but since both showed similar results, the focus it will be on the Java web-service.

The created web-service consists in one method called “carPaint” that takes for input a car and a date of delivery and returns the date that the car will be painted and ready for delivery. In order to reach the pretended results, it will be applied some changes in the previously created web-service that will consume some time that is different depending on the change complexity. The changes and the time consumed are listed below:

1. In the first change, only the port of the web server has change from 8080 to 8081. This change requires some changes in WSDL file and requires generating the client classes and will consume between thirty seconds and one minute.
2. The second change was made in the delivery date that now becomes to be validated in the server side. This change requires a validation in the date returned by the method and requires a rebuild. This will consume about one and a half minute.
3. In the third change was added a new argument to the previous method. A color variable to determine the color that the car will be painted. For that, it is necessary some changes in WSDL file, requires generating the client classes and requires a rebuild. This will consume about three minutes.
4. The fourth change was in the semantic side. Previously, the class named Car, that represents a car, can be interpreted differently by different people. So, it is necessary to develop an ontology in order to represent the Car equally for all clients that use this method. Developing an ontology includes some steps, like defining classes in the ontology, arranging the classes in a taxonomic (subclass-superclass) hierarchy, defining and describing allowed values for this slots and filling in the values for slots for instances. This change can be made in about three or four days.
5. The fifth change occurs when an entity wants to do a modification in the car ontology used in the previous example. For example, previously the ontology represents a toy car, whereas now, the ontology represents a real car. This change may take very long time to be matched, since the client needs to know exactly what the server are “talking”. The changes are the same as in the fourth point adding just one more task before, that is know exactly what has changed. These changes will take five or 6 days to do all the tasks.
6. The last change is the most complex, since it's a change in the process of an operation. In this case is a change in the painting process. The previously process of painting was done by spray and now the car is painted by submersion technique. This change seems to be a simple change, but looking more deeply, it reveals that this change has a lot of complications in the client side. These complications are the method that the client delivery the car, because now, the client have to delivery only the car body and not the whole car, the car body needs to resist painting temperature which in this case the temperature is very high, all that the submersion process may

do to the car body, for example, corrosion, deformation and the screw holes can be covered and finally, changes in the logistics, since it must be delivered only the car body and the rest of the car should be treated separately, which may be cause some problems with other supplier companies. This complex change will consume between fifteen to twenty days.

In the chart of the Figure 6-13 it is shown a curve that characterizes in a very good way the time spent in the client side in terms of the complexity of the change. This time represents the time that the client needs to change his system to continue interoperable with the system that make the change, in other words, represent the system “downtime”. The curve that is represented in the Figure 6-13 is an illustration based on the above six changes that were made to the previous created web-service, which means that the axis that refers to “Interoperability Complexity” is just a sensitivity representation of complexity, since this characteristic is not measurable in practical terms, meaning that the scale goes from 1 to 6 where, 1 is the low complexity and 6 is high complexity of interoperability, and the “downtime” axis represents the real time spent in the web-service modifications.

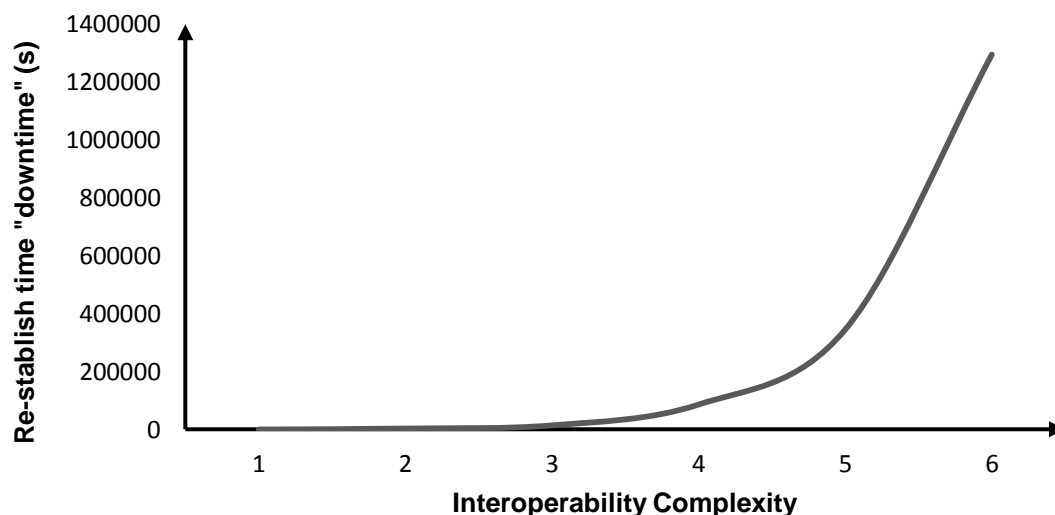


Figure 6-13 - Time spent in the system change vs. the complexity of the changes without negotiation

Regarding the chart in the Figure 6-13 that shows a representation of the required time to change the system relatively to the interoperability complexity of the change, when is introduced negotiation between entities allied to the systems knowledge, it is possible to change the interoperability curve drastically in a long term way, as illustrated in the chart of the Figure 6-14. This curve shows that the interoperability complexity grows over the time, but since the negotiation is allied to the knowledge, in some point of the time, the complexity tends to decrease. This characteristic is due to the system knowledge, which increases with the time. Analyzing more deeply the chart in the Figure 6-14 and since that this chart is just a sensitivity representation, the growing part of the curve is represented by the system learning which may be compared to the curve represented in the chart of the Figure 6-13. But, when the system

reach a point of knowledge that may allow it to take some decisions in the negotiations, the curve of the interoperability complexity tends to decrease, meaning that the system negotiations will be more efficient, requiring less downtime from the systems. The turning point vary from system to system, because depends on some variables, like, for example, the number of interoperability changes performed, the level of system learning and of course, the time elapsed.

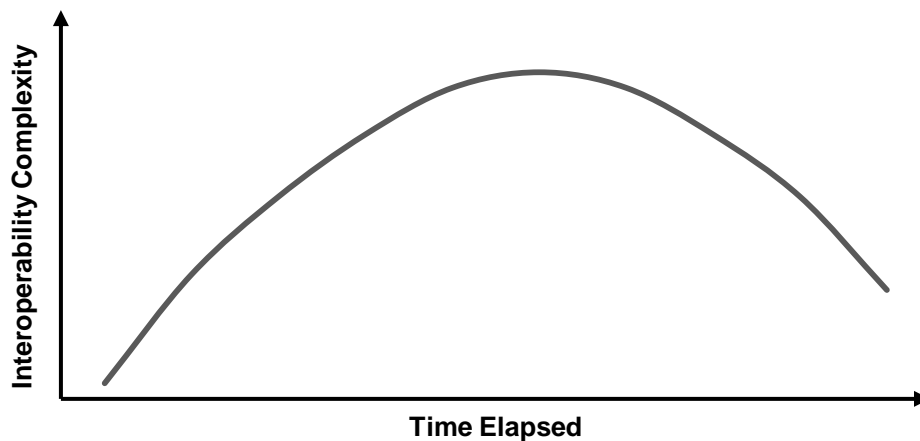


Figure 6-14 - Complexity of the system change over time with negotiation

Representing the chart of the Figure 6-13 in a different way, it is possible to characterize a line that grows over time as the interoperability complexity grows. This behavior is explained due to the inexistence of some mechanism that minimizes the system “downtime” when a harmonization break occurs. This line is illustrated in the Figure 6-15 over the curve that was already illustrated in the Figure 6-14. As it is possible to retain in this chart, when the system with negotiation reaches the point when the negotiation allied to the knowledge allows it to make decisions, it is very clear that the interoperability problems becomes much more easy to deal and much more easy to reestablish the harmonization breaking.

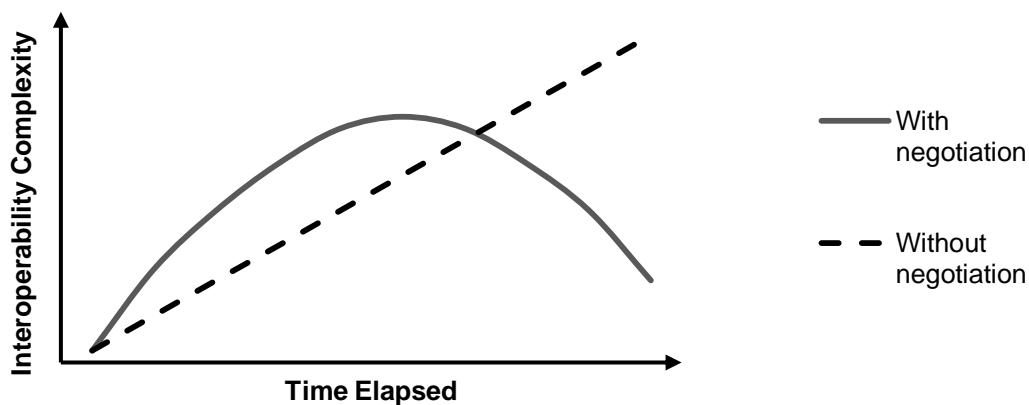


Figure 6-15 - Interoperability complexity over time with and without negotiation

Now, picking up the curve in the Figure 6-14, but this time representing it in terms of the time that the system needs to be reestablished as the interoperability complexity grows, it is possible to estimate, through the performed tests in the above web-service and comparing to the curve in the chart of the Figure 6-13, that the system downtime will be greater, for low interoperability complexity but for high levels of interoperability complexity the negotiation allied to the knowledge becomes an asset, making the systems capable of reduce their downtime. Observing the two curves in the Figure 6-16 that illustrates, once more in a sensitivity way, the above analysis, it is possible to conclude that a system with negotiation will show higher “downtime” for low levels of interoperability complexity than a system without negotiation, but in the important region of the chart (for high values of interoperability), the “downtime” will be smaller than the system without negotiation. The advantage of the system without negotiation shown for low interoperability complexity levels is due to, mainly, the time spent in the negotiation process which will pay off for higher interoperability complexity.

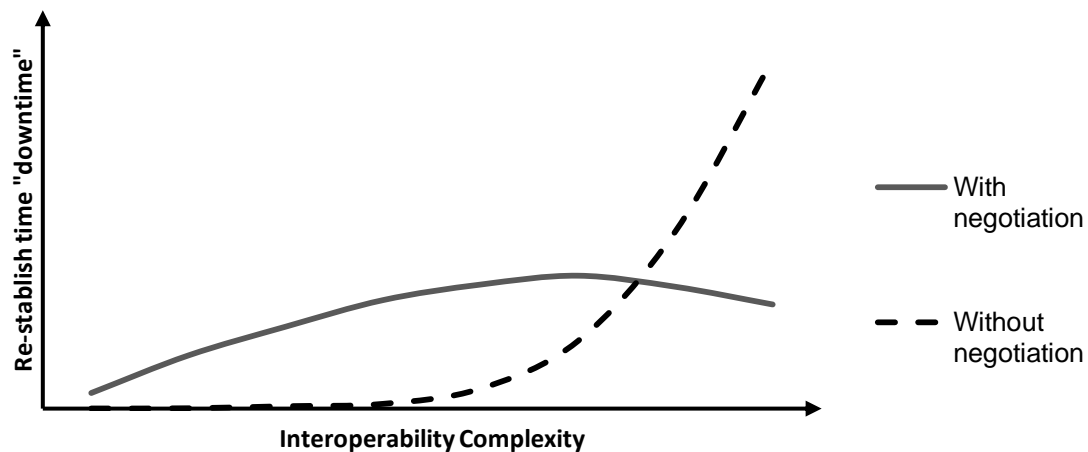


Figure 6-16 - Systems re-establishment time vs. interoperability complexity, with and without negotiation

With the above analysis over the presented sensitivity charts and the real test performed over the web-service, it is possible to conclude that for low levels of interoperability allied to immature systems, the approach without negotiation will show smaller system “downtimes” because when a system made a simple change, the interoperable systems will adapt themselves with a simple modification, regardless the system knowledge. So, the real problem is when a mature system makes a higher complexity change, which, without negotiation, will cause a huge downtime in the entire system, but if the entire system were capable of perform some interoperability negotiations and additionally capable of learning with those negotiation, this kind of system changes becomes much more simplest, reducing drastically the time that the systems need to be inoperable.

6.4. Hypothesis Validation

In the sub-section 1.4 was defined the hypothesis of this dissertation as well as the objectives of this work. After the conclusion of the proof-of-concept implementation and with all tests executed, it is possible to say that the question made in the hypothesis was successfully achieved during this dissertation. The developed framework fulfilled the main objectives and it was able to control a negotiation interoperability sustainable environment, making a much more stable and prepared for the harmonization breaks environment due to the negotiation and the knowledge offered by the framework.

6.5. Scientific Validation

In order to validate the proposed work, two scientific publications were made. The first one was published in the 17th IEEE International Conference on Computer Supported Cooperative Work in Design (CSCWD 2013), from 27th to 29th of June 2013 in Whistler – Canada. The second publication was made in International IFIP Working Conference On Enterprise Interoperability (IWEI 2013), from 27th to 28th of March 2013 in Enschede – Netherlands. The descriptions of the two papers are listed below:

- Santos, T., Coutinho, C., Jardim-Goncalves R. and Cretan, A., “Negotiation Environment for Enterprise Interoperability Sustainability”, accepted in: 17th IEEE International Conference on Computer Supported Cooperative Work in Design (CSCWD 2013). Jun 27-29, Whistler, Canada, 2013.
- Santos, T., Coutinho, C., Cretan, A. and Jardim-Goncalves R., “Agents and Rules for the Negotiation of Interoperability Solutions”, accepted in: International IFIP Working Conference On Enterprise Interoperability (IWEI 2013). Mar 27-28, Enschede, Netherlands, 2013.

7. FINAL CONSIDERATIONS AND FUTURE WORK

This dissertation and the developed proof-of-concept was based on the project proposed by Adina Cretan and Carlos Coutinho et. al. in (Cretan et al. 2012) and (Coutinho et al. 2012) considering the work published by Ricardo Jardim-Goncalves in (Jardim-Gonçalves et al. 2010), which basically consists in a framework that offers to companies the possibility of negotiate their interoperability in order to make all companies involved satisfied when some harmonization break occurs. As described in the beginning of this dissertation, nowadays with the current turbulent economy is an asset to companies living in a sustainable environment which can offer stability and efficiency. Having this in mind, this prototype tries to offer an environment with these characteristics to the companies, where each company will not work alone, but becomes work together with all the companies in the environment.

With the implementation of the proof-of-concept that was succeed, all the tests done to the proof-of-concept, regarding the defined functionalities and requirements, were well succeed, showing that the developed framework is able to create a sustainable environment providing two types of negotiation that will meet the needs of the companies in the environment. The proof-of-concept is prepared to support the creation of more methods of negotiations, which is an asset to the environment if the involved companies need other negotiation methods. With the validation of the prototype through the executed tests over the framework, the dissertation hypothesis becomes validated, since that the framework was capable of create a negotiation sustainable environment, where the companies downtime when an harmonization break occurs, tend to decrease with time, since that in an early state of the environment, this time tend to be greater than a simple environment, without negotiation, as proven in the testing section.

This leads to a conclusion that this environment will be much better when it reaches a certain state or maturity, and when this state is reached, the environment create stronger and healthier interconnections between the involved companies which is much more resistant to changes and improvements which are inevitable in the course of time. Since today's economy markets are facing enterprises that changes their system frequently (Coutinho et al. 2012) the developed framework becomes more reliable, since that the environment will reach a maturity state much more quickly.

7.1. Future Work

In the future the proof-of-concept has much to improve since there are some points that need more attention, like the proposal questionnaire filled by the negotiation initiator in the beginning of a negotiation round. This questionnaire in the current version is static, i.e., the questions in the formulary are the same for all enterprises, which in the real world is not how should be, because each enterprise has their characteristics and the points that will change in

the system varies, not only from enterprise to enterprise but also varies from negotiation to negotiation.

Additional work foreseen in the future is the improvement in the knowledge acquired by the system controller. This work should be done in order to improve the accuracy of the suggestions that the System Controller does for each participant in the negotiation, which can contain more variables in order to be more realistic.

In a distant future it is great to think that this proof-of-concept can be converted to work on the cloud, which will bring much more flexibility to the enterprises to living on the environment and with that, a larger group of enterprises could work together over the environment.

8. REFERENCES

- ATHENA, 2010. *MDI - Model-driven interoperability*,
- Agostinho, C. & Jardim-Gonçalves, R., 2009. Dynamic Business Networks : A Headache for Sustainable Systems Interoperability. In *OTM '09 Proceedings of the Confederated International Workshops and Posters on On the Move to Meaningful Internet Systems*. pp. 194–204.
- Alonso, E., 2002. AI and agents : State of the art. *AI Magazine*.
- Anon, 2010. iSurf European Project. Available at: <http://www.srdc.com.tr/isurf/> [Accessed February 1, 2013].
- Bayegan, E. & Moslehi, K., 2011. Experience with rule engines in an outage scheduling system. In *2011 IEEE Power and Energy Society General Meeting*. Ieee, pp. 1–8. Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6039892>.
- Bellifemine, F. et al., 2003. JADE A White Paper. *EXP in search of innovation*, 3, pp.6–19.
- C4ISR, 1998. *Levels of Information Systems Interoperability (LSI)*,
- Camarinha-Matos, L.M., 2010. Scientific Research Methodologies and Techniques - Unit 2: Scientific Method.
- Chen, D. & Doumeingts, G., 2003. European initiatives to develop interoperability of enterprise applications—basic concepts, framework and roadmap. *Annual Reviews in Control*, 27(2), pp.153–162. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S1367578803000257> [Accessed November 9, 2012].
- Chmiel, K. et al., 2004. Testing the Efficiency of JADE Agent Platform.
- Coutinho, C., Cretan, A. & Jardim-Gonçalves, R., 2012. Cloud-based negotiation for sustainable enterprise interoperability. In *Engineering, Technology and Innovation (ICE), 2012 18th International ICE Conference on*. pp. 1–10. Available at: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6297698 [Accessed January 25, 2013].
- Cretan, A. et al., 2012. NEGOSIO: A framework for negotiations toward Sustainable Enterprise Interoperability. *Annual Reviews in Control*, 36(2), pp.291–299. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S1367578812000454> [Accessed December 12, 2012].
- Drools Rule Engine, Drools Rule Engine. Available at: <http://www.jboss.org/drools> [Accessed January 24, 2013].
- ETSI ES 201 873-1, 2012. Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language. , 1, pp.1–289.
- Elvesæter, B. et al., 2006. Towards an Interoperability Framework for Model- Driven Development of Software Systems. In *Interoperability of Enterprise Software and Applications*. pp. 409–420.
- Foundation for Intelligent Physical Agents, FIPA. Available at: <http://www.fipa.org> [Accessed January 29, 2013].

- Geraci, A. et al., 1991. *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*,
- 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>.
- IEC TC65, 2002. Industrial Process Measurement and Control.
- ISA, 2010. *European Interoperability Framework (EIF) for European public services*,
- ISO/IEC 14598-6, 2001. Software engineering – Product evaluation – Part 6: Documentation of evaluation modules.
- ISO/IEC CD 25040, Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Evaluation reference model and guide.
- ISTQB, 2011. *Certified Tester - Foundation Level Syllabus*,
- JACK Agent Framework, JACK Agent Framework. Available at: <http://aosgrp.com/products/jack> [Accessed January 20, 2013].
- JADE Agent Framework, JADE Agent Framework. Available at: <http://jade.tilab.com> [Accessed January 19, 2013].
- JBoss Drools team, 2012. *Drools Expert User Guide*, Available at: <http://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf>.
- JESS Rule Engine, JESS Rule Engine. Available at: <http://www.jessrules.com> [Accessed January 23, 2013].
- Jardim-Goncalves, R. et al., 2007. Harmonising technologies in conceptual models representation. *Int. J. of Product Lifecycle Management*, 2, pp.187 – 205.
- Jardim-Gonçalves, R., Agostinho, C. & Steiger-Garcia, A., 2010. Sustainable Systems ' Interoperability : A reference model for seamless networked business. In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*. pp. 1785–1792.
- Jason Agent Framework, Jason Agent Framework. Available at: <http://jason.sourceforge.net/wp> [Accessed January 20, 2013].
- Kim, T. et al., 2010. MiRE4OWL: Mobile Rule Engine for OWL. In *2010 IEEE 34th Annual Computer Software and Applications Conference Workshops*. Ieee, pp. 317–322. Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5615817> [Accessed November 20, 2012].
- Leeton, U. & Kulworawanichpong, T., 2012. Multi-Agent Based Optimal Power Flow Solution. In *Power and Energy Engineering Conference (APPEEC), 2012 Asia-Pacific*. pp. 1–4.
- Liu, D.I., Gu, T.A.O. & Xue, J.I., 2010. Rule Engine based on improvement Rete algorithm. In *The 2010 International Conference on Apperceiving Computing and Intelligence Analysis Proceeding*. Ieee, pp. 346–349. Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5709916>.

- Liu, X. et al., 2011. Research and application of Agent Communication Language Extended XML. *2011 Seventh International Conference on Natural Computation*, pp.1309–1313. Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6022380>.
- Massawe, L.V., Kinyua, J. & Aghdasi, F., 2010. An Implementation of a Multi-Agent Based RFID Middleware for Asset Management System Using the Jade Platform. In *IST-Africa 2010 Conference Proceedings*.
- Miller, J. & Mukerji, J., 2003. *MDA Guide Version 1.0.1*,
- Myers, G.J. et al., 2004. *The Art of Software Testing, Second Edition*, John Wiley & Sons, Inc.
- Nguyen, M.T., Fuhrer, P. & Pasquier-Rocha, J., 2009. Enhancing e-health information systems with agent technology. *International journal of telemedicine and applications*, 2009(ii), p.279091. Available at: <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2592555&tool=pmcentrez&rendertype=abstract> [Accessed October 26, 2012].
- OMG, 1999. *OMG Unified Modeling Language Specification*, Available at: www.omg.org [Accessed January 30, 2013].
- Oliveira, A.I. & Camarinha-Matos, L.M., 2012. Electronic Negotiation Support Environment in Collaborative Networks. *Advances in Information and Communication Technology*, 372, pp.21–32.
- Open Agent Architecture, Open Agent Architecture. Available at: <http://www.ai.sri.com/~oaa/> [Accessed January 21, 2013].
- OpenRules Rule Engine, OpenRules Rule Engine. Available at: <http://openrules.com> [Accessed January 24, 2013].
- Panetto, H., 2007. Towards a Classification Framework for Interoperability of Enterprise Applications. *International Journal of CIM*, 8, pp.727–740.
- Peristeras, V. & Tarabanis, K., 2006. The Connection , Communication , Consolidation , Collaboration Interoperability Framework (C 4 IF) For Information Systems Interoperability Defining interoperability. , 1(1), pp.61–72.
- Petzmann, A. et al., 2007. *Applying MDA ® Concepts to Business Process Management*. Interchang.,
- R. Jennings, N., Sycara, K. & Wooldridge, M., 1998. A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems*, 1(1), pp.7–38.
- Ray, S.R. & Jones, a. T., 2006. Manufacturing interoperability. *Journal of Intelligent Manufacturing*, 17(6), pp.681–688. Available at: <http://www.springerlink.com/index/10.1007/s10845-006-0037-x>.
- Ruggaber, R., 2006. ATHENA–advanced technologies for interoperability of heterogeneous enterprise networks and their applications. Available at: http://interop-esa05.unige.ch/INTEROP/Proceedings/IST/IST2_ATHENA.pdf [Accessed January 25, 2013].
- Schmidt, D.C., 2006. Model-Driven Engineering. In *IEEE Computer Society*.
- TTCN-3, Tree and Tabular Combined Notation 3. Available at: <http://www.ttcn-3.org/> [Accessed February 4, 2013].

- Tretmans, J., 2001. An Overview of OSI Conformance Testing. , pp.1–14.
- Truyen, F., 2006. The Fast Guide to Model Driven Architecture - The Basics of Model Driven Architecture. URL: <http://www.omg.org/mda/presentations.htm>, Available at: http://secure.omg.org/mda/mda_files/Cephas_MDA_Fast_Guide.pdf [Accessed January 25, 2013].
- Vernadat, F., 2003. Enterprise modelling and integration: From fact modelling to enterprise interoperability. *Enterprise inter- and intra-organizational integration: Building international consensus*, pp.25–33.
- Vernadat, F., 2004. Interoperable enterprise systems: Principles, architectures and metrics. In *Proceedings of international conference on management control and production logistics (Keynote paper)*.
- Vernadat, F.B., 1996. *Enterprise modelling and integration: Principles and applications* Chapman & ., London.
- Vernadat, F.B., 2007. Interoperable enterprise systems: Principles, concepts, and methods. *Annual Reviews in Control*, 31(1), pp.137–145. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S1367578807000132> [Accessed November 4, 2012].
- White, L.J., 1987. Software Testing and Verification. *Advances in Computers, Academic Press*, pp.335–391.
- Wycisk, C., McKelvey, B. & Hülsmann, M., 2008. “Smart parts” supply networks as complex adaptive systems: analysis and implications. *International Journal of Physical Distribution & Logistics Management*, 38(2), pp.108–125. Available at: <http://www.emeraldinsight.com/10.1108/09600030810861198> [Accessed November 6, 2012].
- Xu, B. & Xie, S., 2008. Research of dynamic rule engine in financial management software. In *Machine Learning and Cybernetics, 2008 International Conference*. pp. 12–15. Available at: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4620622 [Accessed January 25, 2013].
- Yin, Z. et al., 2012. Rule Engine-based Web Services Composition. In *World Automation Congress (WAC), 2012*. pp. 1–4.
- Zhang, Gang, Shan, W. & Wang, F., 2010. Research on the Promotion of Rule Engine Performance. In *Intelligent Systems and Applications (ISA), 2010 2nd International Workshop on*. pp. 1–3.
- i-Surf, 2009. Deliverable 8.1.1 - Functional and NonFunctional Evaluation Criteria for i-Surf Components and Integrated Platform. , pp.1–64.