Carlos Pedro Carvalho Raposo

Licenciado em

Engenharia Eletrotécnica e de Computadores

Methodological Framework for Detection of Harmonisation Breaking in Service Environments

Dissertação para obtenção do Grau de Mestre em

Engenharia Eletrotécnica e de Computadores

Orientador: Doutor Ricardo Luís Rosa Jardim Gonçalves, Professor Associado com Agregação, Faculdade de Ciências e Tecnologia (FCT), Universidade Nova de Lisboa

Coorientador: Doutor Carlos Manuel de Melo Agostinho, Investigador, Centro de Tecnologia e Sistemas (CTS), UNINOVA

Júri:

Presidente:	Doutor Pedro Miguel Ribeiro Pereira
Arguente:	Doutor João Francisco Alves Martins
Vogais:	Doutor Carlos Manuel de Melo Agostinho

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Fevereiro, 2015

Copyright

Methodological Framework for Detection of Harmonisation Breaking in Service Environments

To my Good Shepherd.

Acknowledgements

I would like to take a brief moment to acknowledge the effort, time, and contributions from others that were required to achieve the completion of this master's degree, which battled to the very last minute.

To my family, I thank them for being there, for being who they are, and for helping shaping who I am. This achievement is also dedicated to them, my grandparents, for their love and wisdom; my parents, for their guidance, support and reprimands to always strive for better; my brother, for his friendship, companionship, and joy.

To my friends, for their support and companionship in times of distress and in times of cheerfulness.

And finally, to all my colleagues at GRIS, UNINOVA, and all those who contributed to this accomplishment in a more educational way, namely my supervisor, Prof. Ricardo Jardim Gonçalves, and co-supervisor, Carlos Agostinho. To Prof. Ricardo Jardim Goncalves, I thank him for the opportunity to take part in his research & development team at UNINOVA.

"To everything there is a season, and a time to every purpose".

Thank you.

Agradecimentos

Gostaria de tirar um breve momento para reconhecer o esforço, tempo, e contribuições de outros que foram necessárias para alcançar a realização deste grau de mestre, que batalhou até ao último minuto.

À minha família, agradeço por estarem presentes, serem quem são, e ajudarem a moldar quem eu sou. Esta conquista é-lhes também dedicada a eles, os meus avós, pelo seu amor e sabedoria; os meus pais, pela sua orientação, suporte, e reprimendas para lutar sempre por melhor; o meu irmão, pela sua amizade, companheirismo, e alegria.

Aos meus amigos, pelo seu suporte e companheirismo em alturas de dificuldade e em alturas de felicidade.

E finalmente, a todos os meus colegas do GRIS, UNINOVA, e todos aqueles que contribuíram para este feito de uma forma mais educativa, nomeadamente o meu orientador, Prof. Ricardo Jardim Goncalves, e coorientador, Carlos Agostinho. Ao Prof. Ricardo Jardim Goncalves, agradeço pela oportunidade de tomar parte na sua equipa de investigação e desenvolvimento no UNINOVA.

"Tudo tem a sua hora; cada coisa tem o seu tempo".

Obrigado.

Abstract

As the complexity of markets and the dynamicity of systems evolve, the need for interoperable systems capable of strengthening enterprise communication effectiveness increases. This is particularly significant when it comes to collaborative enterprise networks, like manufacturing supply chains, where several companies work, communicate, and depend on each other, in order to achieve a specific goal. Once interoperability is achieved, that is once all network parties are able to communicate with and understand each other, organisations are able to exchange information along a stable environment that follows agreed laws. However, as markets adapt to new requirements and demands, an evolutionary behaviour is triggered giving space to interoperability problems, thus disrupting the sustainability of interoperability and raising the need to develop monitoring activities capable of detecting and preventing unexpected behaviour.

This work seeks to contribute to the development of monitoring techniques for interoperable SOA-based enterprise networks. It focuses on the automatic detection of harmonisation breaking events during real-time communications, and strives to develop and propose a methodological approach to handle these disruptions with minimal or no human intervention, hence providing existing service-based networks with the ability to detect and promptly react to interoperability issues.

Keywords: Interoperability, Dynamic Enterprise Networks, Web Services, SOA, Sustainable Interoperability, Monitoring.

Resumo

À medida que a complexidade dos mercados e a dinamicidade dos sistemas evolui, a necessidade de sistemas interoperáveis capazes de fortalecer a eficácia da comunicação entre empresas aumenta. Tal facto é particularmente significante quando se trata de redes de empresas colaborativas, tais como cadeias de produção de manufatura, em que diversas empresas trabalham, comunicam, e dependem umas das outras, a fim de atingirem um específico objetivo. Uma vez que é alcançada a interoperabilidade, isto é uma vez que todas as entidades de uma rede são capazes de comunicar e entenderem-se entre si, as organizações são capazes de trocar informação ao longo de um ambiente estável que segue leis acordadas. No entanto, à medida que os mercados se adaptam a novos requisitos e condições, um comportamento evolucionário é ativado dando espaço a problemas de interoperabilidade, e assim desregulando a sustentabilidade de interoperabilidade e levando à necessidade de desenvolver atividades de monitorização capazes de detetar e prevenir comportamentos inesperados.

Este trabalho procura contribuir para o desenvolvimento de técnicas de monitorização para redes interoperáveis de empresas baseadas em SOA. Foca-se na deteção automática de eventos de quebras de harmonização durante comunicações em tempo real, e esforça-se para desenvolver e propor uma aproximação metodológica para manipular este tipo de perturbações com o mínimo de ou sem intervenção humana, proporcionando assim a redes de empresas baseadas em serviços com a capacidade de detetar e prontamente reagir a problemas de interoperabilidade.

Palavras-chave: Interoperabilidade, Redes Dinâmicas de Empresas, Web Services, SOA, Interoperabilidade Sustentável, Monitorização.

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

AP	Application Protocol
API	Application Programming Interface
ASB	Adaptable Service Bus
B2B	Business-to-Business
BPD	Business Process Diagram
BPEL	Business Process Execution Language
BPM	Business Process Modelling
BPMN	Business Process Model and Notation
CAS	Complex Adaptive Systems
CAS-SIF	CAS to support Sustainable Interoperability Framework
CN	Collaborative Network
DMN	Dynamic Manufacturing Network
EI	Enterprise Interoperability
ES	Enterprise Systems
ESB	Enterprise Service Bus
ESB-CII	ESB Component for Integration and Intelligence
FurnitureLL	Furniture Living Lab
GRIS	Group for Research in Interoperability of Systems
HTTP	Hypertext Transfer Protocol
i-platform	IMAGINE platform
IDE	Integrated Development Environment
IDL	Interface Description Language
IMAGINE	Innovative end-to-end Management of Dynamic Manufacturing Networks
IoT	Internet of Things
IS	Information Systems
ISO	International Organization for Standardization
iSurf	interoperability Service utility for collaborative supply chain planning across multiple domains supported by RFID devices
IT	Information Technology
JBI	Java Business Integration
MAS	Multi-Agent Systems

MBSE	Model-Based Systems Engineering
MDA	Model-Driven Architecture
MDD	Model-Driven Development
MDE	Model-Driven Engineering
MIRAI	Monitoring morphIsms to suppoRt sustAinable Interoperability
MOM	Message-Oriented Middleware
MonALISA	MONitoring Agents using a Large Integrated Services Architecture
MySQL	My Structured Query Language
OS	Operating System
QoE	Quality of Experience
QoS	Quality of Service
SE	Systems Engineering
SMTP	Simple Mail Transfer Protocol
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol And RDF Query Language
SQL	Structured Query Language
SQuaRE	Software product Quality Requirements and Evaluation
SUT	System Under Test
TTCN	Tree and Tabular Combined Notation
UDDI	Universal Description, Discovery and Integration
UML	Unified Modelling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VAT	Value Added Tax
W3C	World Wide Web Consortium
WS	Web Service
WSDL	Web Services Description Language
XML	Extensible Markup Language
XPath	XML Path Language
XSL	Extensible Stylesheet Language

Chapter 1

"[Interoperability is] the ability of two or more systems or components to exchange information and to use the information that has been exchanged." - (IEEE Computer Society, 1990)

1 INTRODUCTION

In order to reach global business and markets, enterprises are required to collaborate and establish partnerships (Commission of the European Communities, 2005). However, different organisations operating in the same business domain may have different views of the same subject, leading to distinct computational models and, therefore, to interoperability problems when their systems intend to share information. Consequently, in order to succeed in the collaboration, Enterprise Systems (ES) and applications need to be interoperable, i.e. be able to share technical and business information seamlessly within and across organisations, and must be adaptable to different network environments at all life cycle phases (Ray & Jones, 2006). Moreover, with the increasing complexity of markets and the emergence of dynamic systems, raises the need to create interoperable systems and software applications capable to adapt to the ever changing market demands (Jardim-Goncalves, Grilo, & Steiger, 2006). This dynamicity hinders the sustainability of interoperability and calls the need to monitor and trace the network at its micro level (local Information Systems (IS)) in order to support interoperability maintenance at the macro level (i.e. the network) (Ferreira, Agostinho, Sarraipa, & Jardim-Goncalves, 2012). This is particularly true when it comes to dynamic enterprise networks (e.g. Dynamic Manufacturing Networks - DMN's), where a distinct group of companies is connected in a chain-like model and where cooperation is crucial to achieve a specific goal.

1.1 CONTEXT AND MOTIVATIONS

Nowadays, many enterprises are starting to communicate through the use of the increasingly popular Service-Oriented Architectures (SOAs) (Lee, Shim, & Kim, 2010), as it facilitates the exchange of information between different organisations, at the same time that provides an interoperable platform for communication with no need for additional requirements

or modifications. Being the Web Services (WSs) approach one of the most widely used implementations of SOA, they may provide a robust way of communication. However, SOA is still a step away from being fully interoperable, i.e. different domains may have different types of interoperability frameworks that focus on different aspects within the network, such as communication, message formats, security, etc.; a web service client can change the service provider dynamically, leading to different outcomes both for the client and the provider; among other factors. Moreover, if the data being exchanged between a client and a provider suddenly takes the form of a new computational/information model, existing systems in the network would not know how to handle it and communication would fail (Lee Y. , 2010), (Ibrahim & Hassan, 2010).

Agostinho and Jardim-Gonçalves in (Agostinho & Jardim-Goncalves, 2009) introduced the sustainable interoperability concept, aiming to improve the Quality of Service (QoS) by contributing to a more robust interoperability, avoiding excessive consumption of resources, like man-power or time, when the dynamicity of the system and the network causes interoperability harmonisation breaking, in which the term harmony defines the agreement and conformity between the various entities in a stable environment. It is a novel concept whose objectives are to reconcile the economic interests ensuring that the network maximizes its efficiency by remaining interoperable at most times. The authors acknowledge that by using knowledge representation techniques for managing the links between different organisations and the use of software agents dedicated to monitoring activities, each organisation gains the capacity of detecting changes and readapt to cooperating companies (Jardim-Goncalves, Agostinho, & Steiger, 2012).

The work here presented seeks to explore the concepts of enterprise and service-based interoperability and analyse and contribute to the development of monitoring techniques for interoperable SOA-based enterprise networks using web services, in an effort to achieve and maintain network interoperability at most times. It focuses mainly on the automatic detection of harmonisation breaking during real-time communications, and strives to develop and propose an approach to automatically handle the disruption of harmonisation with minimal or no human intervention. The goal is to provide existing SOA-based networks the ability to not only automatically detect occurring problems, but also to automatically approach and solve them whenever possible.

1.2 The Research Method

Before any approach can be taken towards the solution of a given problem, it is important for an adequate research method to be defined. For the purposes of this work and the developments here presented, a methodology based on the scientific method is used.

The scientific method consists in a set of techniques used to gather information related to a pre-defined subject, the formulation and testing of hypotheses according to the information obtained, and finally the retrieval of conclusions concerning the tests performed (Murray, 1999). Although there are several variations concerning the scientific method, these form the basic set of elements upon which all the variants are based on.

1.2.1 Adopted Research Method

While the scientific method is presented in a wide range of different models, it is ultimately the author's choice which method is the most suitable and which way is the most appropriate to address the issue in question.

After evaluating the subject presented in this master thesis work, a research methodology was formulated as suggested in (Schafersman, 1997) and is displayed in Figure 1.1.

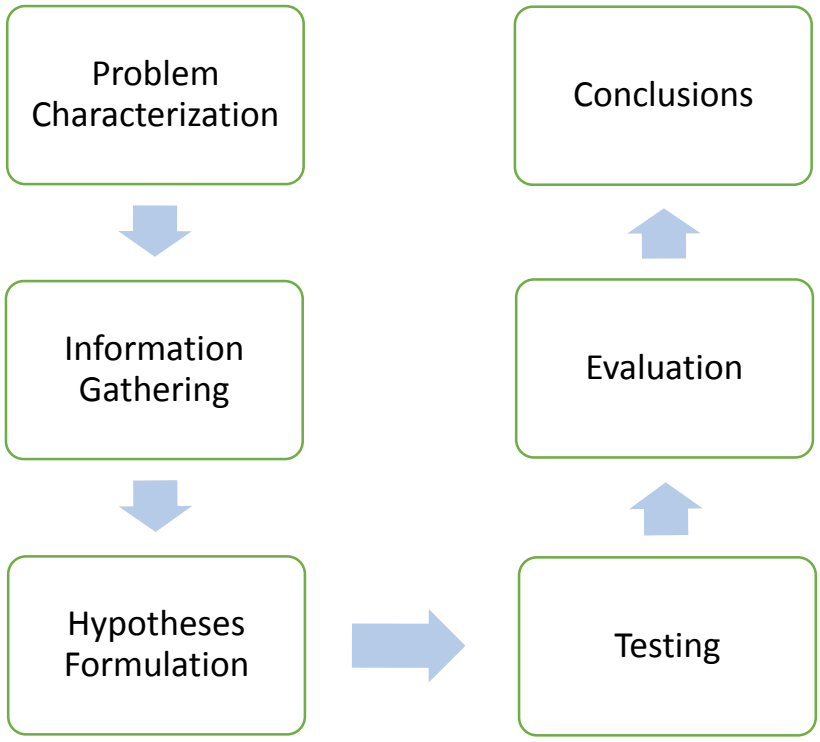

Figure 1.1 – Adopted research method.

Problem Characterization comprises identifying a meaningful question or problem, the type of factors that may be involved, and possible ways to address the issue. It is a significant part of the methodology as all the research work will be centred on the selected problem, which is why it is important to properly identify and describe the issue in detail, taking into consideration if it

is plausible and whether a viable solution exists. Then comes **Information Gathering** which, as the name implies, consists in gathering and retrieving relevant information about the issue being treated, enabling, consequently, the **Formulation of Hypotheses**. This step consists in analysing the retrieved information and proposing a valid and testable solution to the problem. Once a solution is proposed, it is time to validate it by **Testing**. This produces the necessary results that allow the **Evaluation** of the proposed answer and whether it serves its purpose or not. If it fails to do so, then it may be needed to perform further testing or to go back and re-formulate a new hypothesis. If it succeeds, then the solution is valid and final **Conclusions** can then be retrieved.

1.2.2 Research Problem and Questions

The topic addressed by this work is the development of a methodological framework for detection of harmonisation breaking in service environments, as indicated in this dissertation's title.

Consequently, the questions that arise in relation to the chosen subject are associated with the capability and feasibility of maintaining a sustainable and interoperable environment in a collaborative service-based network, as described as follows.

- Considering a collaborative service-based network where interoperability has been successfully established and where members exchange information following established laws, is it possible to promptly detect a break in the system's harmony as its members adapt to new requirements, avoiding therefore a disruption in interoperability?
    - If it is possible to timely detect these harmonisation breaking events, can they be avoided?
    - In the event that a break in harmonisation cannot be prevented, is it possible to recover from it and prevent or reduce the lack of interoperability?
        - If recovery from a break in the system's harmony is possible, can it be automated or with minimal human intervention?

As this thesis seeks to contribute to the sustainability of interoperability in service-based environments, these questions form the basis on which the research work here presented is based on.

1.2.3 Hypothesis

Considering the research questions previously identified it is conceivable to conclude that, in a collaborative service-based network, if one can effectively monitor the interactions that occur between different entities, then one can also strive to timely intervene in the event of harmony

disruptions, seeking in this manner to maintain and uphold sustainable interoperability across the network and prevent harmonisation breaking.

1.3 Dissertation Outline

This master thesis focuses on studying monitoring techniques and capabilities in service-based dynamic enterprise networks, namely in the domain of manufacturing. Moreover, it strives to develop a methodological framework to aid this process of monitoring and also enable the ability to autonomously react to unexpected harmonisation breaking events. This dissertation is outline according to the following topics:

- Chapter 1 introduces the main context and motivations behind this work, as well as the research methods used to approach it;
- Chapter 2 describes the concepts of interoperability, service-based systems and WSs related technologies;
- Chapter 3 discusses the concepts of modelling and its effects on the sustainability of interoperability and presents the state of the art on some existing monitoring technologies and frameworks related to service-based networks monitoring;
- Chapter 4 discusses the preparatory steps for the development of the proposed methodology as well as the methodology itself;
- Chapter 5 regards the proof of concept scenarios, developments and implementations achieved during the elaboration of this research work;
- Chapter 6 validates the implementations previously discussed in chapter 5 and presents the achieved results;
- And finally, chapter 7 discusses the conclusions obtained in regard to the results obtained, and discusses some of the future work that should be performed to further validate and progress on the developments here achieved.

Chapter 2

"[Enterprise is] one or more organisations sharing a definite mission, goals, and objectives to offer an output such as a product or service." - (ISO, 2000)

2 SERVICE-BASED INTEROPERABILITY

Interoperability is the ability that two or more systems have to communicate and understand each other. Hence, when all the entities in a network of enterprises can uniformly interact between themselves and work jointly towards a given goal, then the network is said to be interoperable. With the globalization of markets, the need to collaborate with different societies, cultures, and languages grows, leading to ever increasing problems of coordination and cooperation and making the ability to homogeneously interact with external partners a key issue in the economic and public sector, where the speed, cost, and quality of production highly depends on the effective cooperation between the multiple parties involved (Chen & Vernadat, 2002). This makes the subject of interoperability even more relevant, since without it there can be no cooperation and therefore no realization.

While enterprise interoperability is of vital importance for the successful growth of a business, companies find it difficult to establish and maintain electronic relationships with a large number of external partners. They often need to exchange resources, data and information, as well as coordinate an array of links with multiple entities to ensure that the objective is achieved at the right time with the specified quantity and quality. Moreover, with the growing tendency to focus on core competencies, the number of external partners grows even larger. This leads to companies operating in a chain-like network, where several entities are bound to each other and depend on each other for their different needs (Lebreton & Legner, 2005).

This chapter seeks to explore the concept of enterprise interoperability, such as existing architectures and technologies. Given that this thesis focuses on service-based networks, the notions of service-based interoperability will be given special emphasis, in order to better understand the models of interoperability behind service environments. Once understood what the requirements for an interoperable system are, methods to sustain such systems are discussed.

2.1 SERVICE SYSTEMS

Over the last years, society has grown to make services one important part of the world economy. However, in spite of such expansion, the definition of service is still hard to describe (Spohrer, Maglio, Bailey, & Gruhl, 2007). One of the reasons for such difficulty, may be the fact that the term service is so broad that can be attributed with a range of different meanings. Nevertheless, according to the English Dictionary, one can generally define service as something meant to perform a given task in order to provide assistance. These service tasks are often achieved with the support of a client or the assistance of another service. The more complex the task, the bigger is the need and dependence on those other external clients and services, whether it is in regard to information, labour, or something else (Sampson & Froehle, 2006). This type of linked structure leads to a network composed by several different services and clients all able to communicate and interact with each other, share information, and directly or indirectly dependent on each other, forming in this manner a service system (Tien & Berg, 2003).

A service system is, therefore, a collection of entities linked together and capable of interaction with each other, governed by a defined set of rules, and working towards the production of value. In order for such a system to work, good coordination is essential. This means that the entities involved in the service system need to be able to effectively communicate and share information with the other existing entities. Such coordination can only be achieved when working under a structured and pre-defined set of laws in which important parts related to information sharing are already well established, as in the case of language, information encoding, and standards in use, between others. Without these governing laws, complex and sophisticated services cannot function effectively and their capability for providing assistance is limited. This also means that each service must have a way to present itself to the remaining entities and make known what type of assistance it provides. This is achieved with service descriptions by means of an Interface Description Language (IDL), in which each service describes in detail all the operations it is able to perform and the requirements to do so.

When dealing with such large systems, the assistance supplied by the different services and the value produced by those, depends on the competencies and capabilities distributed among the existing entities. It reflects the essence of the motto *unity makes strength*, as the collection of all the distinct abilities and proficiencies forms the bulk of the system's overall capabilities. This also means that when a part of the system is incapable of producing value, then the whole system is hindered and affected, leading to a disruption in the system's overall output. Therefore, it is essential to develop and understand ways and techniques that can help to improve and maintain the system's efficiency, stability and sustainability, as well as how a particular service or entity may be improved in order to boost the system's overall potentiality.

This type of chain-like systems where coordination is crucial are, therefore, rather fragile. If one of the existing entities decides to go against the previously agreed laws, if a network failure occurs and communication is disrupted, or even if a new and uninstructed entity firstly joins the system, the entire service architecture is put at risk and faces the possibility of being hindered.

2.1.1 Distributed Systems and Networks

A distributed system is a system in which networked computer components interact and communicate with each other only by exchanging messages (Coulouris, Dollimore, Kindberg, & Blair, 2011). This means that the definition of distributed system and service system are closely related, since a system in which components interact by exchanging messages through the use of services, fits in the description of both concepts. However, the definition of distributed system implies three important characteristics.

The first important aspect in regard to distributed systems is related with concurrency. In a network of computers it doesn't matter what each particular machine is executing. Computer A can be executing program X while computer B can be performing a specific task in program Y. Moreover, both computer A and computer B can be sharing resources even though they are executing two different applications at the same time. This is a particularly important aspect of distributed systems: the capability to execute multiple concurrent programs while coordinating and handling shared resources simultaneously.

This closely monitored coordination brings up the subject of time. Usually, when two concurrent programs interact with each other, they base their interactions on a shared idea of time. However, in a computer network where communication is centred on the exchange of messages, this global notion of time is hard to extend through the various existing machines, which means that each computer ends up working based on its own independent clock. Time synchronization, therefore, plays an important role in the interaction of distributed systems.

Finally, another relevant characteristic of this type of systems is failure independence. When computer A experiences a given fault and crashes, it doesn't impede computer B from performing its task and interact with computer C. In a similar way, if a part of the network is experiencing problems, it doesn't mean that the machines affected need to stop executing their duties. This means that in a distributed system each component can fail independently, while leaving the remaining components unaffected. However, the fact that computer B is not directly affected by a crash in computer A, it doesn't mean that the task being performed at computer B will succeed, since its realisation may be dependent on computer A. While the overall system's stability may be enhanced by this failure independence characteristic, several issues can originate

from it that can eventually lead to a disruption of the system's interoperability, turning this specific aspect into a subject of particular significance.

While all the enumerated characteristics have their benefits and drawbacks, inciting for further exploration and study, the grand value of distributed systems resides in the ability to seamlessly interact and share resources with an external array of different machines, a rather important aspect when dealing with service systems. They are used across a wide variety of different applications and encompass an extensive range of distinct technologies. Search engines and web search technology are one example of the powerful potentiality that these type of systems provide, as they constitute a vast and complex architecture, from physical structures to resource sharing, storage systems, security, and the like. Another example of a distributed systems implementation can be observed in online video games, in which a massive number of users interact through the Internet. This type of implementation requires not only high-end machines capable of handling huge data loads, but also a fast and reliable end-to-end communication in order to preserve the gaming experience.

As economy shifts from a paradigm of mass production to mass-customization, companies are forced to search for partners that can focus on their production needs while minimizing the time to market (Pine & Gilmore, 1999). This trend leads companies into Collaborative Networks (CN), where they can easily cooperate and interact with each other in order to achieve their goals. DMN's are an example of such networks, where globally distributed entities form a temporary partnership in order to perform joint manufacturing, at the same time that they avoid production management problems (Papakostas, Efthymiou, Georgoulias, & Chryssolouris, 2012). Having each company focused on its own proficiency allows for a distributed-based production chain, where each entity is responsible for a different part of the product. This not only ensures a greater product quality, as each company is able to maximize its core skills by focusing on smaller tasks, but also a production time and cost reduction. A different example of a collaborative network is that from the Internet of Things (IoT), in which multiple devices are linked to each other and are able to interact with each other, independently of hardware and middleware differences. It enables the integration of everyday objects and devices leading to a highly distributed network of systems, capable of interaction between them and the Internet (Atzori, Lera, & Morabito, 2010).

Despite the differences and characteristics of each device, it is possible to form a collaborative network of distributed systems where different entities are connected together and able to cooperate towards a specific goal. However, there's one similarity in all of these systems: they all enable interaction through the use of services, whether when collaborating within the network or when communicating with the outside world. While a range of different applications implementing distributed systems and providing a stable and reliable experience can be

enumerated, technology keeps moving forward and so does the need to upgrade and adapt the existing network structures. In the latest years for instance, the demand for online multimedia services has grown drastically, raising the need to improve many of the existing architectures in order to be able to respond to such demand. This and other factors lead to a fluctuating system that is constantly facing change in an effort to cope with new network requirements. For such a system to retain stability and prevent itself from failing its users, a great deal of effort must be put into achieving and persevering interoperability.

2.2 Multi-Agent Systems (MAS)

In a similar way as the definition of service the concept of agent can be rather ambiguous, since the various characteristics associated with agency have different relevance in different areas and domains. While there is no universal accepted definition of the term agent, Wooldridge and Jennings, in (Wooldridge & Jennings, 1995), have defined an agent as a computer system located in a given environment that is capable of autonomous action in order to achieve a given objective. As depicted in Figure 2.1, the agent is able to receive input from the environment it resides in and produces output capable of exerting influence over it. However, the agent does not have complete control over its environment. Furthermore, an agent may have a range of different actions available to be performed, being the choice about which action to perform the major challenge the agent faces. In this manner, software architectures comprised of multiple agents, described as agent architectures, function as decision making systems that help in the decision making process of an agent in a given environment.

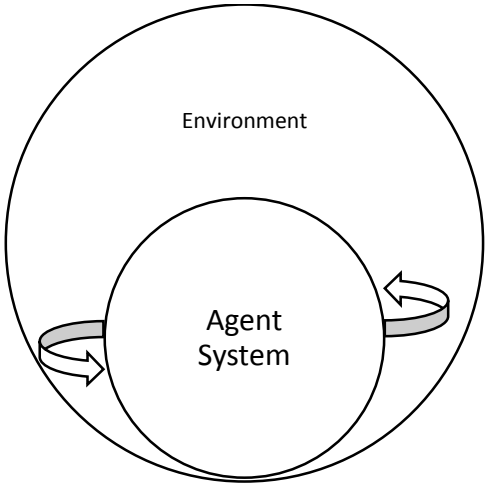

Figure 2.1 – Relation between a system of agents and its environment.

As suggested by Wooldridge and Jennings, an agent is therefore a hardware or software-based computer system that comprises four main characteristics, as enumerated below:

- *Autonomy* – an agent is able to operate and act without the direct intervention of humans or other external factors and has a minimum amount of control over its own actions;
- *Reactivity* – an agent is able to perceive its surrounding environment and timely react to occurring changes according to its purposes;
- *Pro-activeness* – an agent actions are not confined to its environment but take into consideration the agent's goals and initiatives;
- *Social ability* – an agent is capable of interaction with other agents and the surrounding environment via some pre-defined agent communication language.

With these four properties in mind, it is safe to affirm that an agent is an autonomous entity that is able to respond to environment changes and is capable of collaboration with other entities towards the accomplishment of a given goal, much like the description of service as illustrated before. A system comprised of multiple agents working together is known as a multi-agent system (Wooldridge, 2009).

One great advantage when using MAS is the seamless ability to integrate the several agents and use them for cooperative work, avoiding some of the inherent interoperability issues that one might expect to exist in such large and distributed systems. By using standardised protocols for communication, MAS are therefore able to achieve and generally maintain a stable and interoperable environment, assuring in this manner a global set of rules and laws that are common between the system entities and contribute to the sustainability of interoperability. While service-based systems tend to focus particularly on service integration, MAS usually take into consideration features like adaptability and dynamism, and although agent technology may not fully realize the core functionality of a service, it may substantially contribute to make it more stable, intelligent, and flexible, as these constitute tasks that can be more easily performed by agents rather than comparing the interfaces of service descriptions (Calisti, Leymann, Dignum, R., & Unland, 2010).

Considering their main characteristics, Calisti et al. suggest that agents can therefore be used to support the implementation and management of services. For instance, service-based networks can be rather dynamic and the feature of agent's pro-activeness may help in the timely detection of system changes and the adaptability of the affected services to those changes. Agents may also be able to better understand service semantics as they do not need to rely on service descriptions, providing in this manner a wider range of choices for service composition and greater flexibility. Due to their effective ability for cooperation, agents may also contribute to the

negotiation of service assembling while providing a more intelligent and reliable fault-tolerance system by dynamically replacing faulty services with alternative ones. These and other features, such as an improved security during service communications, add to the notion that MAS can effectively contribute to the implementation of service-based networks by offering fundamental features like flexibility, adaptability, and dynamism. Moreover, agent-technology can also help in the monitoring of these networks and assist in the operation of a more stable and reliable system, as briefly discussed in chapter 3.

2.3 Web Services

As distributed systems applications grow, so does the need to promote and ease the sharing of virtual resources. However, one of the great advantages of these systems is their greater flexibility when it comes to resource management. This is achieved by means of virtualization, the transformation of something into an artificial and virtual version of itself, whether it is a physical or virtual resource. Hence, this virtual sharing is often performed over services that do not necessarily own their resources but rather provide a virtual version of the same. For instance, a user can make use of a service to obtain a specific file that is in fact located on a different part of the network in a storage facility. This type of technology is key in the adoption and growth of distributed systems and is applied both to software resources, such as data, and hardware resources, such as printers (Coulouris, Dollimore, Kindberg, & Blair, 2011).

WS technology provides a service interface that allows service consumers to directly interact with servers by means of a specific operation. To know which operations are available to be performed by a given service and which requirements need to be fulfilled, clients need to access the service interface, which allows to fully describe a service in an IDL. This interface is usually presented in eXtensible Markup Language (XML) (W3C XML Core Working Group (WG), 2008), a markup language that defines a set of rules for document encoding and allows for an easier reading and debugging, over the Hypertext Transfer Protocol (HTTP) (Fielding, et al., 1999), an application protocol for data communication. It describes not only the service operations but also the encoding and communication protocols available as well as the service locations. As the usage of WSs grows wider, they become an increasingly important part in distributed systems, allowing a greater support of interoperability throughout the Internet (Coulouris, Dollimore, Kindberg, & Blair, 2011).

While a general browser intended for Internet navigation enables the user to interact with the server in a seamless manner, the potential of applications for more complex interactions is somewhat limited when using such clients. WSs allow a specific client application to directly interact with a specialized service, providing a more structured infrastructure for communication

and increasing in this manner the level of interoperability. Moreover, WSs allow the development of complex applications by providing means for a WS to integrate several other WS, a process known as service orchestration. Similarly to the way a browser interacts with a server over HTTP, the consummation of a WS is based on a request-reply process, in which the client initially requests the service to perform a specific operation, as described in the service interface, the server handles the data and executes the request, and finally replies to the client with a message. This communication process can be achieved by means of several existing protocols, as discussed further below.

In order for a web server to be consumed by a client, it first needs to be publish by the provider and then discovered by the consumer. Any organization that intends to build its applications centred on WS, will find it effective to use a WS registry or service directory, such as the Universal Description, Discovery, and Integration (UDDI) specification (OASIS, 2004), which provides a registry system where web services applications can be registered and located. WS registries function similarly to the Yellow Pages, where one can look for a given service application by name or attribute through the use of a specific Application Programming Interface (API). In short, it allows the registration, publication, and look up of web services in a central directory in a process known as service discovery. For the WS to be published however, it needs to be described by means of an IDL, as the Web Services Description Language (WSDL) (W3C, 2007). The WSDL is an XML-based IDL used for describing the functionality of a web service, such as the operations it performs, the encoding that it uses, and the communication protocols it allows. Each operation described in the interface must specify the requirements needed for its invocation, such as the input and output parameters. In this manner, a service description assembles all the information needed to consume the service in one place, and forms the basis for client to provider communication. WSDL documents can be accessed either directly, via their Uniform Resource Identifiers (URI), or indirectly, via a WS registry. Once a WS is described and published it is then ready to be consumed by a client. The Simple Object Access Protocol (SOAP) (W3C, 2007) allows the exchange of XML messages via HTTP, Simple Mail Transfer Protocol (SMTP) or other application protocols, serving as one of the main communication protocols used by WS. SOAP uses XML to represent the contents of the messages and allows for asynchronous client to server communication. Messages are encapsulated within an envelope with a header and a body that can contain additional information for establishing the context of the service or to keep a log of operations. In this way, clients use the service description to generate the message envelope, with the appropriate contents, and the server then parses the information and validates it. An alternative to using WSDL and SOAP is the Representational State Transfer (REST) approach (Fielding, 2000), commonly known as RESTful, in which clients use Uniform Resource Locators (URL) and HTTP operations, such as GET, POST, PUT and DELETE, to manipulate

data. The RESTful approach emphasizes on resource manipulation rather than service interfaces by supplying the client with the entire state of a resource rather than an operation to interact with it. Although widely used, REST is not standardised and will not be discussed further. In summary, a WS possible operations and message patterns are described by means of an IDL, such as WSDL, and published on a server by the provider. Optionally, the WS can be registered in a WS registry or service directory to ease to process of WS discovery. Once published, the client can consume the WS by reading the description document and generating a message with the required contents that can then be transmitted using the SOAP protocol. Figure 2.2 illustrates a WS infrastructure and components in a hierarchal view, summarising the technical functionality of a WS.

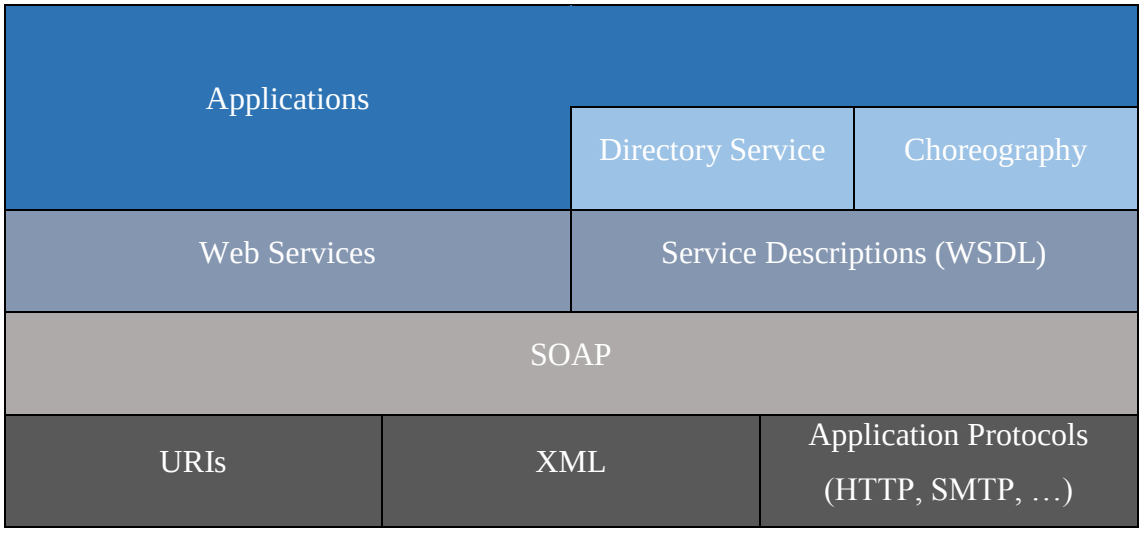

Figure 2.2 – WS infrastructure and components. (Adapted from (Coulouris, Dollimore, Kindberg, & Blair, 2011))

One of the great advantages of WSs is the ability to seamless integrate them between each other in order to accomplish a specific task. WSs interact between them in the same way they interact with a client, and their interfaces allow for their operations to be shared and combined with the operations of another. This arrangement of different WSs is known as service orchestration, as mentioned previously, and allows for a greater functionality and overall system capability, enabling different services to perform joint tasks. In this manner each WS can specialise itself and focus on a small and specific task rather than a big and complex one, since the latter ones can be accomplished with the assistance of multiple specialised WSs. Another similar concept is the one of service choreography (Peltz, 2003). In service orchestration, interaction logic is performed from the point of view of a specific service, called the orchestrator. It functions as an organized and coordinated chain of events in which each participant's action is the follow up and the consequence of the one before. Service choreography, on the other hand,

functions as a more dynamic mode of operation where interaction logic is specified from a global perspective and collaboration is the main focus, allowing for a more flexible system with no centralised point of control. It is intended and more appropriate to support interaction between WSs managed by distinct companies. However, both concepts represent functioning and effective composition solutions with their own benefits, allowing for a greater control, coordination, and WS variety, while also helping to maintain system interoperability, since a failure in a particular WS represents a failure in a specific and minor part of the overall system, enabling the possibility for other WSs to seek for alternate partners capable of performing the same operations. Nevertheless, as the work here presented focuses on the concept of interoperability, a central solution such as the one of service orchestration is more appropriate, as it allows for a better coordinated and planned chain of events in a centralised and controlled manner, as opposed to the somewhat unpredictable and dynamic mode of operation of service choreography. The Business Process Execution Language (BPEL) is an example of a service orchestration language and will be discussed with more detail in section 2.4.1.

2.3.1 The Service-Oriented Architecture

Being WSs one of the most dominant and rapidly growing paradigms for programming distributed systems (Curbera, Khalaf, Mukhi, Tai, & Weerawarana, 2003), their applications extend a wide range of different areas. Such is the case of SOA, an architectural style which aims to promote and facilitate the seamless integration of distinct services that can be dynamically discovered and then communicate with each other in an organized and coordinated manner (Erl, 2005). As markets grow and become increasingly dynamic, SOA offer a flexible, robust, and interoperable solution for service-based systems and their external partners. They enable WSs to be looked at from a global perspective and use this technology, by means of WS interfaces, as a solution to deal with the issue of different entities adopting different technologies, promoting in this manner the concept of system interoperability. This mode of operation is known as Business-to-Business (B2B), in which a given company may interact with a wide range of different partners in order to achieve its goal. In summary, SOA seek to achieve a service-based infrastructure capable of rapidly reacting to dynamic market demands, by establishing a task-oriented environment that aims to use current technologies efficiently in order to perform companies' key services. As service systems, SOA consist of an assembly of services in which each one is capable of realizing a customized and specific task, promoting business cooperation in order to achieve greater overall business performance.

While SOA have emerged with the intent to promote better system collaboration and interoperability, they still face relevant issues and challenges in achieving their purposes, making the development of interoperability frameworks an important and most needed achievement

(Ibrahim & Hassan, 2010). Even so, the interest in this type of architecture keeps growing with many organizations already using or planning to adopt it (Lee, Shim, & Kim, 2010). SOA present one of the most common applications of WS, although other applications exist, such as Grid computing and services, a middleware technology designed to ease the process of resource sharing on a large scale (Ahmed, 2006), and Cloud computing, an Internet-based application composed with a range of different services and storage solutions used to support most user's needs (Mirashe & Kalyankar, 2010).

2.4 The Enterprise Service Bus (ESB)

The necessity to integrate different services and applications between different companies raised the need to develop infrastructures capable of easing the communication process between them in an interoperable way. This implied the creation of a service-oriented infrastructure combined with other solutions for enterprise integration, such as Message-Oriented Middleware (MOM) and routing intelligence. MOM is a software or hardware infrastructure that enables the exchange of messages between different entities in a distributed system, in which each client is able to communicate with the other clients by means of an intermediary server, using a unified interface to receive and send messages, in a flexible and cohesive environment (Curry, 2004). The server system to which MOM applications are connected functions as a message broker that receives the messages sent by the client applications and stores them, relieving the recipients of the need to be connected at all times. It also allows to route the messages to multiple receivers and, if need be, to transform them in order to comply with the requirements of each one. Although the MOM infrastructure presents a traditional solution for enterprise communication in distributed systems, it faces several problems in regard to the protocols and interfaces used by each of the applications, lagging behind in terms of interoperability.

Even so, as organizations seek for new ways to integrate different companies and services in an effort to enhance distributed system's communication and interoperability, new infrastructures combining existing technologies and solutions were proposed. The combination of the flexible and structured SOA with traditional communication solutions such as the MOM, led to the emergence of the ESB, a software architecture model that serves as a middleware layer and is responsible for enabling the transportation and transformation of data between different applications and services (Menge, 2007). By means of specific languages and modelling tools, the ESB allows to manage and coordinate all the interactions between the different resources in a simplified way and without the need to write great amounts of code. It constitutes an open standards infrastructure based on SOA and WS technology that provides routing, invocation and mediation services in order to ease the integration and interaction process between different services and applications, ensuring in this manner a reliable and interoperable environment while

allowing for a flexible and scalable system. As enumerated by Menge, the ESB compiles a number of compelling features, some of which are described below:

- *Invocation* – The ability to interact with integrated services and applications by means of a request/response method. Hence, for such ability to work effectively, the ESB must support all WS related technology as well as the underlying protocols and required standards;

- *Routing* – The capability to route messages between different services, whether by directly addressing it towards its destination or by means of content-based routing, in which case the destination is extracted from the message itself;

- *Mediation* – The ability to perform transformation and translation operations as needed at the moment of interaction between distinct services or applications, whether they differ in regard to protocols, message formats, or message contents. Mediation is an important feature for integration, since it is very common for different services or applications to differ in regard to data formats, a factor that could potentially disturb interoperability. By using XML for data encoding, this feature is powered by standards such as the Extensible Stylesheet Language (XSL) and the XML Path Language (XPath);

- *Adapters* – Allows to ease the process of integrating the existing applications into the system's infrastructure, at the same time that minimizes the skills required to use each connected system and enables the reutilization of data structures;

- *Security* – The ESB enables the encryption and decryption of information and content messages, while also providing authentication handling and other security mechanisms;

- *Management* – By using a central point of control, the ESB allows for an easier system management and configuration;

- *Process orchestration* – One distinct feature of the ESB is the ability to orchestrate entire business processes without the need to write code. This can be achieved by use of specific orchestration and modelling languages such as the BPEL, addressed further ahead in this section.

The features here listed present some of the most notable strengths when using an ESB. However other useful features could be mentioned, such as the ability to handle and process complex events, the possibility to perform process choreography, the ability to use data validation mechanisms, or the existence of specific tools and development environments that contribute to ease the creation of applications and processes, between others (Chappell, 2004).

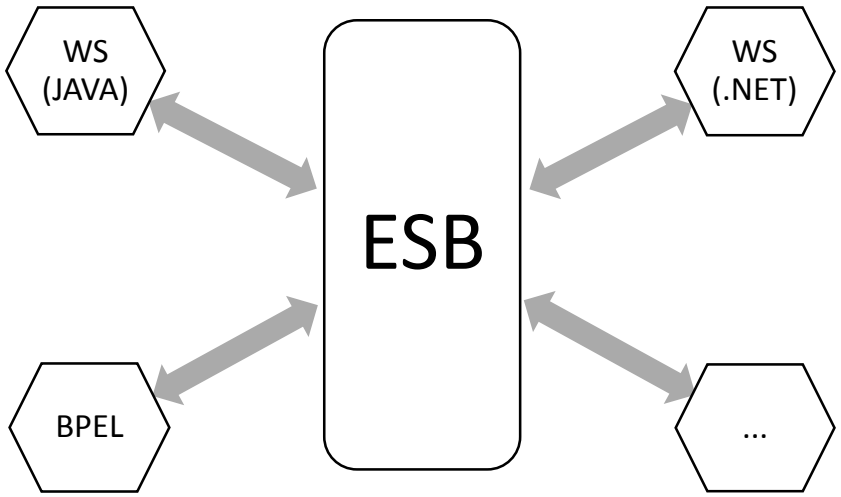

Figure 2.3 –The ESB, a software architecture model design to integrate services in a distributed system.

In summary, the ESB is a SOA-based software architecture model used to integrate distinct services and applications in a distributed system, providing the system managers with a central point of control for simplified management and configuration, an easier way to integrate existing resources, and the flexibility and scalability that such an infrastructure needs to remain stable and interoperable in the long term future. When using an ESB, all integrated services and applications are able to exchange information with each other without the need to worry with possible interoperability issues, as the ESB first receives the message from the sender, processes all the required steps such as data formatting, message transformation, and routing, and then sends the standardised message to its recipient, as illustrated in Figure 2.3. Hence, for a complex and distributed system with multiple services and applications each with its own technology and standards, the ESB represents a relevant and important benefit, as demonstrated by all the features listed previously.

2.4.1 Business Processes and Business Process Modelling (BPM)

In order to describe the interactions between different services, the ESB allows the use of specific languages and modelling notations to specify the processes that govern them. A business process is a collection of tasks logically structured and orchestrated in order to perform a specific activity. There are several different tools and programming interfaces that enable the creation of business processes, as is the case for the already mentioned BPEL, short for WS-BPEL (OASIS, 2007), or the Business Process Model and Notation (BPMN) (OMG, 2011). The logical structuring and coordination of multiple services is called service orchestration, as explained previously in section 2.3.

As described in (OASIS, 2007), the WS-BPEL defines a model and a grammar for specifying the interactions between different partners in what is known as a business process. These interactions occur by means of WS interfaces and are coordinated by the BPEL in order to achieve a specific business goal, which provides the functionalities and logic required to specify such orchestration. The BPEL also enables the use of fault handling mechanisms to prevent exceptions and allows to define how a certain activities might be compensated when an error occurs. It is based on several exiting standards and specifications such as WSDL and XML for data models, or the XSL and XPath for data manipulation, and delivers clearly specified execution semantics, while benefitting from a strong support in the industry. Figure 2.4 illustrates an example of a WS-BPEL, deployed in a BPEL service engine within an ESB, describing the interaction between a consumer and multiple WS. The consumer is able to invoke the business process as if it were invoking a WS and starts by sending a request. The BPEL then processes the consumer's request by accomplishing the orchestrated interactions it describes and invoking various WSs itself. Finally, once fulfilled the request, the process finishes its execution and the response is returned to the consumer in one single request-reply interaction.

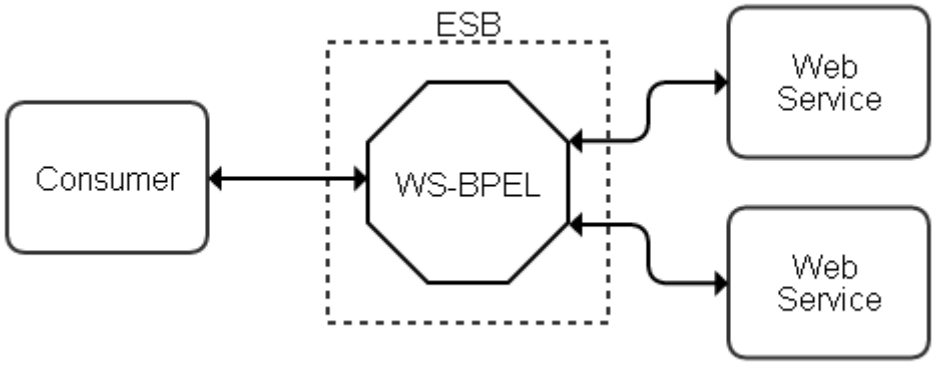

Figure 2.4 – The WS-BPEL, deployed in a BPEL service engine, allows the orchestration of WS interactions.

Another commonly used method of describing business processes is through the use of BPM, namely the BPMN (Owen & Raj, 2003). BPM is the activity of representing a business process in a way that it is easier to analyse and improve upon. A commonly used example of a BPM is the BPMN, which allows to represent business processes through a graphical and simplified presentation. It was designed with a focus on WSs and in order to facilitate the use and understanding of business processes while allowing to model and describe complex interactions at the same time. This is achieved by means of a Business Process Diagram (BPD) which

graphically describes the process flow in a simplified and easy to understand manner, while offering extensive modelling tools. By providing an intuitive notation, one of the main purposes of BPMN is to favour business users that lack technical skills while offering the depth and complexity of business process development to technical users, allowing for a wide range of users to understand and manage business processes.

Commonly speaking, one can look at BPM as a higher level language when compared with the BPEL, as the graphical capabilities of the BPMN adorn it with a visual dimension and allow to take advantage of a more generalised and simplified process flow, while the BPEL presents itself as a more raw and detailed description of each occurring interaction, since its standard fails to provide a general graphical notation, although several notations have been created by different developers. While both the WS-BPEL and the BPMN provide their own benefits and are commonly used in the industry, the lack of standards by the BPMN, as opposed to the BPEL which works closely with widely used standards, makes it difficult to be broadly accepted. For this reason, this master thesis work makes use of the WS-BPEL to research and accomplish its motivations.

Chapter 3

"Sustainability – the capacity to endure – implies, among other things, the discovery of new and lost enterprise capabilities in the network, and the adaption of knowledge, processes and interacting ES to the new circumstances." - (Taxen, 2012)

3 SUSTAINABILITY OF SYSTEMS AND INTEROPERABILITY

As competitive markets get more and more complex while promoting dynamic evolution, companies find that the traditional way of conducting business through their own individual efforts does not provide the necessary results and efficiency. Their success depends heavily on the activities they are able to perform with others and their ability to effectively cooperate (European Comission, 2008).

Thus, in order to achieve previously set business goals, enterprises must be interoperable and adaptable to the ever changing business environment. This means that, while attaining interoperability is essential for success and may prove hard to achieve, it is just as important and demanding that, once achieved, enterprises are able to sustain it so they can remain interoperable at all times. Maintaining interoperability however, may be easier said than done, as enterprises are subject to constant internal and external changes that may reflect themselves in the enterprise's capability to interoperate. The scope of these changes may range from the natural and dynamic evolution of the business environment, to the very own standards that form the basis for communication and network management (Noran, 2012). The enterprise's own lifecycle may also play a role in this changing behaviour, as the requirements and capability for an entity to interoperate may vary along time. These different aspects tend to promote a holistic view on interoperability, where all the relevant levels of an enterprise must be considered.

Hence, the establishment of a sustainable environment in which interoperability is assured in spite of the encompassed alterations is of utmost importance that, beyond the benefits of interoperability, brings forth a number of boons, such as a more robust and resilient system capable of answering and adapting itself to new discoveries and requirements.

3.1 INFORMATION MODELLING

As mentioned, in order for interoperability to be effective and sustainable, all the relevant levels within an enterprise must be taken into account. This includes data formats and models. In order to better understand the content that follows in this chapter, this section briefly introduces the concept of information modelling.

Information modelling is used as a medium to increase the productivity of software developers and is defined by computer-based symbol structures, such as items, entities, and relations, which are able to describe the meaning of information and knowledge about a system (Mylopoulos, 1998). This allows the developer to address complex problems while retaining the readability and understanding of the process being developed. As enterprises increase their use of IS, the complications of integration become ever more complex, raising the need to use and manage data efficiently. The use of information models enables to ease the process of dealing with complex systems by offering a higher level of abstraction during development, having become an important tool in modern organizations (Sundaram & Wolf, 2009). These models are nothing more than formal descriptions of the data artefacts being manipulated, such as object diagrams, interface definitions, XML schema, between others. When working with multiple and varied models, it often emerges the need to design transformations between them. These transformations are represented through the use of mappings, which allow to describe how two different models relate to each other (Bernstein, 2003).

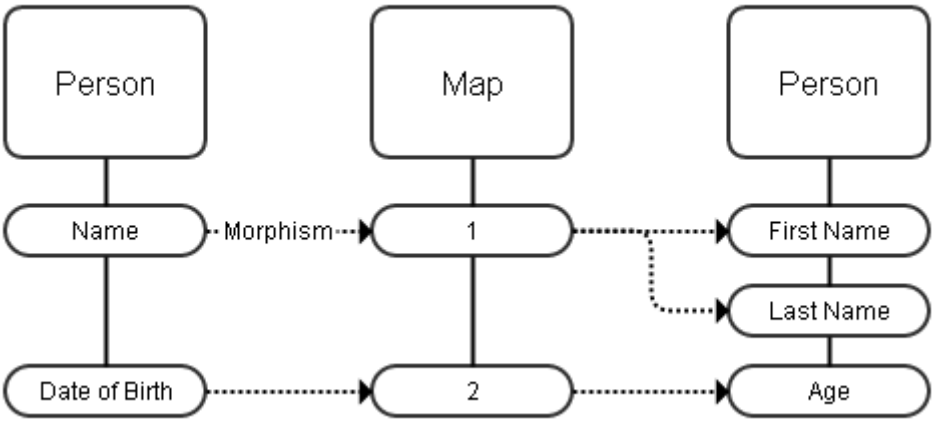

Figure 3.1 – Mapping example between two distinct models.

Figure 3.1 illustrates an example of a mapping between two different models and the way they relate to each other. As described by Bernstein, the binary relation (i.e. mapping,

transformation, etc.) between two information model specifications is defined as a *morphism*, and is depicted in the figure by the dashed lines. The image presents two different information structures in which both of them describe a person by means of various attributes. Because both specifications are distinct from each other, problems may occur if and when there comes a need to establish a relationship between them. For instance, in one of the models the person's name is defined by the sole attribute of *name*, while in the opposing model the person's name is not defined by one attribute alone, but by two different attributes, *first name* and *last name*. This means that a direct relationship between the two models cannot be established, since there is a need to map the different attributes accordingly. Hence, this is where the relevance of mappings comes into play. In order to establish a link between the two information structures, a map that describes the morphisms amongst the different attributes must be defined, allowing in this manner to correctly establish a connection between them in both information models. Often, in order to define these mappings it is necessary to seek specific knowledge that indicates whether they are complete or if further elements must be tackled. However, when a mapping is established between distinct models some inconsistencies of information may occur that can result in semantic conflicts. Semantic is defined by the IEEE as relationships of symbols or groups of symbols to their meanings in a given language (IEEE Computer Society, 1990). These information conflicts are described as semantic mismatches and can be classified as either *lossless* or *lossy*. In *lossless* mismatches no relevant information is lost and the newly related element can fully capture the semantics of the original one. In *lossy* mismatches however such is not the case, and a mapping preserving the original element semantics cannot be constructed (Agostinho, Malo, Jardim-Goncalves, & Steiger, 2007).

The subject of information modelling plays an important role when it comes to system interoperability, since it represents a required piece in the struggle to achieve and maintain interoperable environments. Models and mappings allow to ease the process of data integration and provide an additional layer of abstraction that effectively contribute to the sustainability of systems. Such an effect can be observed in Model-Driven Engineering (MDE), sometimes referred to as Model-Driven Development (MDD), a practice for developing model driven applications that seek to address a system's complexity by simplifying and formalizing the system's several activities and tasks, allowing to describe complex problems in a simplified manner while increasing general productivity by avoiding to dwell in specific technical details (Schmidt, 2006). As MDE works primarily with models, which are easier to specify and understand when compared to other programming languages, it enables to extend the range of development to several domains beyond that of computer specialists, allowing for a simplified development and maximizing compatibility between systems.

3.2 SUSTAINABLE INTEROPERABILITY

As business requirements become increasingly more complex, several areas of expertise are required to be integrated with one another in order to achieve collaborative development while taking into account both the business and technical requirements of all the involved entities. Moreover, as systems become more and more dynamic, adapting to market and environment changes, different costumer requirements and others, it leads to a constant and unpredictable evolution and fluctuation, making interoperability hard to maintain (Agostinho & Jardim-Goncalves, 2009). Hence, today's EI is a combination of multiple dimensions and multidisciplinary issues which need to be tackled efficiently using a more systemic manner, one that is able to provide sustainability while considering the capability for dynamicity. On the domain of services and web technology, the relevant issue is not how to establish interoperability, which exists by definition, but rather how to maintain it and ensure that the system remains interoperable at all the relevant levels and flexible to new modifications.

The concept of sustainable interoperability categorises ES and networks that focus on maintaining interoperability along the adaptive system's lifecycle, as depicted in Figure 3.2. As the system operates, the network is monitored for possible occurring changes. If and when new requirements are introduced, an evaluation that seeks to decide their feasibility is performed before proceeding into the design, implementation, and verification phases.

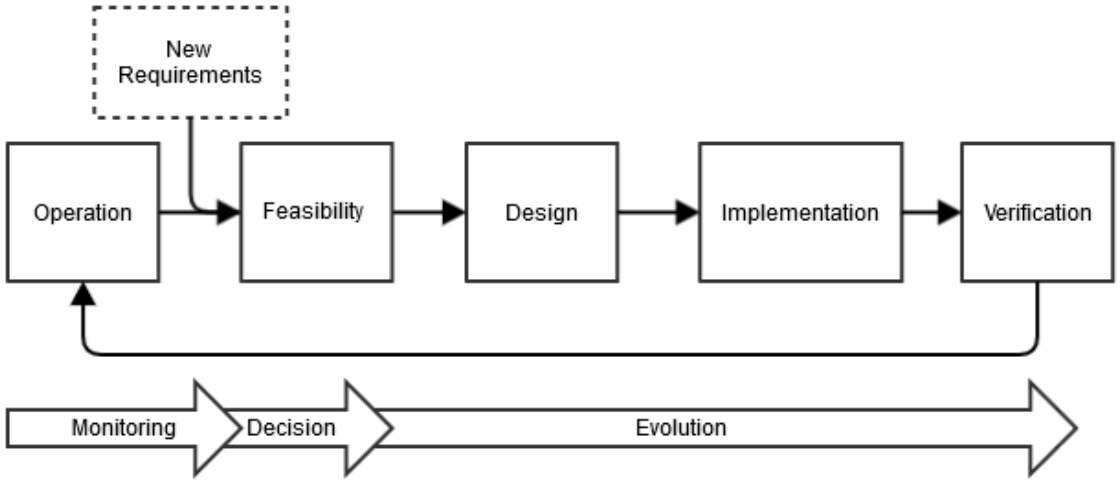

Figure 3.2 – Adaptive system's lifecycle. (Adapted from (Agostinho, 2012))

Depending on the nature and progress of evolution, the related entities can either continue interoperable with their partners, lose integration with some of them, or break and invalidate the entirety of the network. Hence, if the system's changeability and evolution is not carefully

monitored and addressed, the whole network faces the possibility of disruption and its collaborative capability is put at risk.

Thus, when dealing with interoperable environments one must take into consideration several problems that are prone to occur and that may in some direct or indirect way, harm the system's overall stability. These may include language incompatibilities, which may inadvertently cause semantic misunderstandings and data models mismatches, different legacy systems, with different characteristics and thus difficult to integrate with each other, communication issues and failures, etc. These and other factors constitute a set of different and multiple issues that may contribute to disarray the system's overall stability and disrupt interoperability. Such events can be named as harmonisation breaking events and are discussed in more detail in chapter 4.

3.2.1 Model-based Interoperability

In order to attain sustainability, it is important that the grounds for it are firmly established when initially working to achieve interoperability. This ensures that in the event of future harmonisation disruptions the system is ready to promptly respond to the environment changes and avoid breaking interoperation. To that effect, it is required to have mechanisms that enable organizations to abstract themselves from the technology in order to attain a common and shared vision regarding the IS. As seen before, data modelling permits to achieve this level of abstraction by representing knowledge in a uniform and formalized way, allowing organisations to avoid technical details and complications and provide a greater focus to business planning and management. In (Agostinho, 2012), Agostinho proposes a modelling language independent framework using a Model-Driven Architecture (MDA) that seeks to establish interoperability while laying down the foundations for sustainability at the same time. MDA is an approach to use models in software development based on MDE principles, which primary goals are portability, interoperability and reusability through architectural separation of concerns (OMG, 2003). Hence, by use of information modelling, the proposed framework allows to provide organisations with an additional level of abstraction, enabling them to become independent from the language their models are described in and removing the need to adjust each organisation's internal structure and system. This is achieved by means of an interface applied to the organisation's output models that serves as a modelling language translator capable of generating formalized and Language Independent Models (LIM). These LIM's enable organisations to refrain themselves from the hassle of needing to relate their models with the modelling language specifications of other entities, placing the focus on the framework's interface layer instead, where they are further worked on and adjusted before being stored in a dedicated knowledge base. Once the final models are obtained and integrated within the framework, they can then be exported to any of the

connected entities. In this manner interoperability is established uniformly among the various organisations without the need to delve into specific technical details, while providing the foundations for a good sustainability simultaneously.

3.2.2 The Complex Adaptive Systems (CAS) Framework

In (Agostinho & Jardim-Goncalves, 2009), a CAS based framework is proposed to support the sustainability of interoperability. CAS represent complex dynamic and evolving systems capable of autonomously adapting themselves to the surrounding environment, without the management or control of a specific entity (Holland, 1992). Thus, building on the existing CAS concept, the former authors proposed a framework that aims to contribute to the sustainability of interoperability by conceiving a sustainability recovery cycle along the adaptive organisation's lifecycle, as mentioned previously, focusing on monitoring and decision capabilities in an effort to ensure a rapid response to harmonisation breaking events. As acknowledged by the authors the framework, described as CAS to support Sustainable Interoperability Framework (CAS-SIF) and displayed in Figure 3.3, must exhibit the following competences:

- *Discovery capabilities* (1) – that is being able to timely detect environment changes that may lead to harmonisation disruption;
- *Learning and adaptability* (2) – once a break in the system's harmonisation is detected, a learning process should be triggered in order to gather knowledge about the occurring changes, enabling thereof the adaptation of systems and the optimisation of the maintenance process;
- *Transient simulation and decision* (3) – the ability to perceive how the network will alter itself during the transient period that results from the previous break in harmonisation, and how future adaptations, if they are implemented, will affect the overall system's capability and behaviour;
- *Notification and communication* (4) – once all the possible outcomes have been analysed and the necessary measures considered, the various nodes along the network should be informed and instructed on how to react so they can properly perform their own adaptations and evolve back into an interoperable state as smoothly as possible.

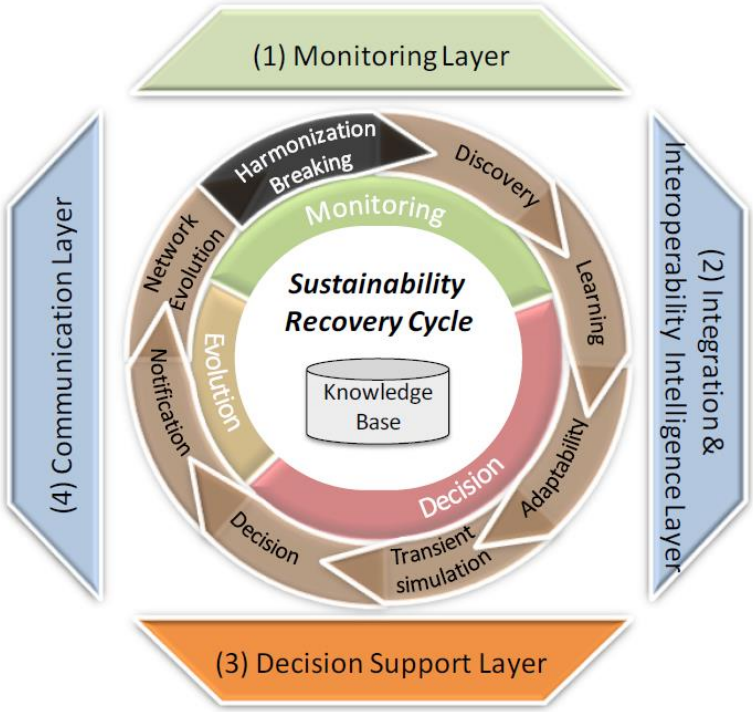

Figure 3.3 – CAS-based framework to support sustainable interoperability (CAS-SIF). (Agostinho, 2012)

The CAS-SIF enables a system to monitor and detect unexpected changes while providing the ability to discover and recognise harmonisation breaking events that may result as a consequence from those environment modifications. The framework then uses an adaptive intelligence to gather information about the occurring problems and evaluates how the network should adapt to cope with the system's evolution. Once an analysis has been performed and a decision has been made regarding the required adjustments, the framework then notifies the different network nodes and properly instructs them on how they should react in order to avoid interoperability disruption.

The work presented in this master thesis focuses mainly on the first and second layers, namely the monitoring layer and the integration and interoperability intelligence layer, as it seeks to contribute to the system's monitoring capability during SOA-based communications, and to provide a methodology that enables a better understanding of the issue while providing the system's manager with the needed intelligence to evaluate and approach the harmonisation breaking event, alongside a possible autonomous response to the latter.

3.3 Monitoring SOA-based Enterprise Networks

While many enterprises may possess important core skills relevant to the market in which they operate, few of them possess the complete set of requirements to effectively compete in such

markets (Thoben & Jagdev, 2001). Therefore, the need to collaborate emerges, allowing for a single company to augment its attributes to those of the network as a whole (Camarinha-Matos & Afsarmanesh, 2005).

Collaborative networks, such as DMNs, are formed by several companies in a chain-like model with the purpose of achieving a specific goal. Such networks constitute very complex systems where management is a difficult task and where the rapid reaction to market demands is essential (Papakostas, Efthymiou, Georgoulias, & Chryssolouris, 2012), since the entire network cannot afford to be stalled due to a troubling partner, risking an impact on product manufacturing, cost, quality, and others. This type of network needs to be flexible with good monitoring techniques in practise in order to guarantee network effectiveness.

Being able to monitor services is an important requirement to effective service management. A single business process may comprise several different components such as orchestration engines, legacy middleware, application servers, and others, making the governance of such processes a complex task that requires an efficient and consistent monitoring system. Hence, several types of monitoring frameworks and techniques have been developed and proposed for SOA-based systems, as illustrated along this section.

Baresi et al. in (Baresi, Guinea, & Plebani, 2006) propose a monitoring framework for WS-BPEL processes that aims to extend the basic features regarding the capability of WSs monitoring, in order to improve both functional correctness and QoS. The proposed approach gives special relevance to client-side monitoring and relies on WS-Policy (W3C, 2007), a standard to define WS requirements and to express the monitoring policies associated with WS-BPEL processes. It can express both functional aspects, such as constraints on exchanged data, and non-functional aspects, such as security or message reliability. By means of a *Monitoring Manager*, it becomes possible to extend existing service platforms with monitoring capabilities. When a given process needs to invoke a service, the *Monitoring Manager* is invoked in its place, which receives the data to be sent and the required information regarding the service invocation. In this manner, the SOAP message to be sent is treated in the respective module where it can be properly analysed and tested for missing data or required encryption rules. This monitoring technique allows for a greater control over the exchanged information contributing to a more consistent and reliable system.

In (Psiuk, Bujok, & Zielinski, 2012), Psiuk et al. suggest a generic ESB framework to strengthen the monitoring of Java Business Integration (JBI) (Ten-Hove & Walker, 2005), an ESB standardisation, being applicable in practice to the whole class of ESB implementations. Through the use of a *Monitoring Goal Meta-model*, which encompasses important ESB entities metric and mechanisms, the framework allows for on-demand selective monitoring, which means

that the process description, structured by the meta-model, is able to specify what should be monitored, when should the monitoring take place, and under which circumstances. The framework, which encompasses the entirety of the ESB communication, is able to gather information regarding the ESB topology as well as its messaging service, measuring activities, tracking changes, and performing autonomous error diagnosis, increasing in this manner the framework usability as the human operator is exempt from the need to analyse and correlate monitoring results.

The Adaptable Service Bus (ASB) suggested in (Chen, Ni, & Lin, 2008) by Chen et al. is a proposition for an enhanced ESB that enables the alteration of business rules at runtime in order to avoid costly shutdowns in billing applications. While it shares many of the conventional ESB characteristics, by maintaining service endpoints, operation names, and parameter values in external storages, the ASB allows to modify the behaviour of the application processes at runtime as developers are able to adjust the information values saved externally. This technique allows for dynamic service composition with manual adaptation support. Whilst requiring human intervention in order to adapt to occurring changes, the ASB in conjunction with active monitoring, contributes to the maintenance of a stronger and more reliable network with much shorter downtimes amid interoperability problems.

Following on the CAS-SIF concept introduced in section 3.2.2, Ferreira et al. in (Ferreira, Agostinho, Sarraipa, & Jardim-Goncalves, 2011) proposed a monitoring technique based on MAS where cooperating agents are capable of communicating and interacting with each other or with the environment, in order to increase collaboration effectiveness. The proposed technique, described as Monitoring morphIsms to suppoRt sustAinable Interoperability of ES (MIRAI), consists of a distributed framework capable of monitoring the existing mappings of a given company, which relate data elements between two different models as seen in section 3.1, detecting occurring changes in the environment and proposing solutions and re-adaptations in order to maintain interoperability. This is accomplished with the help of several decentralised agents divided along four main blocks, each one responsible for a given task: the *Intelligent Supervisor* block, responsible for the detection of harmonisation problems in the network and the discovery of possible solutions, the *External Communicator* block, responsible for the interaction with other MIRAI's in the network, the *Administrator* block, responsible for the interaction between the user and the system each time a decision concerning possible adaptations must be made, and the *Life Cycle Monitor* block, responsible for agent maintenance. Although, it suffers from the need of having those mappings formalized and stored along with the cooperating software systems, this framework can prove itself useful to SOA-based networks due to the automation and stability it provides.

Another monitoring technique known as MONitoring Agents using a Large Integrated Services Architecture (MonALISA) is proposed by Newman et al. in (Newman, Legrand, Galvez, Voicu, & Cirstoiu, 2003). MonALISA is based on dynamic distributed service architecture and provides complete monitoring, control, and global optimization services for complex systems. Similar to the MIRAI, the MonALISA provides a multi-threaded monitoring service based on MAS working in a collaborative environment and being able to track and monitor the network in real time. This allows for the monitoring service to be deployed using independent threads, providing a greater reliability in case of system failures.

Other relevant works have been published, such as (Ou, Sun, Guo, & Li, 2008), which discusses the potential for virtual process monitoring and introduces a business user oriented virtual business process monitoring model, or (Wang, et al., 2009), which introduces an online monitoring approach for WS requirements by making use of specified constraints that allow to process and analyse specific events.

Software monitoring consists in gathering relevant information concerning the state, behaviour, and environment of a given software system, in order to discover and identify potential system breaking events as timely as possible. Monitoring is usually performed alongside the system's normal execution without influencing or interrupting its operation. As such, being monitoring an important part of a systems performance, it is relevant that an effective monitoring is implemented when working with services and complex networks. The techniques and frameworks described in this section present a few of the many contributions made in the domain of service and network monitoring. Although there are several works concerning this specific subject, some issues still call for the need of further investigation, such as issues concerning information extraction or monitoring costs (Wang, et al., 2009). In regard to the capability of systems to autonomously react to harmonisation breaking events, there is also a lack in existing solutions and feasible approaches. Hence, the work performed during this master thesis and presented in this document, does not intend to close a gaping hole concerning SOA-based monitoring, but rather contribute to the ongoing exploration of such relevant matters.

Chapter 4

"Governance of complex business processes requires a monitoring system as a starting point for the enforcement of relevant management decisions. (…) Quality of Experience (QoE) and QoS cannot be ensured without sophisticated monitoring of interactions between consumers and providers." - (Psiuk, Bujok, & Zielinski, 2012)

4 Methodology for Detection of Harmonisation Breaking in Service Environments

In the previous chapters, the concept of interoperability along with its relevance in service environments was introduced. Several frameworks and different technologies aiming to promote interoperable environments were briefly discussed and the fundamental means to achieve and sustain interoperability were presented. Once interoperability is achieved, that is once all network entities are able to communicate with and understand each other, organisations are able to exchange information along a stable environment that follows pre-established laws. However, as markets tend to adapt to new requirements and demands, an evolutionary behaviour is triggered giving space to interoperability problems that may cause harm to the system's overall stability. Such problems may include language incompatibilities, which may inadvertently cause semantic misunderstandings and data models mismatches, different legacy systems, with different characteristics and thus difficult to integrate with each other, communication issues and failures, and other factors. These harmonisation breaking events disrupt the sustainability of interoperability and raise the need to develop effective monitoring activities capable of detecting unexpected behaviour in communications between cooperating companies. Several different approaches to monitoring frameworks and techniques are discussed in section 3.3 and present various possible methods to monitor networks, while paving the way to tackle interoperability problems that may occur out of the mentioned events.

This chapter aims to further explore the occurrence of harmonisation breaks in interoperable environments as well as different types of approaches in order to monitor and detect

such events. Finally, the grand contribution of this master thesis work is presented: a proposal for a methodological framework for the detection and possible system recuperation of harmonisation breaking events in service environments.

4.1 HARMONISATION BREAKING

When dealing with science, certain results that are attained out of experimentation can be accomplished even when working under different circumstances. Hence, one can therefore conclude that the laws that govern such outcomes behave in the same manner for similar variables and inputs. However, experimentation has shown that unexpected events may cause a system to experience unwanted behaviour. This disorder, described in (Nicolis & Prigogine, 1989) as symmetry breaking, is illustrated in Figure 4.1, which exemplifies the behaviour of an initially stable system and its possible outcomes in response to a given input or occurrence.

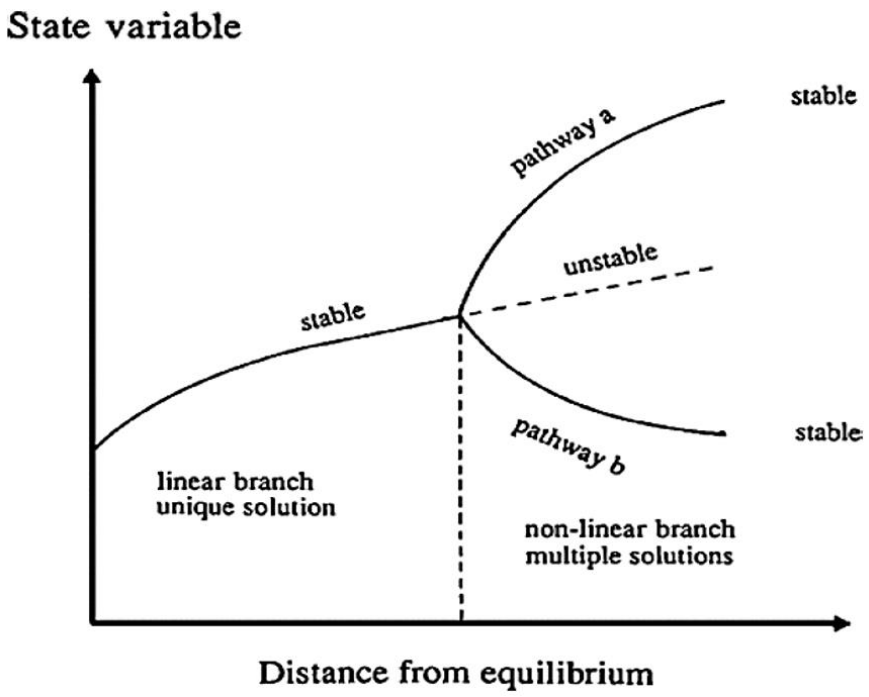

Figure 4.1 – Symmetry Breaking. (Adapted from (Nicolis & Prigogine, 1989))

The same theory can be applied to IS. As mentioned, small fluctuations acting on a system, such as a partner becoming unresponsive or introducing corrupt data into the network, may cause it to cross a critical point and evidence unexpected behaviour. This disorder is designated as symmetry breaking and is a characteristic of complex behaviour (Couture, 2007). Harmonisation breaking is the parallel concept for enterprise networks (Agostinho & Jardim-Goncalves, 2009).

Once interoperability is achieved, organisations within a network interact with each other according to a pre-established set of laws. However, in the case of an unexpected event occurring, such as an organisation adapting to external factors, the system's harmony may be disrupted and stability may be lost, potentially leading to further interoperability problems. Thus, a harmonisation breaking event can be described as an unexpected or unwanted alteration occurring in an initially stable system that leads to a change in its normal behaviour and, consequently, provokes a disturbance in the system's harmony and stability. Such events are therefore the cause of interoperability problems and may cause significant setbacks in large and complex networks. Hence, the detection of such events and their possible prevention or, if necessary, system's recuperation and adaptation, should be a priority when attempting to work towards sustainable interoperability.

4.2 Detecting Harmonisation Breaking

As mentioned before, in order to prevent harmonisation breaking events that may lead to interoperability disruption, a feasible and effective monitoring architecture must be implemented. This allows to observe the system behaviour at all times, record relevant results and information, and attempt to perceive possible system fluctuations that may evolve into bigger problems. To timely detect harmonisation breaking events is crucial when trying to prevent them from harming the system's harmony and overall stability.

For the purpose of this work, and according to the monitoring techniques previously presented, three main types of monitoring were identified and are described along this section. One should note that these are not exclusive but rather present some critical factors that are more prone to show signs of instability and that should be taken into account, in order to effectively monitor the system's behaviour.

4.2.1 Information Streaming Monitoring

An important characteristic when it comes to network interoperability is the stability and reliability of direct and real-time communications – or the fluid stream of information – between cooperating companies. By constantly monitoring the direct communication between two collaborating entities, one may be able to timely detect communication problems and promptly react and seek other alternatives, which is particularly relevant in case of communication failures or general network problems. Several monitoring techniques can be applied in order to measure and monitor direct real-time communications, from the traditional PING, a tool used to measure the delay of a given internet connection and test the reachability between a client and a target host, to more advance methods such as packet tracing and monitoring, which emphasises on

transmitted, lost, or erroneous packets during a real-time connection (US Patent No. US 7583613 B2, 2009). These methods allow to measure the quality and reliability of communications as well as to recognise possible communication failures due to packet loss, while compiling detailed statistics and registering relevant connection values and signs of degradation. Hence, when aiming towards sustained interoperability, the monitoring of information streaming presents a relevant factor that should always be taken in consideration and comprised in the global network monitoring plan. During this master thesis scientific developments, information streaming monitoring is performed at a higher level during business process execution, in which the process attempts to reach the target service and fires an exception in case of failure. The exception is then caught and properly handled in real-time. Lower level techniques, such as packet tracing and monitoring, were out of the scope of this work and were therefore not approached.

4.2.2 Message Packages Monitoring

As important as monitoring the flux of information streaming between two cooperating companies, is the monitoring of the information that is actually being exchanged between them, whether it is the message specification or message content. To such effect, it is important that both entities are able to understand each other continuously and, in case of misunderstanding, that both of them are able to recover from it. In order to effectively perform this type of monitoring it is required not only a previous knowledge about the data formats and specifications that circulate on the network between entities, but also the rights to access the respective content, which to some may prove to be more of an ethical problem as it may raise privacy concerns. However, monitoring the message packages that run around the network has great potential in terms of sustaining interoperability, as it may allow to prevent harmonisation breaking in case of a corrupt or invalid message being detected, enabling the rapid alert of the entity responsible for corrupting the data in order to prevent future problems. The ESB itself enables access to message payloads as long as they are in fact accessible and unencrypted, allowing therefore to extract their content and further send it to other components in order to validate it.

4.2.3 Information Structure Mappings Monitoring

Finally, and complementary to the previous, we have the monitoring of information structure mappings. When attempting to detect harmonisation breaking in a stable and interoperable network, one useful approach consists on monitoring the information structure mappings of the members of that network. Given two different information models, mappings relate the data elements between the source and the target model, as explained in section 3.1. In the case they are duly formalized and stored, knowledge reasoning techniques enable to implement monitoring routines capable of checking the correctness of the information exchanged

among partnering organisations. Hence, each individual member of the collaborative network benefits with the ability to detect changes in the partner's information model structure, as they become able to detect occurring conflicts and immediately respond to them as they occur, further enabling them to propose possible solutions and re-adaptations in an effort to maintain network interoperability (Ferreira, Agostinho, Sarraipa, & Jardim-Goncalves, 2012). To such effect, the MIRAI framework described previously (Ferreira, Agostinho, Sarraipa, & Jardim-Goncalves, 2011), is an effective tool since it takes particular focus on monitoring and validating the existing mappings of existing companies. The solution here proposed can thus be further enhanced by integrating the MIRAI within its architecture and expand the range of application of both solutions, a motivating approach that could be considered for future work.

4.3 Approaching Harmonisation Breaking

As seen, harmonisation breaking events consist of unexpected and unpredictable events that may disrupt system's harmony and hinder its global productivity. Thus, in order to properly approach them, these events must be discovered as soon as possible to minimize interoperability problems and significant setbacks, especially when it concerns to industrial service-based networks. To achieve this goal, proper network monitoring is relevant, as already discussed. However, for the monitoring to be effective it is important for its object of focus to be easily accessible and understandable. This is where information modelling plays an important role. When dealing with large projects, companies generally use hard-copy or electronic files to share the project's technical specifications and design among its users, developers and collaborators. This document based approach comprises the system's requirements and information that engineers must control in order to guarantee the project's validity and consistency (Ogren, 2000). However, due to the inherent complexity of the process and the network itself, this method presents several limitations that may in some way contribute to slow product development. Hence, to overcome these limitations and optimise network engineering activities, the Model-Based Systems Engineering (MSBE) strategy was introduced. As defined in (Kossiakoff, Sweet, Seymour, & Biemer, 2011), the function of Systems Engineering (SE) is to guide the engineering of complex systems, in which the word *complex* restricts it to systems with diverse elements and intricate relationships. It focuses on the system as a whole, both internal and external factors, and is responsible for the entire concept development, guiding and coordinating the design of each individual element in order to guarantee compatible interactions. With the emergence of formal modelling languages, such as the Unified Modelling Language (UML) (Pooley & King, 1999), and MDA, discussed previously in section 3.2.1, systems engineers gained the ability to represent system requirements and behaviours with increased consistency and according to a set of principles, concepts, and model definitions. In this sense, in the attempt to integrate software and

SE processes and principles, MDD, introduced in section 3.1, was used and applied to systems development, which converged to the MBSE solution. In essence, MBSE consists in the development of a system's model from its initial lifecycle phase, in which the models are mostly used for decision making, until its full development, when the models can be used for design and transformed into the build-to baseline. Unlike the traditional SE method, in which the product and respective specifications are described by means of static representations in the form of written documents, with MBSE models are used to perform this description, allowing for a more flexible and evolving process. Hence, MBSE is the formalised application of modelling to support cooperative engineering processes, namely system requirements and design, analysis, verification, and validation activities starting at the conceptual design phase and continuing throughout development and later lifecycle stages (Wymore, 1993).

Figure 4.2 represents a CAS-like IS adaptive lifecycle, following the concepts introduced in section 3.2, which attempts to illustrate the important role of SE and how it seamlessly integrates into the system's lifecycle.

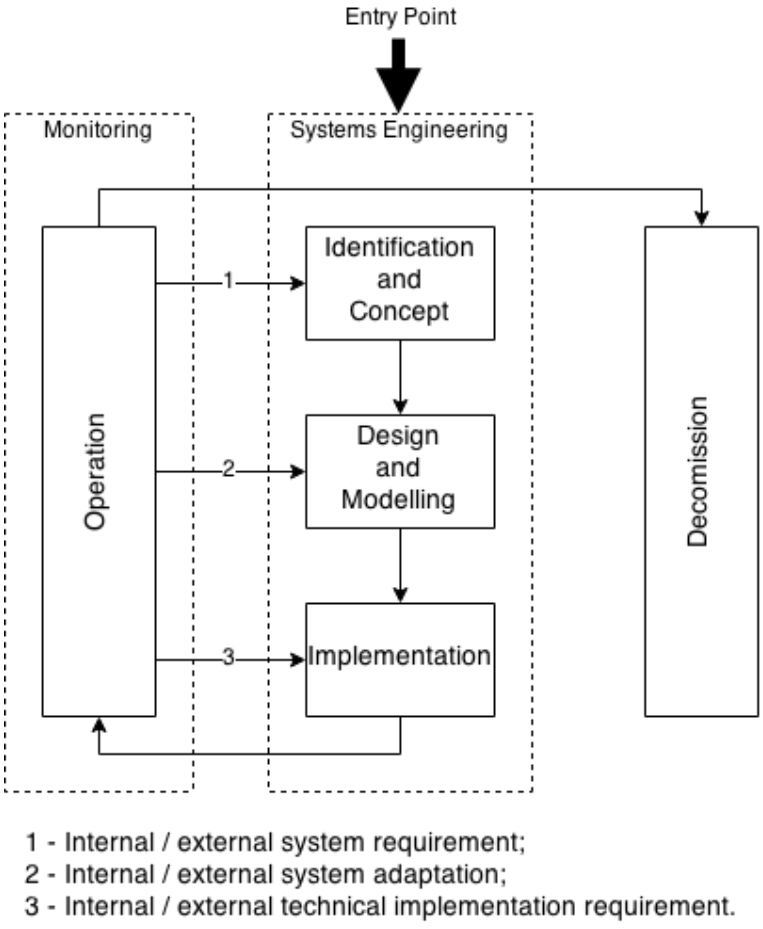

Figure 4.2 – Adaptive IS lifecycle.

It presents the fundamental steps that take place prior to normal system operation. The entry point is triggered by a traditional engineering process that goes through the concept specification; requirements analysis and preliminary design, that following the previous discussion should include some degree of formal modelling; and finally, a technical implementation that concludes with the deployment of executable code. As the system operates in an assumable interoperable state, the network is monitored for possible changes that can occur in one of three ways:

1. New requirements that add upon the initial specified behaviour. In such case, a process of system engineering takes place, leading to a need to decide on design and implementation questions and paving the path to system's reengineering. The new requirements may be exclusively motivated by internal needs of growth or can be derived by a request of one of the external partners. In any case, a process of identification and concept construction is performed, establishing the necessary foundations before moving forward into the design and modelling phases;

2. Requirement concerning an already existing concept. If an adaptation/change occurs driven by the re-specification of an existing concept or relationship, then the system moves straight into design and modelling phase, as the concept concerning the adaptation already exists and there is no need to rethink it. Automatic adaptation may be possible when it is externally motivated;

3. Requirement introduced due to an erroneous internal implementation (mistake that needs to be corrected) or motivated by means of external factors that allow the system to adapt using the same concepts. In this case, it proceeds straight into the implementation phase.

Once the system detects new changes and adapts accordingly, and after all the necessary steps have been taken and implementation has taken place, normal operation is resumed. In this manner, self-organisation and self-adaption capabilities are achieved through the adaptive feedback loop that triggers the lifecycle once again after new input information has been introduced, occurring naturally as the system evolves. However, during the time that the system seeks to implement the required changes in order to adapt to the new requirements, the network inevitably experiences a state of non-equilibrium in which interoperability is not fully present. This oscillation among a state of network equilibrium, where organisations are operating normally, and non-equilibrium, characterized by non-linear behaviour, is designated as a state of quasi-equilibrium, as evidenced by Figure 4.3. From a workflow point of view, the adaptive system's lifecycle has two different entry points. It can either be through a change in the environment (1) that leads the organisation to an unexpected adaptation (2), or through an internal change at system level (3), derived (or not) from the actions of (2), that creates positive feedback

into the environment causing the remaining networked systems to react accordingly (4), in which case the adaptation of the network as a whole is achieved by several sub-self-organisation processes triggered at system level after an interoperability harmonisation breaking detection.

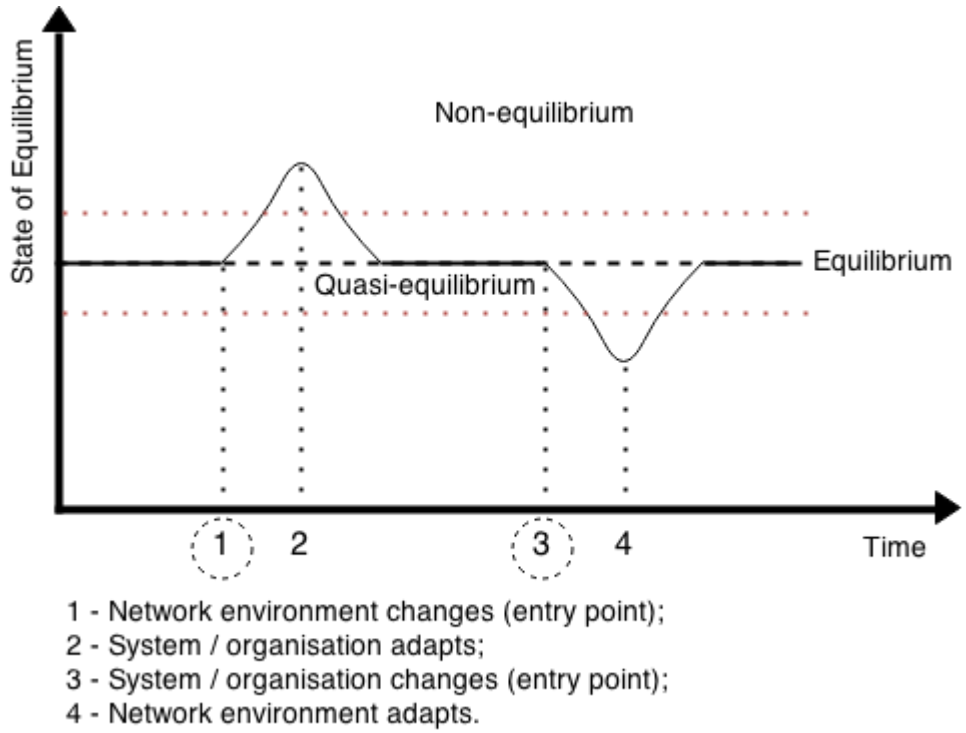

Figure 4.3 – Adaptive system's equilibrium and perturbations.

Therefore, by reducing the transitory phase between these different equilibrium states at an organisation level, it is possible to achieve a greater range of efficiency from the interoperable relationships, and maximise the time networks are operating linearly. Thus, with this knowledge, one can conclude that the process of tackling harmonisation breaking events in IS networks can be significantly improved when a firm foundation is established during the initial SE phases that take place during project development, in which the concept is firstly identified and built upon. This can be achieved by building a good modelling structure that allows not only to explicitly model the concept but also the different IS entities, their relationships, and semantic knowledge, which will then facilitate eventual future adaptations and reengineering work. In regard to the research performed during this master thesis, which is mainly concerned with the event of interoperability disruption in service-based networks, service modelling – the activity of designing and specifying service processes and systems – and service orchestration – the logical structuring and coordination of multiple services – which are both detailed in chapter 2, are used as a means to properly construct business processes and, in this manner, achieve the design and

modelling purpose that concerns the SE process during the initial development stage of a new system, thus increasing its flexibility and adaptability.

4.3.1 Towards a Methodological Solution

Before introducing the proposed methodology that seeks to approach harmonisation breaking in service-based networks, a hypothetical scenario is discussed along this section in order to facilitate and ease future understanding. The discussion is centred on a scenario where a consumer (C) wishes to exchange information with a provider (P). Before communication can be accomplished and the exchange of data take place, the process that describes the interaction between the two cooperating entities must be described. This consists in specifying every single interaction and procedure that occurs between both and structure them accordingly. As mentioned before, this is achieved by means of service modelling and orchestration and results in a working business process that, after being deployed in the proper server engine, performs the described tasks and enables the interaction between both companies. Naturally, because it is C who wishes to access and consume the services made available by P, the business process must be deployed within its control and management, assuming that a centralised entity responsible for such tasks does not exist. Thus during this time, and supposing that no other issues occurred, the business process operates normally in a stable environment allowing the exchange of information between the related entities in an interoperable way, as illustrated in Figure 4.4.

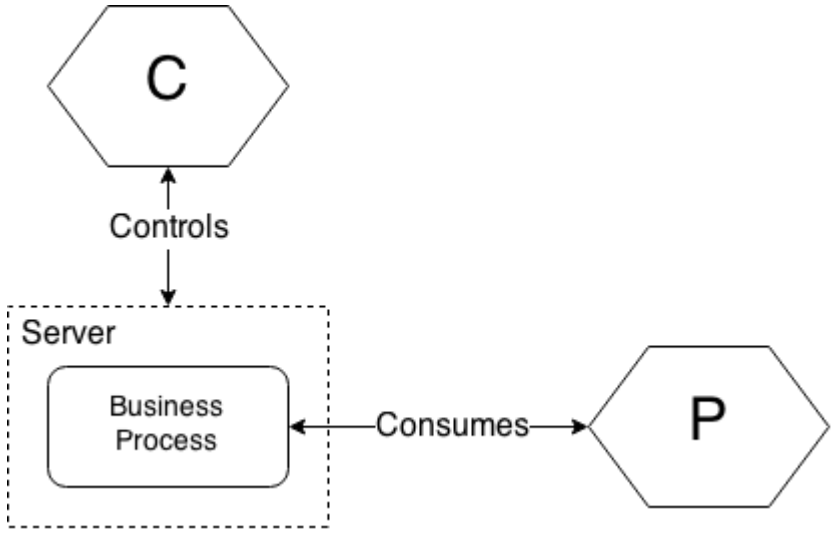

Figure 4.4 – Example of a Consumer (C) interacting with a Provider (P).

However, if for some given reason the communication between C and P were to fail, an exception would raise within the business process in consequence of its inability to reach its

destination. Generally, assuming that it is possible to catch and handle this exception, then it is also possible to identify the disrupting issue and take the proper steps to notify C about the inability to reach P, while alerting for the eventual need of human intervention. Nevertheless, in large networks such as a DMN, where several consumers and providers communicate simultaneously and depend on each other to achieve a common goal, this type of solution could temporarily hinder the system's production capability, as part of the system would be stalled until a proper solution were to be implemented and the connection to P restored. Therefore, during this transitory phase in which interoperability does not exist and needs to be reinstated, the system would unavoidably suffer setbacks.

While the inability to communicate with P could be due to some unsolvable situation, from C's point of view, such as a physically broken cable or other similar condition, it could also be as simple as an incorrect mapping during the invocation of a given WS due to an unexpected change in the service's interface. Thus, in order to properly evaluate the cause of interoperability disruption, this thesis suggests that by incorporating into the system an external intelligent component, which were to be invoked once an exception were to be raised within the business process itself, it would be possible to assess and analyse the situation from an exterior standpoint while also enabling the capability to act and alter the faulty business process. In this sense, a methodological approach was conceived in order to properly tackle harmonisation breaking events that may arise as a consequence of similar scenarios, and in this manner contribute to reduce the transitory phase that naturally occurs between the state of network instability and interoperability.

4.3.2 Methodology

Following the scenario introduced previously, this section introduces a methodology that seeks to approach the issue of harmonisation breaking events in interoperable service-based networks. The main objective is to design a methodological approach that can effectively reduce the time required to establish interoperability after a harmony disruption has occurred, by resorting to automatic interventions that directly act on the faulty procedures and in this way avoid the need for human intervention. The process is centred on the capability to properly handle exceptions that occur during normal business process operation and subsequently invoke an external intelligent component that enables to directly approach the cause of error. Thus, it was considered that if such an implementation could be achieved then three different outcomes would be possible, as illustrated in Figure 4.5.

As already described, the consumer (C) controls the server where the business process that runs the interaction between C and the services provider (P) is deployed (1). If a harmonisation breaking event occurs when consuming the services provided by P (2), then an exception within

the business process is raised and handled by its own error handling capabilities, after which it proceeds to invoke an external intelligent component (3) whose objective is to evaluate the cause of the issue and correct it by adapting the business process according to the newly introduced requirements (4).

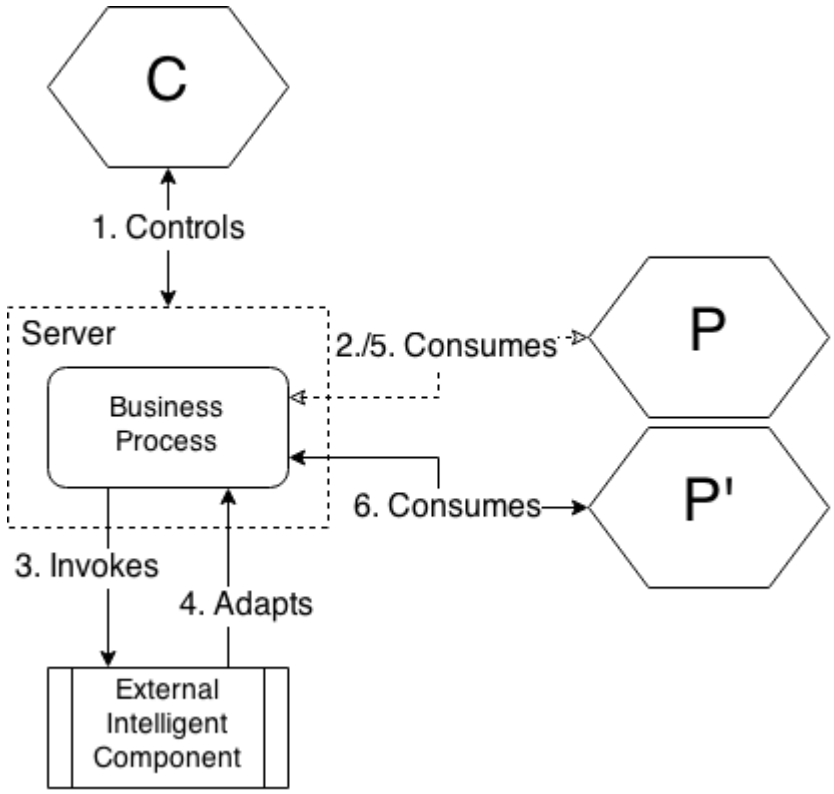

Figure 4.5 – Methodological steps to approach harmonisation breaking.

In a primary intervention, the external intelligent component would proceed to analyse the situation and attempt to restore the connection to P by directly modifying and automatically re-deploying the business process in the respective server engine (5). If there were no success and a process adaptation could not be implemented, then a secondary intervention would be attempted by searching the network and trying to connect to a similar provider (P') with the same characteristics and services as P (6). If an alternative provider were to be found, this solution would also require the business process to be adapted in order to properly reflect the necessary changes and start interacting with P' and disregard the previous service provider P. It should be noted however, that this particular step can only be achieved in very specific situations. To be successful, the network must be prepared for such modifications since its initial construction and design. Furthermore, not all domains can be considered. A service provider that supplies consumers with weather information for instance, represents a service somewhat simpler to

replicate and seek for in case there's a need to find an alternative, since the information it provides is rather generic. In the domain of manufacturing however, this approach is more challenging to implement since each provider in the network usually supplies consumers with very specific and customized information.

Finally, if none of the two interventions mentioned could be accomplished successfully, then no automatic solutions would be possible and a request for human intervention would inevitably be required. With this in mind, a methodology was devised to reflect the necessary steps to be implemented in dynamic service-based networks when seeking to sustain interoperability, as depicted in Figure 4.6.

The methodology starts by delineating a procedure of SE in which the concept of the process to be developed must be properly identified and modelled which, in respect to the subject of this research, implies the modelling of data, services, and others. Service orchestration is what allows to take these modelled concepts and structure them in such a way that it enables the establishment of interoperability between different companies. It is achieved through the formulation of a business process that regulates the tasks and interactions that occur during the exchange of information between collaborating entities. The relation between the business processes across different companies constitutes a virtual business process to the network that represents the interactions that take place from company to company. Although required for the methodological approach here proposed, these SE concerned steps are not obligatory for every company. However, only the companies that choose to implement them will be able to sustain interoperability according to the proposal here described, which means that if a company that has implemented these methodological steps is communicating with a partner that hasn't, the former will still be able to activate the recovery process as described. Although still valid, this can increase the difficulty to restore interoperability as one of the companies cannot adapt to the new requirements, and thus result in a possible setback to the overall recovery process or even the automatic methodological exclusion of the rogue entity.

Once the SE procedure has taken place, the developed business processes are then deployed in the ESB, which contains the server engine that allows to operate them. It is during this time that process monitoring takes place and is of greater relevance to aid in the detection of unexpected events. This can be achieved by the ESB own monitoring capabilities as well as the implementation of monitoring techniques as described in section 3.3. An effective monitoring is most relevant when working towards the sustainability of interoperability, as explained before, since it allows for the detection of occurring changes and enables the possibility to quickly react to them and avoid harmony disruption.

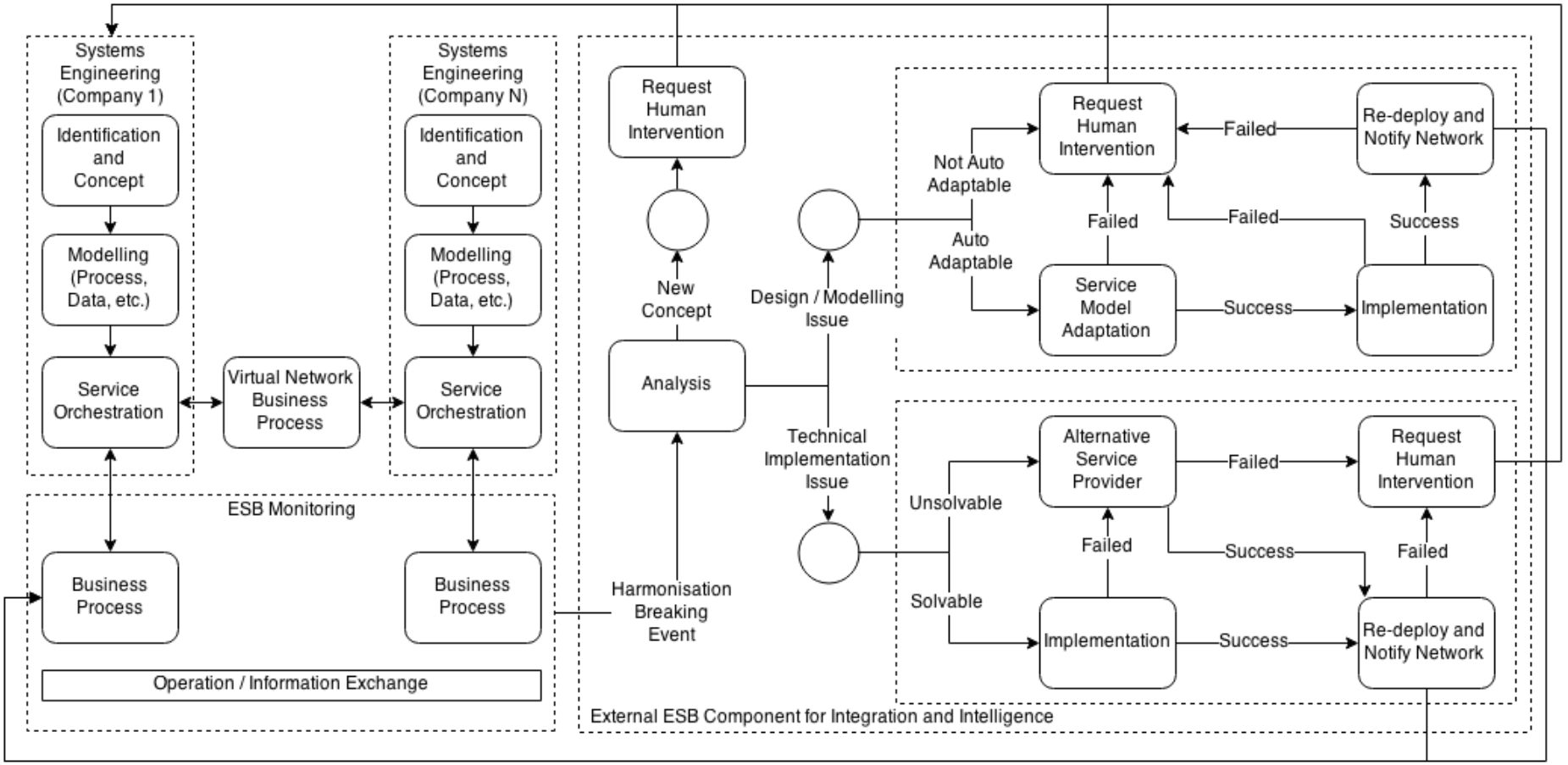

Figure 4.6 – Methodology for the sustainability of interoperability in service-based networks.

For the purpose of this methodology however, no third-party monitoring software was used, besides the one already incorporated with the ESB. The focus is on the capability to handle exceptions as they occur along system operation, and consequently launch an external intelligent component to assess the situation. Thus, as the system operates and process execution takes place, eventual harmonisation breaking events lead to the emergence of exceptions that are handled by the business process itself. When an exception is raised, if it pertains to an approachable issue, then the process proceeds to invoke the external component before terminating execution and warning the user that an unexpected error has occurred and that the issue is being automatically evaluated. Thus, once a harmonisation breaking event takes place, the external ESB Component for Integration and Intelligence (ESB-CII), as described in the methodology, is invoked in order to assess the situation and attempt the automatic recuperation of the affected process. At this point, once the ESB-CII has analysed the cause of the issue and following the concepts previously presented in Figure 4.2, three different scenarios are possible:

1. When the failure concerns a non-existing concept then no adaptation is possible, since the concept must first be created. In this case, human intervention is necessary so that the proper SE steps can be fulfilled in order for the new concept to be implemented;

2. In case the failure is due to a design or modelling issue that concerns a problem capable of being automatically approached, then an adaption of the business process may be attempted in order to integrate the new requirements into the system. However, if the process adaptation fails or the change in question cannot be taken automatically, then no automatic recovery is possible and human intervention is required;

3. In a third possibility, if the cause of error is due to a technical issue instead, which usually does not imply an adaptation to the process's own code and occurs in result of an internal implementation or an external factor, then it must be determined if it can be automatically corrected or not. From here, three distinct developments are possible: if the issue is solvable, then the process adaption is performed and implemented before automatic re-deployment; if the issue is unsolvable or if the former solution fails to succeed, then the ESB-CII attempts to find an alternative provider, associated with the same partner, capable of providing the same services as the original, after which the required adaption takes place and the process is re-deployed; if none of these solutions is possible or if they fail to succeed however, then once again human intervention must take place.

If process adaptation according to the required changes is indeed possible, then the ESB-CII proceeds to performing the necessary implementations itself and re-deploying the process in the server, accompanied with a notification to the user and system manager detailing the events that took place and the alterations that were performed. After these the user can once again invoke

the business process and return to normal operation. However, in case no process adaptation was possible and the issue could not be automatically solved, then a notification to both the user and the system manager are performed warning about the disrupting error and the inability to automatically solve it.

Thus, the proposed methodology suggests that by means of monitoring and an external aiding component, it may be possible to automatically resolve unexpected issues that occur during the normal execution of a business process, and in this way reduce the transitory phase that takes place during the state of quasi-equilibrium, as detailed in Figure 4.3. In consequence, the sustainability of systems interoperability could be improved, resulting in large benefits to CAS and similar networks, namely those in the industrial sector which operate in a chain-like manner and are heavily dependent on the existing partners, and where an interoperability disruption can have greater consequences on the production capability and therefore production costs.

Chapter 5

"[Implementation is] the process of translating a design into hardware components, software components, or both." - (IEEE Computer Society, 1990)

5 PROOF OF CONCEPT AND IMPLEMENTATION

This chapter aims to prepare for the validation of the proposed methodology as described in chapter 4, by introducing the architecture in which the proof-of-concept scenarios will be centred on, as well as the practical scenario concerning the IMAGINE (Innovative end-to-end Management of Dynamic Manufacturing Networks – http://www.imagine-futurefactory.eu) project that served as a basis for validation. The developments here described were achieved whilst working with the Group for Research in Interoperability of Systems (GRIS) at the UNINOVA research institute, Portugal.

Two proof of concept scenarios were considered for the detection of harmonisation breaking events in service-based networks which were effectively implemented and validated. Additionally, one of these scenarios was also implemented within the context of a real industrial environment, accomplished by means of the IMAGINE project, a IS being developed with the objective of providing support to modern manufacturing industries that comprise DMNs with globally distributed partners, suppliers, and production facilities (Markaki, Panopoulos, Kokkinakos, Koussouris, & Askounis, 2013). Its goal is to enhance companies with the capability to combine resources, share knowledge, increase time to market, and allow access to foreign markets (Barringer & Harrison, 2000), providing them with an easy, faster and more efficient way to search for the best partnering enterprises to support the manufacturing of their products, while aiming to enhance the automation and self-sufficiency of the processes that are involved in this exchange of information.

The following sections detail the technical requirements necessary to implement the proposed methodology as well as the technological tools that served in the implementations performed during the elaboration of this master thesis, why they were important, and how they contributed for validation.

5.1 TECHNICAL ARCHITECTURE AND PROOF OF CONCEPT SCENARIOS

As it was already noted several times, this master thesis proposes a methodological framework for the detection of harmonisation breaking events in service environments. However, in order for its implementation to be possible, it must be centred on a technical architecture that fulfils the necessary requirements. This architecture, illustrated in Figure 5.1, must be comprised of an ESB by which communication is achieved between the different entities along the network. Because the methodology is not based on a central solution, this means that each company that desires to implement it must have an individual ESB operating within itself. This enables the deployment of business processes in the ESB server engine, which in turn are responsible for orchestrating all the interactions that occur between that company's WSs and the various external WSs.

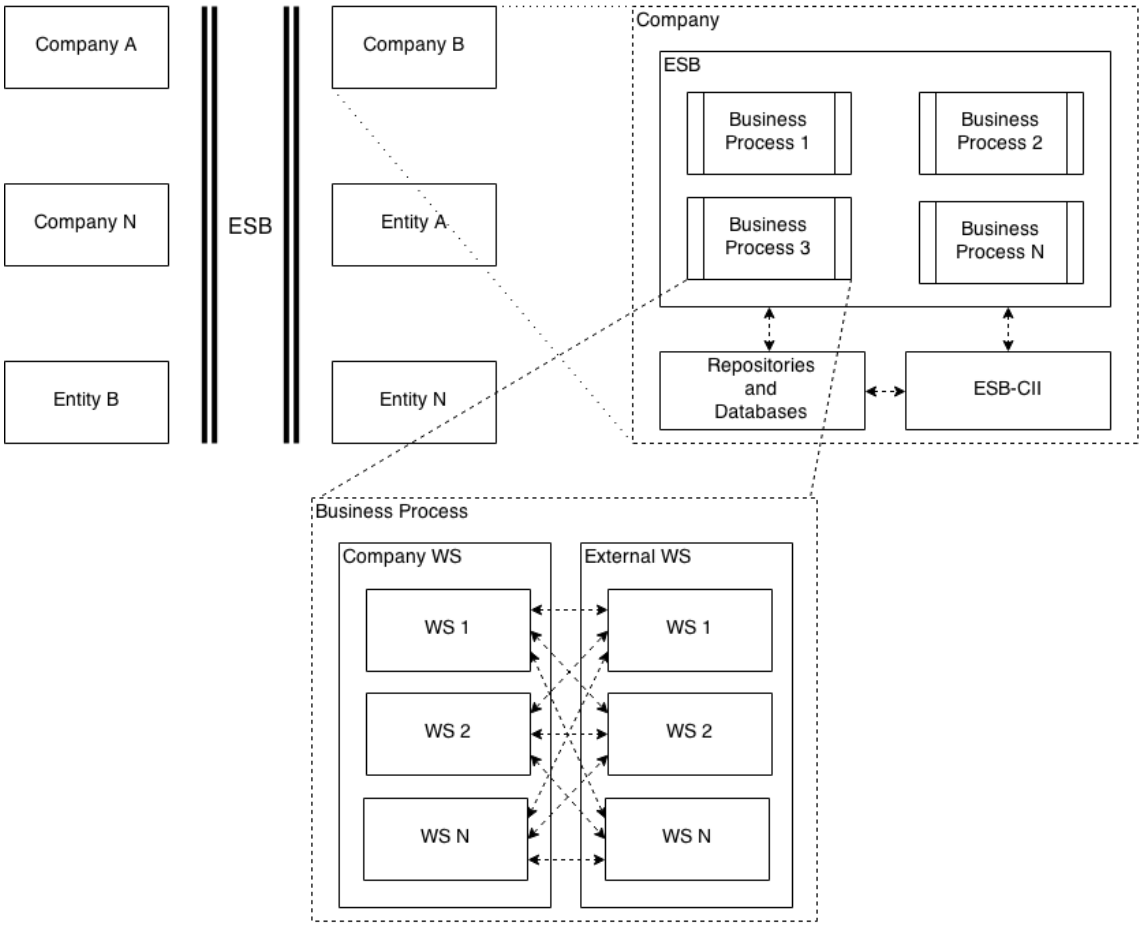

Figure 5.1 – Technical architecture for the implementation of the methodological framework.

The methodology also relies on an external component, the ESB-CII which is a simple application located somewhere within the system. In this manner, when a given interaction related

to a business process fails to succeed for any given reason, the process handles the raised exception itself and proceeds to invoke the ESB-CII by executing the referred application. Once this is done, the component then proceeds as defined in the methodological steps and attempts restore the erroneous business process within the ESB server engine. Thus, in this way, by possessing these essential technical requirements, the methodology for the detection and recuperation of harmonisation breaking events can be implemented and fulfilled. The following sections present two scenarios built around this architecture, that were considered as possible disrupting situations prone to occur in service-based networks.

5.1.1 The Standard Mismatch Scenario

When cooperating in a network companies often use standards, whether universal or not, to assure that the exchange of information is correctly achieved. The use of standards ensures a greater reliability and security regarding communications and the data that goes across the network. However, just like most of everything in the Information Technology (IT) domain, due to the constant economic and business evolution, standards are occasionally revised and consequently updated or adapted due to a variety of reasons. This means that in a scenario where two companies are exchanging information that is being ruled by a given standard, if one of those companies updates its standard but the other doesn't, interoperability may be broken. As such, this scenario, where two or more companies are using standards with different versions, may be classified as a harmonisation breaking situation. Figure 5.2 shows two companies exchanging information along an ESB where data is being represented according to the International Organization for Standardization (ISO) 10303 standard, an ISO standard that rules the representation and exchange of product manufacturing information. However, although both companies are using the same standard, they are using different versions of the Application Protocol (AP) 236, an ISO 10303 AP concerned with furniture product and project data. Because of this, a harmonisation breaking event is bound to occur. Supposing that both companies have access to a repository containing updated information about all the available standards, that company A is the one using the correct version of the ISO 10303 standard, and that the proposed methodology for approaching these disrupting events is effectively implemented in both of the illustrated companies, two possible situations could follow:

- Company A firstly detects an inconsistency regarding the received information and fires up the ESB-CII which, if capable of successfully detecting the cause of error as being due to an erroneous representation of data, then consults the referred ISO standard repository in order to validate its standards. Once it confirms that the standard in use is updated it concludes that no solution is possible, as the error is due to external factors (company B).

If there is an alternative service provider in the network an attempt to connect to it is made and company B can be disregarded. Otherwise, human intervention is requested;

- Company B firstly detects an inconsistency regarding the received information and fires up the ESB-CII which, if capable of successfully detecting the cause of error as being due to an erroneous representation of data, then consults the referred ISO standard repository in order to validate its standards. Once it detects that the ISO 10303 standard is outdated it can further examine the situation and attempt to correct the erroneous business process. If the standard in evidence is implemented through the use of coding libraries or XSL for instance, which are responsible for rendering the information that is being sent, this can be achieved by simply updating those libraries and re-deploying the business process. In case of success communication can proceed, otherwise company B attempts to find a different partner, hoping that it is also using an identical version of its standards, or in last case, calls for human intervention so that the process can be properly adapted to the new requirements.

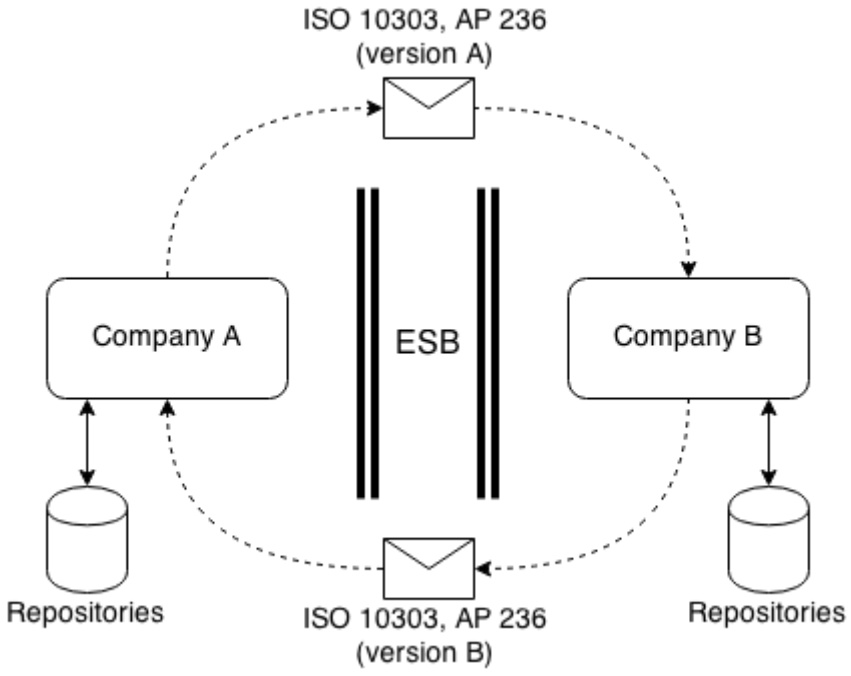

Figure 5.2 – The standard mismatch scenario.

Thus, this scenario represents a valid situation in which a harmonisation breaking event can result from a standard mismatch between partnering companies that can be effectively approached by implementing the proposed methodology, wherein the consequent result will vary depending on the company that firstly detects the error and responds to it. In a similar way, and

although this scenario concerns the use of standards, the same principles can be applied when specific programming libraries or APIs are being used by a given process, in which the representation of information relies on a particular shared programming class, for instance, that may suffer adaptations and consequently give rise to interoperability problems.

5.1.2 The Service Description Mismatch Scenario

Another possible interoperability disrupting situation concerns the interaction between the WSs of cooperating companies and their service description interfaces, such as the WSDL described in section 2.3. When a business process is initially created, it usually extracts the service descriptions of the services it is bound to consume, allowing to understand those services and know how to interact with them ahead of time. This provides the service consumer with all the provider's services information, its operations and input parameters. However, and as illustrated in Figure 5.3, if for some reason the service provider decides to modify its WSs and in consequence their respective WSDL files, the consumer will not be able to continue communication and interoperability may be lost.

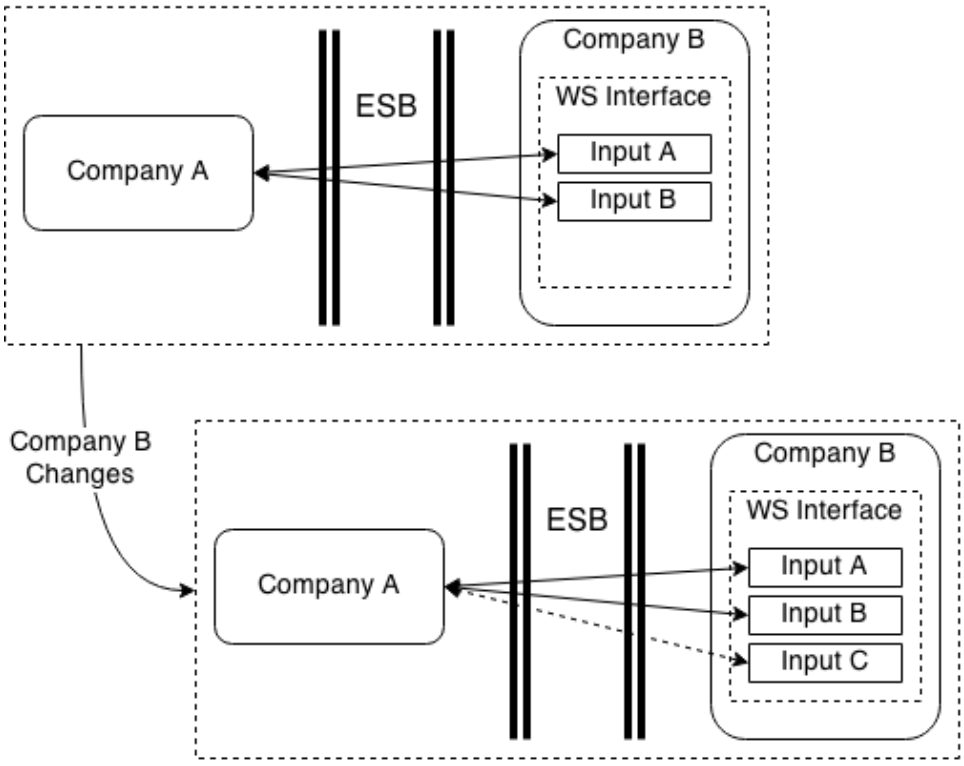

Figure 5.3 – The service description mismatch scenario.

In an analogous situation, it is the same as if someone changes the input parameters or even the name of a given function in a coding library or API, which would obviously result in an error for the applications using it. Thus, although a company shouldn't change its WSs during operation on a collaborative network, one must account for the possibility that it can indeed happen and if it does, the remaining system may suffer as a consequence. As mentioned, this scenario is further illustrated in Figure 5.3, where a company A is consuming an external WS as provided by company B. Once the WS interface description is modified, company A will no longer be able to consume the WS and an inevitable exception will raise within the respective business process. Because the error occurs entirely on company's A side, since company B is merely providing the service, it must be the former to implement the suggested methodology. Thus, once the exception is raised and the ESB-CII is launched, an analysis of the business process's code is performed and a comparison between the local service descriptions, obtained at the time of the process's creation, with the WS interface supplied by company B is made. Because they differ, the ESB-CII will detect an erroneous interface description is in use and will attempt to correct it. To achieve this, all the invocations of the respective WS must be analysed and compared with the valid WS interface after which, if previously pre-defined solutions are available, they are further adapted and corrected. Once all the required modifications have been performed, the business process is finally re-deployed and operation resumes. As already known, if the solution fails to succeed however, a search for an alternative provider with the corresponding service interface is performed. In last resort, once again, human intervention is requested.

Hence, this scenario constitutes another situation in which a harmonisation breaking event resulting from a service description mismatch can be properly tackled with the previously proposed methodology. Although presenting a possible solution, it should be reasserted that its eventual implementation can only be accomplished in case the occurring scenario was considered for when building the ESB-CII, as the modification of a service interface description is an extremely random and unexpected situation that can result in a vast number of unconsidered changes. On a last note, it should also be noted that because SOAP and REST WS implementations vary concerning service descriptions, the feasibility of the discussed solution is also dependant on the WS technology in use.

5.2 Approaching Harmonisation Disruption in the Manufacturing Domain

Beyond the implementation of the previously discussed proof of concept scenarios, an implementation concerning a real case system in the manufacturing domain was considered, in order to further validate the contribution and effects of the work here developed within a real world environment. Since the IMAGINE project fulfilled these necessary characteristics, it was selected as a real case scenario to test and verify the proposed solution for the sustainability of

interoperability in service-based networks. Hence, this section introduces the IMAGINE project as well as the developments accomplished during its execution, while it seeks to explain how they contributed to the validation of the referred methodology further discussed in chapter 6.

5.2.1 IMAGINE Project – The Concept

Relevant and reliable enterprise relationships are most important when companies seek to improve their productivity and decrease the time to market. Thus, to better achieve their goals, companies tend to join CN where they can share knowledge and work together towards a common purpose. This means that, as they cooperate, they are reaching for benefits such as reduced production time and cost. In this sense, to further improve these capabilities, the IMAGINE project seeks to implement a IS capable of aiding companies in the establishment of relationships and optimise the creation of DMN in the manufacturing domain, as it seeks to address the needs of modern manufacturing by providing an easier and simplified way to multi-party collaboration (Beca, et al., 2014). In order for this to be possible, each company that wishes to connect to the network and take advantage of it, must first register and provide all the information pertaining to its identity, capabilities, and resources. This allows the remaining companies to access this shared knowledge, and know who else is in the network and what their contribution is, accomplished with the help of a dedicated knowledge base that assists in the search and selection of the best partners available for collaboration. In this way, by providing companies with the best available partnerships, the IMAGINE platform (i-platform) enhances the cooperating capabilities of the CN, thus improving production profitability and cost reduction.

5.2.2 The IMAGINE Technical Architecture

The i-platform has been developed with the intent of contributing to the enhancement of the manufacturing industry. It is based on open standards to ensure interoperability, flexibility, and security, and focuses on the use of WSs to perform the interactions between the different entities. With this in mind and according to (IMAGINE Partners, 2013), a technical architecture was devised to better illustrate the functioning system from a high level perspective, as depicted in Figure 5.4.

The architecture comprises the following entities:

- The *companies*, that wish to be integrated into the network and use its platform to enhance their cooperation capability;
- The *IMAGINE platform*, which is the main implementation of the concept and contains the bulk of the system information and data;

- The *IMAGINE adaptor*, which allows a company to be incorporated into the network by transforming its data into the correct format;
- The *IMAGINE portal*, which presents the public interface through which the companies interact with the network.

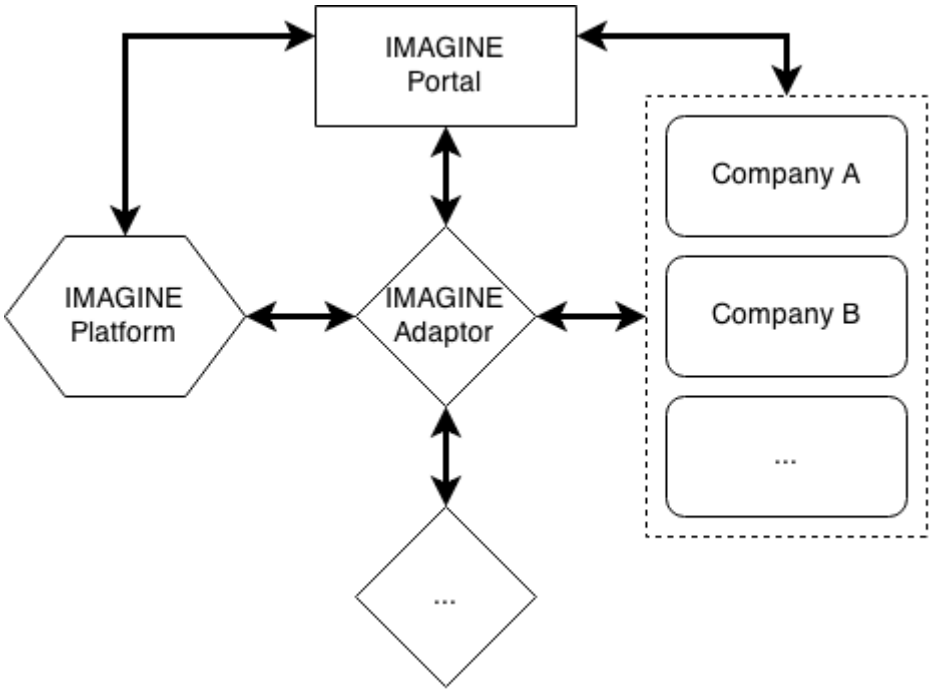

Figure 5.4 – The IMAGINE architecture.

Network communication is achieved by means of a communication bus, the ESB, which, as previously described in section 2.4, allows to easily implement a SOA and improve the integration and interaction between different services and applications, further providing a vast set of features and benefits such as network monitoring, reliability, scalability, and others. Furthermore, to allow for a seamless integration and increased network interoperability, the platform uses its own information models, the IMAGINE Blueprints, to store, exchange, and manage all of its information. By allowing to specify the IMAGINE knowledge base, the Blueprints allow to improve product quality and enhance the overall manufacturing capability, enabling the representation of companies and their resources, quality specifications, product characteristic and requirements, and others. Moreover, being generic in design, the Blueprints also allow the ability to further extend them to better adequate to new situations and requirements. Thus, the Blueprints serve as a way to enable network interoperability and facilitate the interactions between the various entities. They are used by the IMAGINE adaptors which are

responsible for monitoring and transforming the data from each company legacy system according to that of the Blueprint models. For companies to be able to directly interact with the platform, the IMAGINE portal is used, which presents the interface between the user and the platform itself. In this manner, companies are able to seamlessly integrate within the network and benefit from the features it provides for improved manufacturing.

Thus, as it can be observed by comparison with the required technical architecture previously illustrated in Figure 5.1 for the implementation of a proof of concept scenario, the IMAGINE technical architecture presents an ideal opportunity to fully implement and validate the methodological framework here proposed, as it contains all the necessary elements that ensure its functionality.

5.2.3 Tasks and Developments

This section details some of the developments achieved concerning the IMAGINE project and the manufacturing domain, which served as a basis for the real case implementation of the proof of concept scenario introduced in section 5.1.2, accomplished in cooperation with GRIS at the UNINOVA research institute. Being a large and complex project, the group was responsible for the development and implementation of the IMAGINE adaptor for integration within the platform which, as mentioned in the previous section, is responsible for transforming the data from each company legacy system according to the IMAGINE Blueprints, in order to enable system interoperability and seamless network communication. Furthermore, being the adaptor a point of passage of many of the interactions that occur in the system, it also presents an opportunistic place to implement network monitoring activities and in this way work towards the system's sustainability of interoperability. As communication is achieved by means of the ESB, which implies the creation and deployment of business processes, these developments presented an ideal opportunity to implement and validate the methodology previously suggested. Hence, because this research work is centred in service-based monitoring and interoperability, this section introduces two of the main business processes that were developed in regard to the referred implementations and which served as the foundation upon which to implement the proposed methodology, in regard to its real world case scenario validation.

5.2.3.1 Category Uploader

When initially configuring the platform for usage, before any company is registered and integrated into the network, one of the first operations that must take place is the upload of a category ontology – a hierarchical arrangement of entities and their relations – into the platform. These categories allow to correctly map and categorise each company resource during future company registrations, contributing in this way for a general and unified categorisation system

while avoiding inconsistencies between the different category systems of each company. To achieve this, the user interacts with the IMAGINE portal which in turn invokes a business process responsible for accessing the category ontology and load it into the platform, as illustrated in Figure 5.5.

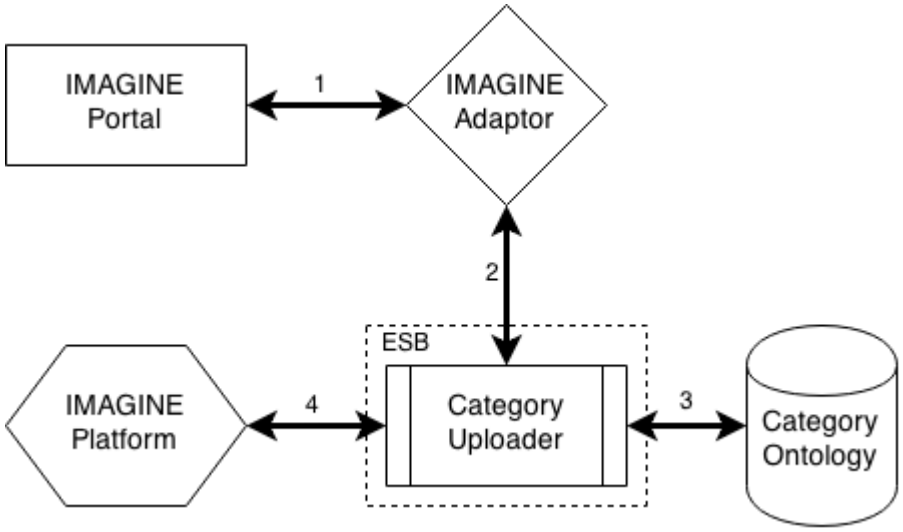

Figure 5.5 – Business process responsible for uploading the ontology categories.

Once the process is invoked, it loads all the existent categories in the ontology up into the platform which in turn saves it in its own information repository. In this way, when a company registers in the platform and uploads its resources, it can then map its own legacy category system according to the IMAGINE's categories, allowing to avoid interoperability problems prone to occur from different categorisation systems.

5.2.3.2 Company Uploader

As soon as all the categories have been uploaded and stored in the platform information repositories, then companies can start registering into the system. In order to achieve this, the company manager must first access the IMAGINE portal and fill a form with his company information, namely the company's WSs address, access credentials, capabilities, regions of operation, and other relevant details. Once all the information has been provided, the platform then starts a business process, through the IMAGINE adaptor, responsible for loading all the company's information and upload it into the IMAGINE platform. This process is depicted in Figure 5.6.

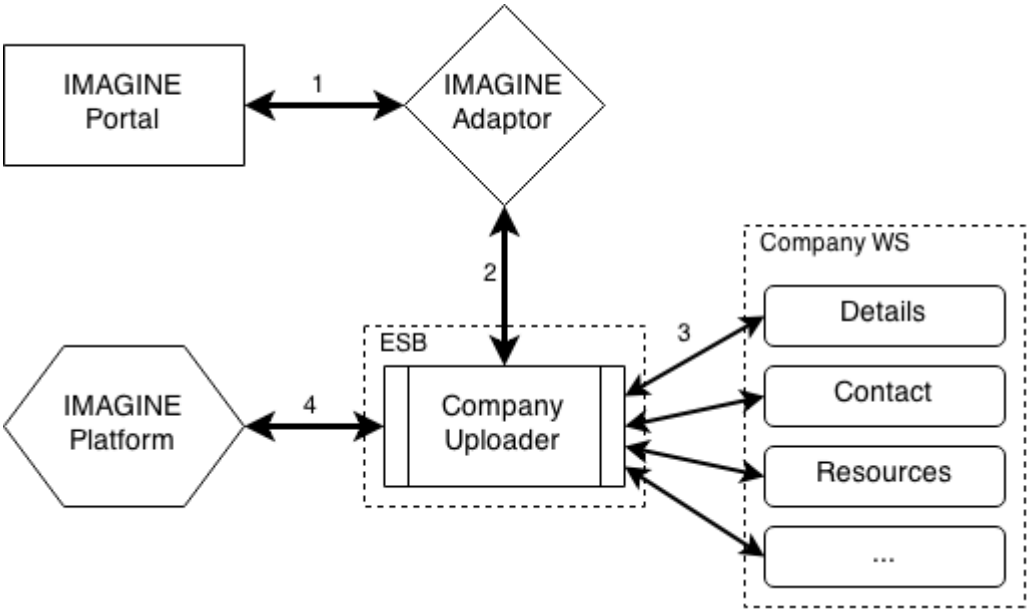

Figure 5.6 – Business process responsible for uploading a company's data.

For the upload process to be possible, companies need to provide a set of pre-defined WSs, which must be identical for each company, capable of retrieving specific information in a pre-defined structure. A list of the available WSs at the time of implementation that must be provided by each company is described as follows:

- *getFile* – Retrieves information related with the company's own file specifications, which are usually associated with its resources;
- *getCompanyRelationship* – Retrieves information related with the company's relationships with partnering companies;
- *getProduct* – Retrieves information related with the products the company provides;
- *getPersonWithContactDetails* – Retrieves the contact details for a specific person associated with the company;
- *getOrderMetric* – Retrieves information related with a specific order's metrics;
- *getCompany* – Retrieves the company's basic information and details, such as the name, the Value Added Tax (VAT) identification number, the number of employees, and other similar information;
- *getSkill* – Retrieves information related with the company's skills;
- *getContactDetails* – Retrieves the company's contact details;

- *getMaterial* – Retrieves information related with the materials the company provides;
- *getProcess* – Retrieves information related with the processes the company is capable of performing;
- *getDynammicSupplierInformation* – Retrieves information related with a specific company's resource, such as cost, duration, capacity rate, and others;
- *getStatus* – Retrieves information related with the status of a given order;
- *getMetric* – Retrieves information related with a specific resource's metrics;
- *getEquipment* – Retrieves information related with the equipment the company provides.

Thus, for the business process to function correctly, the company must provide all the enumerated WSs with all the pre-defined structuring and configuration. In this manner, the process can access any company's information and upload it into the platform, without having to worry about how to obtain the information for a specific company. This is the only instance in which a company needs to properly adapt itself in order to use the IMAGINE platform. Even so, due to the fact that this process is rather complex, as the amount of information is usually very extensive, it would undoubtedly benefit with the implementation of monitoring capabilities and eventual harmonisation recovery systems such as the one proposed in this document.

Finally, once all the categories have been inserted and a company has been registered and integrated into the platform, the company manager can then proceed to map his own company's resources according to the available categories, which as mentioned, allows for a general and unified categorisation system. This is an essential process for the seamless integration of each company into the network, enabling the understanding between the different entities and thus establishing interoperability. Once all the company's resources have been properly mapped, the company manager may then start using the platform to search for the desired partners and create an optimal DMN.

5.3 TECHNOLOGY OVERVIEW

The previous section introduced the required architecture to properly implement the methodology proposed in chapter 4, as well as its application within the IMAGINE project. However, in order to properly achieve the formerly referred implementations, namely those concerning the business processes and the methodology itself, several technologies had to be evaluated and selected for execution. Thus, this section introduces those very same technologies in order to elucidate the reader on how validation was achieved and how their contribution was important. One should note that, being this work for research purposes as well as for possible

integration in an international project, most of the chosen technologies were selected by having in consideration if they were standardised, as this ensures a greater reliability, functionality, and most importantly, interoperability.

5.3.1 WSs and Business Processes

As it was already mentioned several times before, this master thesis purposes a methodology to aid in the sustainability of interoperability in service-based networks, where communication is achieved by means of WSs usually orchestrated through business processes.

In terms of WSs, there were two major options concerning their technological implementation: SOAP and REST, as explained in section 2.3. However, unlike SOAP, because REST is an architectural style and not a protocol, there is no standard for RESTful APIs. For this reason SOAP was chosen for the implementation of WSs, along with its WSDL. Moreover, as it is already known, SOAP uses XML to build message contents and communicates via HTTP, SMTP, or other application layer protocols. Because XML is a structured language with well-defined and human-readable sets of rules, it also greatly facilitates the adaptation and transformation of code, which is often necessary during the methodological process previously suggested in chapter 4.

In regard to business process implementation, the means by which WSs interactions are orchestrated and in a similar manner to the WSs technological choices, there were two main possible solutions: WS-BPEL and BPMN, as explained in section 2.4.1. Although BPMN provides a more easily accessible visual implementation with simplified process flows, its lack of standards lead the choice to the WS-BPEL, which works with widely used standards. Furthermore the WS-BPEL integrates easily with SOAP-based WSs and it also uses XML to describe its processes, making it a most ideal choice for the reasons already mentioned.

5.3.2 Choosing the ESB

Once the appropriate technology was chosen concerning WSs and business process implementations, it was then necessary a server in which those developments could be deployed and operated. This is where the ESB introduced in section 2.4 becomes relevant, as it provides an architecture capable of integrating different services and applications in a centralised manner. The ESB choice was mainly based on the BPM standards available in each solution, as it was necessary for it to be natively integrated with the already selected BPEL, as well as other relevant factors such as cost, it needed to be free, licensing, and others. Due to the large amount of available ESB implementations, and in an effort to ease the selection process, a table with all the most

relevant candidates and the most important features and properties was elaborated and is presented in Table 5.1.

To further facilitate the process, a score system ranged from 0 to 2 was created to valorise the different features of each ESB, in which 0 is the lowest score possible and represents an unwanted feature or property, 1 is an average score and represents a feature that is not unwanted but it's not ideal, and 2 is the highest possible score that represents an ideal and wanted feature. The ESB with the higher score is the one selected for implementation usage.

By observing the selection table, it is evident that one the factors of choice was the ability to work with JAVA, since this allows not only to deploy the ESB across different platforms, but also to achieve the same purpose in the case of future developments. As described, only the most relevant and free ESB implementations were considered for the selection method, which although practical and useful, did not go into specific detail concerning each available ESB. Nevertheless, the correct choice of an ESB was rather important since this unavoidably affects the operation and management of the WSs and business processes themselves, while providing the infrastructure for stability and system interoperability. In the end, the choice fell on the OpenESB.

Table 5.1 – ESB selection process comprised of a scoring system ranged 0 to 2.

	MuleSoft Mule ESB	**OpenESB**	**OW2 Petals ESB**	**Red Hat Fuse ESB**	**Talend ESB**	**WSO2 Enterprise Service Bus**
Operating System (Score)	Cross-platform (2)	Cross-platform (2)	Cross-platform (2)	Cross-platform (2)	Cross-platform (2)	Cross-platform (2)
Cost (Score)	Free / Commercial (1)	Free (2)	Free / Commercial (1)	Free / Commercial (1)	Free / Commercial (1)	Free / Commercial (1)
Open-Source (Score)	Yes (2)	Yes (2)	Yes (2)	Yes (2)	Yes (2)	Yes (2)
Java (Score)	Yes (2)	Yes (2)	Yes (2)	Yes (2)	Yes (2)	Yes (2)
Documentation (Score)	Limited (1)	Detailed (2)	Detailed (2)	Limited (1)	Lacking (1)	Limited (1)
IDE (Score)	Eclipse; Mule Studio (2)	NetBeans (2)	Eclipse; Petals Studio (2)	Eclipse; Fuse IDE (2)	Eclipse; Talend Open Studio (2)	Eclipse; WSO2 Developer Studio (2)
Integrated BPM (Score)	No (0)	Yes (2)	Yes (2)	No (0)	Yes (Commercial) (0)	Yes (2)
BPM Standards (Score)	BPMN 2.0 / BPEL 2.0 (2)	BPEL 2.0 (2)	BPMN 2.0 / BPEL 2.0 (2)	BPEL4WS 1.1 / BPEL 2.0 (2)	BPMN 2.0 (0)	BPEL 2.0 (2)
WSDL Integration (Score)	Yes (1)	Yes (JBI) (2)	Yes (JBI) (2)	Yes (JBI) (2)	Yes (1)	Yes (1)
Total Score	13	**18**	17	14	11	15

5.3.3 Building the ESB-CII

As the ESB-CII is an external component to the ESB, it could be developed in practically any programming language, as long as it could achieve the desired purpose. However, because the requirement for cross-platform developments was present and because some of the previous technologies were already using JAVA components, the JAVA programming language was chosen to achieve this particular development. Furthermore, the JAVA language is widely used in the IT domain, providing rich libraries and APIs, a vast documentation, good overall performance, good Integrated Development Environment (IDE) tools, seamless integration with the various Operating Systems (OS), and other important features.

5.3.4 Other Technologies

Besides the already mentioned technologies, other less relevant software tools were used during the developments here presented, such as the ones described below:

- *Resource Description Framework (RDF)*, which consists in a World Wide Web Consortium (W3C) specification for metadata and information description. The RDF is present in the IMAGINE platform's information repositories and was relevant when validating the implemented proof of concept scenario;
- *SPARQL Protocol And RDF Query Language (SPARQL)*, an RDF query language able to retrieve and manipulate the data stored in RDF repositories, used in the same situation as described previously;
- *My Structured Query Language (MySQL)*, a widely used open-source relational database management system, for storing all types of information;
- *Structured Query Language (SQL)*, a MySQL query language able to retrieve and manipulate the data stored in MySQL databases;
- *Microsoft Windows*, the operating system in which all the developments were achieved.

Chapter 6

"[Validation is] the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements." - (IEEE Computer Society, 1990)

6 PROOF OF CONCEPT AND HYPOTHESIS VALIDATION

Following the previous chapter, which introduced the proof of concept scenarios considered for implementation, as well as the technical architecture and technical requirements over which the proposed methodology must be based on, this chapter proceeds to fulfil the accomplished developments and further validate the hypothesis firstly introduced in section 1.2.3. This suggested that if it is possible to properly monitor and timely detect harmonisation breaking events in service-based networks, then it may also be possible to timely react, adapt, and reduce the time in which interoperability cannot be entirely established, contributing in this manner to the sustainability of interoperability.

Two different proof of concept scenarios were presented in section 5.1, one concerning a standard mismatch and the other concerning a service description interface mismatch. While both these scenarios were implemented and tested for validation purposes, only the service description interface mismatch scenario was effectively integrated within the context of a real industrial environment by means of the IMAGINE project, introduced in section 5.2, which due to its technical architecture and domain of expertise, presented an ideal opportunity upon which to implement and validate the methodological framework for the detection of harmonisation breaking events in SOA-based networks. It should be noted however, that due to timing restrictions and other limitations, only the situations that are considered as initially solvable, as described in the proposed methodology, have been effectively implemented and validated. This means that issues that cannot be immediately approached and require the secondary solution of searching for an alternative provider, were not implemented during the developments hereby presented and remain as future work and research.

6.1 TESTING METHODOLOGY

In order to find faults within a given process and effectively validate it, implementation and experimentation must take place. This is usually accomplished within a closed environment in which normal and abnormal operation of the process in test is simulated. Even so, while testing and validation allow to gain confidence on the ability to perform, testing and behaviour simulation can only go so far, since the testing of real systems can never be exhaustive and fully accomplished. The following sections present some methodologies that can be used to better test and validate the proof of concept scenarios previously described. Nevertheless, it should be noted that a successful validation of the referred scenarios does not necessarily imply that they are fully ready for integration within real and commercial systems, but rather that the fundamental concept is valid and can be further built upon.

6.1.1 A Functional and Non-Functional Evaluation Methodology

This section briefly introduces a functional and non-functional testing methodology based on the iSurf project (interoperability Service utility for collaborative supply chain planning across multiple domains supported by RFID devices). The iSurf was a research project integrated in the EU Seventh Framework Programme with the objective of providing an environment to ease the collaboration of trading partners while satisfying costumer demands in a supply chain (iSurf Partners, 2009). However, beyond the research and development on inter-enterprise collaboration along a supply chain, the iSurf project also invested in the development of an evaluation and testing framework, following and complementing the standard process envisaged by the reference model ISO/IEC CD 25040 (ISO/IEC, 2007) of the SQuaRE (Software product Quality Requirements and Evaluation) series of standards.

SQuaRE details the activities and tasks providing their purposes, as well as the outcomes and complementary information that can be used to guide a software product quality evaluation. The iSurf project describes in detail the procedures used to generate the evaluation criteria to be applied for the functional and non-functional characteristics of intermediate and final products, namely functionality, reliability, usability, efficiency, maintainability and portability. These techniques and their evaluation criteria were modularised as recommended in ISO/IEC 25041 (ISO/IEC, 2012). The modules are documented as specified on the standard ISO/IEC 14598-6 (ISO/IEC, 2001) and are described as follows:

1. Provides formal information about the evaluation module and gives an introduction to the evaluation technique described;

2. Defines the scope of applicability of the evaluation module;

3. Specifies the input products required for the evaluation and defines the data to be collected and measures to be calculated;
4. Contains information about how to interpret measurement results.

These functional and non-functional evaluation modules provide a flexible and structured method that allows to define criteria for monitoring the quality of intermediate products during the development process and for evaluation of final products. They provide a set of structured instructions with the purpose of ensuring that software evaluations are easily reproducible and objective. The goal is to make the various characteristics (such as principles, metrics, activities, and others) of evaluation, visible and understandable. Furthermore, the iSurf evaluation modules define the criteria for the evaluation of components considering the quality characteristics specified on the SQuaRE series of standards, as described below:

- 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 The Tree and Tabular Combined Notation (TTCN)

The Tree and Tabular Combined Notation is a test notation standardised by the ISO/IEC 9646-1 (ISO/IEC, 1994), that concerns the specification of tests for communication systems developed within the framework of standardised conformance testing (ETSI, 2009). In the TTCN, tests behaviour is defined by a sequence of events and specified through tables which are divided in general description, constraints, behaviour and verdict. This sequence can be approached as a tree, containing branches of actions based on the evaluation of the system output after one or a series of executed events, which can be of one of two types:

- Actions, which are preceded by an exclamation point before its brief description, and represent actions performed on the System Under Test (SUT);

- Questions that 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. Once the TTCN test table is complete, three verdicts based on the sequence of events and the outputs of the system are possible: *Success*, *Failure* or *Inconclusive*. To better illustrate the testing procedure, Table 6.1 presents a simplified example describing an evaluation description for a phone call.

Table 6.1 – TTCN table test example.

Test Case			
Test:	Basic Phone Call Connection		
Group:	N/A		
Purpose:	Check if a phone call can be established.		
Comment:	N/A		
Behaviour		**Constraints**	**Verdict**
! Pick up headphone			
? Dialling tone			
! Dial Number			
? Calling tone			
? Connected line			
! Hung up headphone			Success
OTHERWISE			Failure
? Busy tone			
! Hung up headphone			Inconclusive
OTHERWISE			Failure
? Dialling tone absent			Failure

Thus, as one can observe, the table test is structured in a hierarchical manner similar to an IF-ELSE programming statement, and can be textually read as:

1. The user picks up the headphone;
2. Tests if the dialling tone is present;

3. If the dialling tone is present, then the user dials the phone number. Otherwise, if the dialling tone is absent, the verdict is a *Failure*, due to the inability of establishing a phone call;

4. If there is a calling tone after dialling the number, then the user may test if the line is effectively connected;

5. If the line is connected, the user may hung up the headphone and the verdict is set as *Success* on establishing a phone call, otherwise the verdict is *Failure*;

6. If the dialling tone is not heard, but a busy tone is present instead, then the user may hung up the headphone and the verdict is set as *Inconclusive*;

7. Finally, if none of the tones corresponds to the calling or busy tone, then the verdict is defined as *Failure* on establishing a phone call.

6.1.3 Adopted Test Methodology

Following the previous sections, and in order to properly implement the proof of concept scenarios considered in chapter 5, a combination of the testing methodologies presented was chosen to validate the proposed solution for the detection and prevention of harmonisation breaking events in service-based networks. Based on the iSurf and TTCN testing methodologies, a simple set of functional tests were conducted in respect to the referred proof of concept scenarios, namely in regard to the domain of manufacturing, while non-functional considerations concerning reliability and efficiency were also briefly mentioned. Nevertheless, as mentioned before, the result of the tests and validations here performed do not guarantee a flawless and complete solution, but rather a working proof for the concept of study.

6.2 FUNCTIONAL TESTING

This section presents all the implementations, tests, and validations concerning the proof of concept scenarios described previously in chapter 5. All tests were performed within the context of dynamic industrial networks and were centred on a technical architecture identical to the one introduced in section 5.1, as well as the technological tools selected in section 5.3, enabling in this manner the effective implementation of the solution proposed in chapter 4.

With this in mind, a simple business process developed in the WS-BPEL language was formulated with the intent of providing a basis for testing and validation. The process, illustrated in Figure 6.1, consists of a simple interaction between a consumer and an external service provider that offers a set of WSs concerning the extraction of information related to a specific company. Thus, in order to obtain basic information regarding a given company, the consumer must invoke

the respective WS, *getCompany*, and supply it with the correct input parameters, as described below:

- *Username* – The username required for authentication purposes;
- *Password* – The password required for authentication purposes;
- *VATNumber* – The VAT number concerning the company on which the consumer desires to obtain the information.

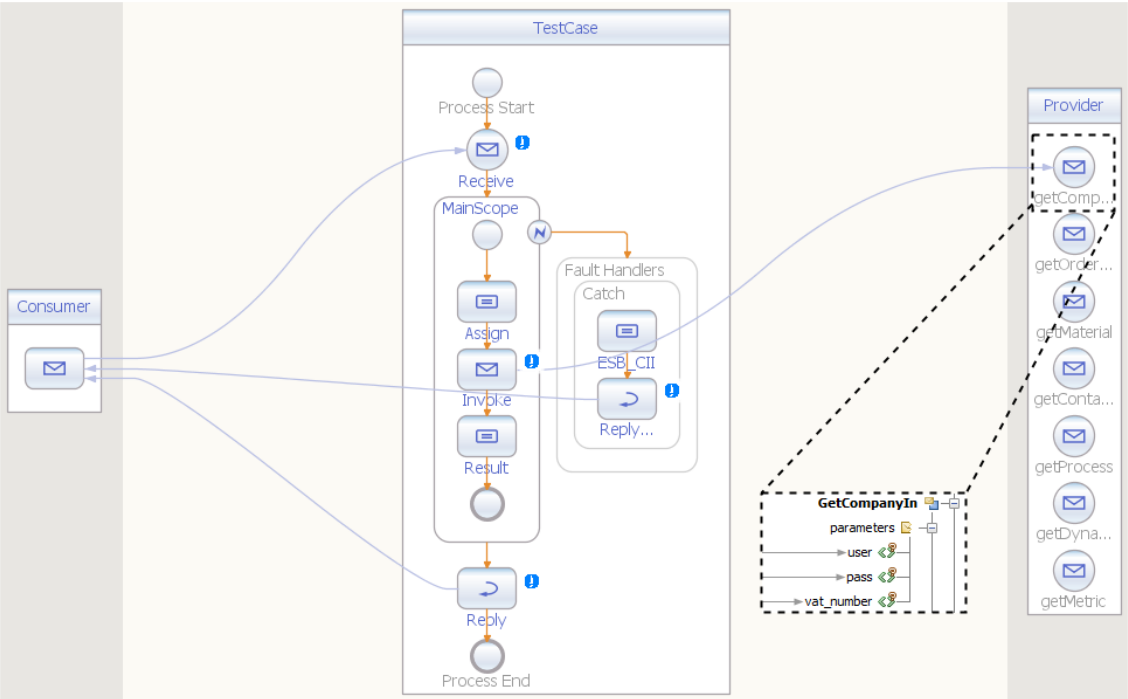

Figure 6.1 – WS-BPEL process that served as a basis for testing and validation.

Hence, once the business process has been initiated and the service has been invoked with the correct parameters supplied, the WS replies with the information regarding the respective company. The returned information is serialised, i.e. marshalled or transformed, according to a data structure pre-defined in a shared JAVA library (*Blueprints.jar*) between the consumer and the provider. In this manner, in case the WS-BPEL wishes to access the contained data, it must reverse the serialisation process and extract the wanted information, which can then be worked on within the JAVA code and converted into JAVA objects. An example of the result obtained after the WS-BPEL process has executed is illustrated in Figure 6.2, which depicts a simple HTTP invocation through a web browser of the referred WS-BPEL process. As it can be observed, the parameters are supplied within the URL itself and if valid, the information is then returned and

displayed in the web browser's window, structured according to the Blueprints JAVA classes, which for demonstration purposes was not deserialised.

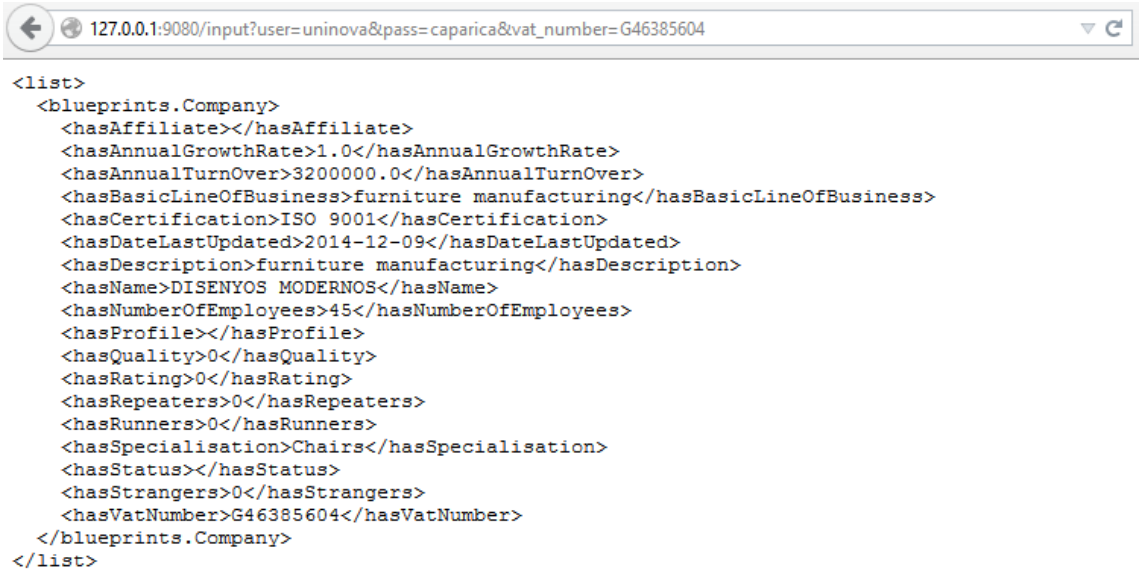

Figure 6.2 – The result obtained from the WS-BPEL test case.

As Figure 6.1 illustrates, the WS-BPEL process is further enhanced with fault handling capabilities. This error handling is what allows to timely catch occurring exceptions, i.e. harmonisation breaking events, in an effort to tackle and mitigate them. Although the OpenESB tool provides a variety of default fault types that can be caught and handled, a JAVA class containing customised exceptions (*Exceptions.java*) was created. This contributed to improve the ESB capabilities by enabling the creation of specific and customised errors, according to the situation in the moment. For the tests hereby presented, two different exceptions were created:

- *BlueprintException* – Used when an error concerning the serialisation or deserialization of data according to the Blueprints classes occurs;
- *WebServiceInvocationException* – Used when an error occurs during the invocation of a given WS.

Thus, the above exceptions contributed to the validation of the standard mismatch and the service description mismatch scenarios respectively, as they enabled to specifically tackle the errors occurring in each situation. This is important because it serves as a first filter of possible error causes when calling the ESB-CII. If the referred component is launched and is confronted with a *SystemFault* exception for instance, it is unable to determine the cause of error due to the wide range of possibilities. However, by using customised fault types, one can programmatically

raise a specific and personalised exception when executing the WS-BPEL process, providing the ESB-CII with an immediate and more evident cause of error, and contributing in this manner to the success of the proposed methodology.

Once an exception has been caught, an error message is returned to the user and the ESB-CII is consequently launched and executed, as illustrated in Figure 6.3, thus starting the methodological steps introduced in chapter 4. This consists in the simple execution of a JAVA application that runs in the same system but externally to the ESB, being able to access it and modify its processes.

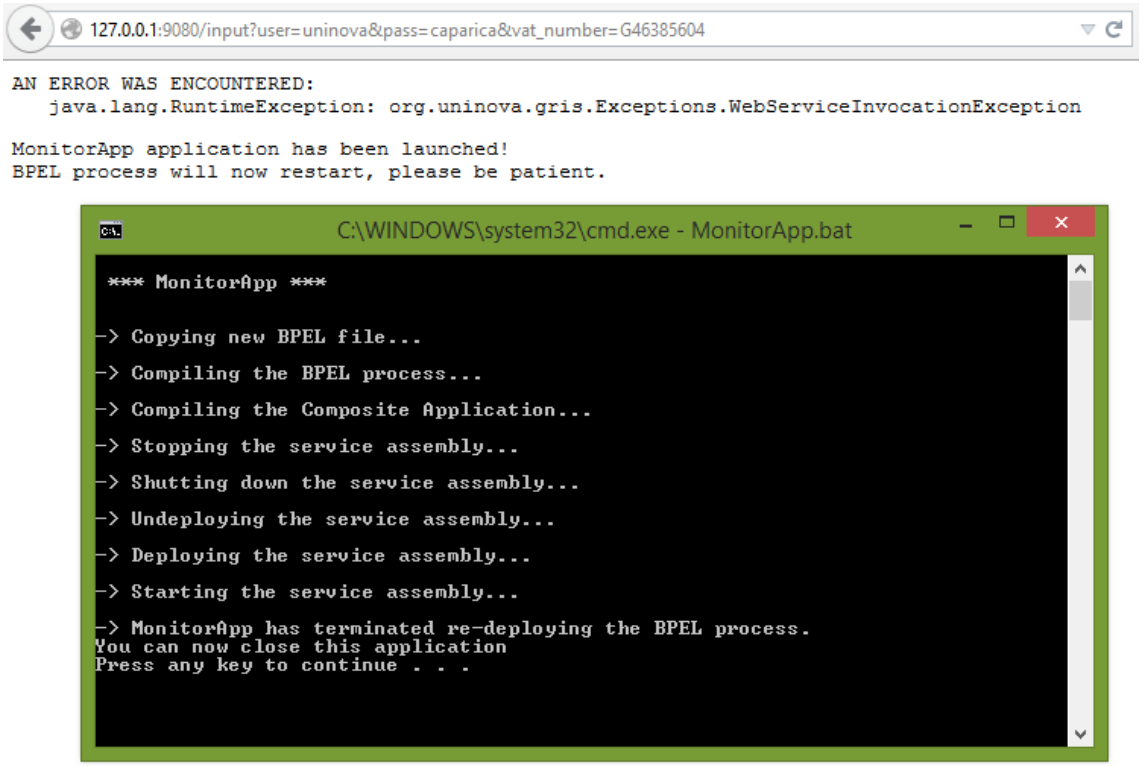

Figure 6.3 – Launching the ESB-CII.

However, for the ESB-CII to be effective, the faulty WS-BPEL must be first stopped and un-deployed before any changes are made and implemented. All the actions concerning the deployment and un-deployment of the business process are made by running shell commands within the operating system, which are specific to the ESB software tool in use. In the case here depicted, which implements an instance of the OpenESB, these commands are:

- *stop-jbi-service-assembly*, to stop the WS-BPEL execution;
- *shut-down-jbi-service-assembly*, to shut down the WS-BPEL;

- *undeploy-jbi-service-assembly*, to un-deploy the WS-BPEL from the server;
- *deploy-jbi-service-assembly*, to deploy the WS-BPEL in the server;
- *start-down-jbi-service-assembly*, to start the WS-BPEL execution.

With this knowledge, the next sections describe the tests concerning the proof of concept scenarios previously considered and attempt to validate the suggested methodology for the detection of harmonisation breaking events.

6.2.1 The Standard Mismatch Scenario Validation

The standard mismatch scenario, described in section 5.1.1, concerns a situation in which a business process is using a specific standard, programming library, or API for the representation and structuring of data that due to an external event becomes outdated and consequently unusable, consisting therefore in a technical requirement type of situation, as described in the methodology. For the tests here presented, the scenario concerns the use of the previously mentioned Blueprints classes, a set of JAVA classes used during the validation phase that served to structure information according to a pre-define format, as illustrated in Figure 6.2 in the previous section. In this sense, the testing WS-BPEL process was altered and a simple extra operation was introduced, consisting in the deserialisation and extraction of the information returned from the *getCompany* WS into a JAVA object, as depicted in Figure 6.4.

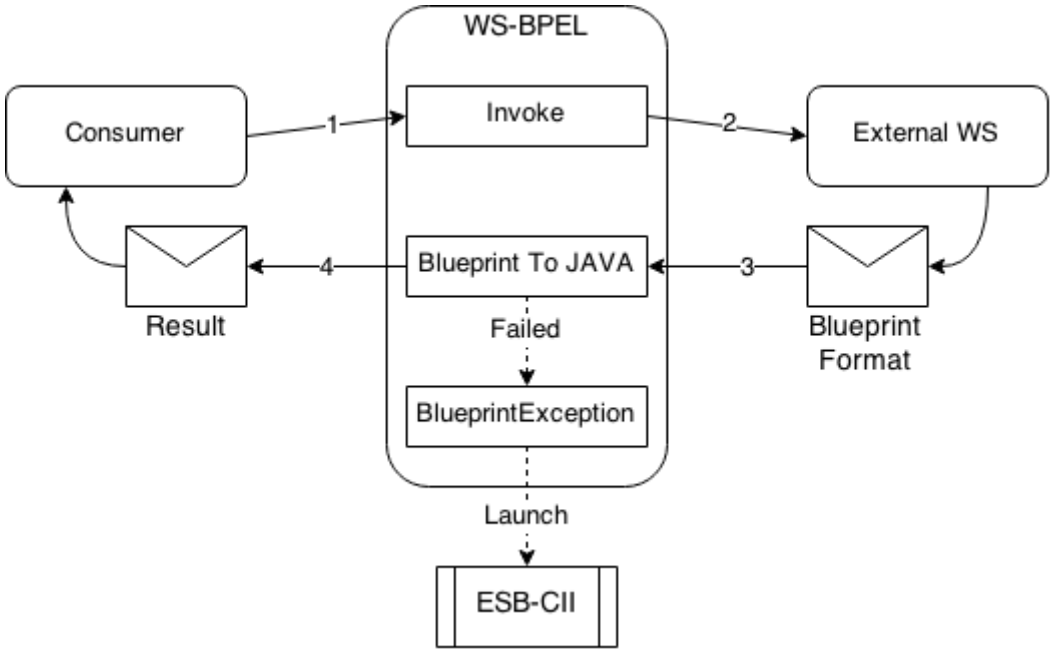

Figure 6.4 – The standard mismatch validation scenario.

With the implemented modification, the WS-BPEL will therefore deserialise the received information and transform it into a JAVA object. If it succeeds no further action is taken and a successful response is returned to the consumer, otherwise the ESB-CII is launched. However, in order to force an exception for testing purposes, the Blueprints classes were modified within the external WSs provider entity and a new version was issued. Table 6.2 presents the testing results achieved during the validation phase while disregarding the effects of the ESB-CII.

Table 6.2 – Standard mismatch scenario functional test without the ESB-CII.

Test Case			
Test:	Standard Mismatch Scenario without the ESB-CII		
Group:	Standard Mismatch Scenario Validation		
Purpose:	Test the given scenario during normal operation.		
Comment:	N/A		
Behaviour		**Constraints**	**Verdict**
! Start WS-BPEL			
! Invoke *getCompany* WS			
? WS input parameters are correct			
? Receives WS response			
? Deserialises successfully			
! Reply			Success
OTHERWISE			Failure
OTHERWISE			Failure
OTHERWISE			Failure

Hence, if the WS-BPEL can successfully invoke the *getCompany* WS, receive its information, and thus deserialise it into a JAVA object, the test succeeds. Otherwise, if any of the mentioned conditions cannot be achieved, the test fails and the ESB-CII is consequently invoked.

Once the ESB-CII is invoked, it receives as an input parameter with the exception that led to its invocation. If the exception is considered for within its code, such as those contained in *Exceptions.java*, then it can approach the issue as it was programmed to. However, as mentioned before, if the exception sent to the external component is not considered for, then the latter will be unable to know the originating cause of the problem and the first solution as described in the proposed methodology in chapter 4 is immediately disregarded, proceeding therefrom to the

second possible solution which is the search for an alternative provider. Table 6.3 presents the testing results achieved during the validation phase after the invocation of the ESB-CII.

Table 6.3 – Standard mismatch scenario functional test with the ESB-CII.

Test Case			
Test:	Effects of the ESB-CII within the Standard Mismatch Scenario		
Group:	Standard Mismatch Scenario Validation		
Purpose:	Test the proposed methodology within the given scenario.		
Comment:	N/A		
Behaviour		**Constraints**	**Verdict**
! Start ESB-CII			
? Exception is provided			
? Exception is the *BlueprintException*		*Exceptions.java*	
? Updated *Blueprints.jar* is available		Repository	
? Replaces library file			
? Re-deploys BPEL			Success
OTHERWISE			Failure
OTHERWISE			Inconclusive
OTHERWISE			Inconclusive
OTHERWISE			Failure
OTHERWISE			Failure

As described, if the exception provided as an input to the invocation of the ESB-CII is the *BlueprintException* defined in the *Exceptions.java* class, then an updated version of the *Blueprints.jar* library is obtained from an available repository, which in the current case was simply a local folder, and integrated with the WS-BPEL process by replacing the old library. Finally, once the adaptation has been made and the WS-BPEL re-built and re-deployed, then the validation is fulfilled and the ESB-CII has been capable of successfully detecting and mitigating a harmonisation breaking event originating from a version mismatch of the referred JAVA library. In the test here performed, the ESB-CII intervention merely consisted of copying the correct JAVA libraries into the WS-BPEL project's folder and thus replace the old files. However, in order to improve its capability, the ESB-CII could be further enhanced to first compare the versions in question and only replace them if necessary. Nevertheless, the test completed and the proposed methodology was successfully validated within the presented scenario, proving to be a

reliable, as no drastic or unsafe modifications took place, and efficient framework, as no human intervention was required, for the sustainability of interoperability.

While the solution proposed in chapter 4 has been proven useful in maintaining interoperability within the current context, several situation could occur that would lead to failed executions, as it can be observed in Table 6.3. For instance, if no exception is provided when invoking the ESB-CII or if the WS-BPEL fails to re-deploy for an unknown reason, then the external component integration would be unsuccessful and human intervention required. Besides these failed situations, several inconclusive results can also be observed. These results concern situations in which the first attempted solution by the ESB-CII fails, as described within the proposed methodology, and a secondary solution is attempted, the search for an alternative service provider. However, as it has been previously mentioned, due to timing restrictions and other limitations, this secondary solution was not implemented during the research and developments achieved along the elaboration of this master thesis and remains a case for future work and research. For this reason, since it could not be validated, the verdict that results from such situations is defined as inconclusive.

6.2.2 The Service Description Mismatch Scenario Validation

The service description mismatch scenario, described in section 5.1.2, concerns a situation in which a business process orchestrates the interaction between a consumer and a provider, when at a given point in time, the provider adapts and changes the interface description of its WSs, rendering the business process ineffective and unable to consume the respective WSs, as illustrated in Figure 6.5.

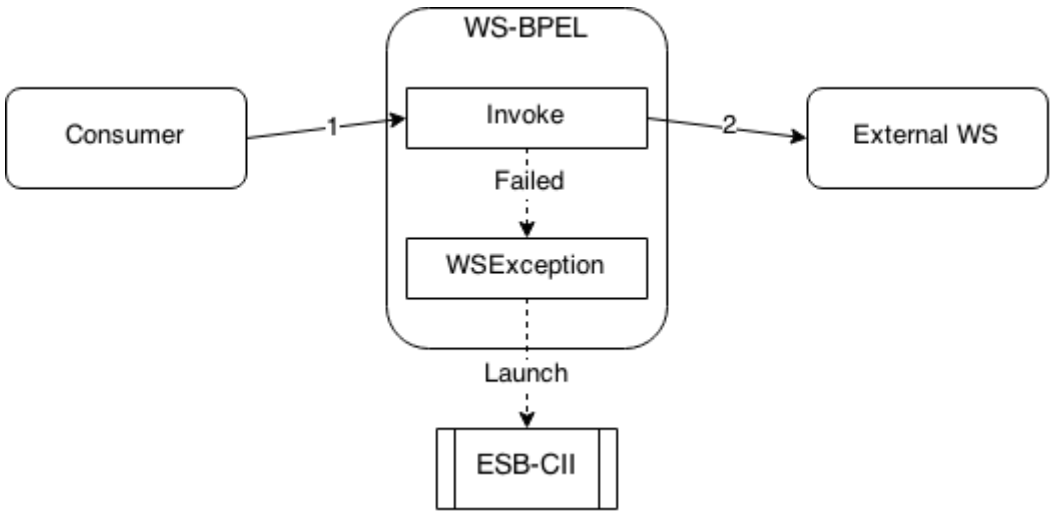

Figure 6.5 – The service description interface validation scenario.

Once again, the tests performed during this scenario validation, make use of the *Exceptions.java* class exceptions, namely the *WebServiceInvocationException*, allowing to immediately limit the cause of error and further provide it to the ESB-CII, which acts according to the provided exception at the time of its invocation.

When a WS-BPEL process is initially created, one must supply it with the WSDL files that concern the WSs that are to be consumed during execution. When this is firstly done and the project is built, it automatically copies the WSDL description files into local folders so that they can be used during operation. Because the process saves its own copies of the WSDL files, this means that if they are to change, the process will not be able to know it. While it is true that in a stable SOA-based network this situation should never happen, it is also true that it can indeed happen and therefore interoperable environments may suffer in consequence of such an unexpected event. While one should consider and prepare for such events, the modification of a WSDL concerning WSs that are operational, is a hazardous and unsafe circumstance that can result in a variety of consequences, since it may be very difficult to detect what the changes were and thus to adapt accordingly. Analogously, it is like changing the input parameters of programming API without warning the users and applications currently using it. Thus, the current scenario represents a difficult to approach situation that can only be tackled in very specific cases that must be previously accounted for when building the ESB-CII. Following the WS-BPEL process previously introduced, and in order to simulate such an event and properly test and validate the methodology capabilities concerning this situation, a modification to the WSDL file of the process's provider was made during normal operation by removing and re-introducing a new required input parameter as depicted in Figure 6.6.

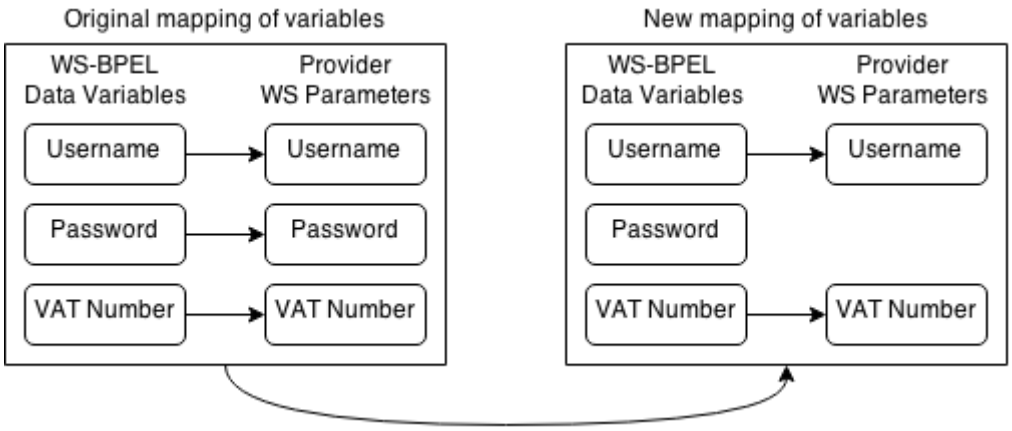

Figure 6.6 – Variable mapping in consequence of a WSDL modification.

Thus, in a first test, the password parameter was removed from the remote WSDL file in order to simulate a system adaptation, while in a second test the same parameter was re-inserted into the remote WSDL file. Table 6.4 describes the test results during the WS-BPEL normal operation.

Table 6.4 – Service description mismatch scenario functional test without the ESB-CII.

Test Case			
Test:	Service Description Mismatch Scenario without the ESB-CII		
Group:	Service Description Mismatch Scenario Validation		
Purpose:	Test the given scenario during normal operation.		
Comment:	N/A		
Behaviour		**Constraints**	**Verdict**
! Start WS-BPEL			
! Invoke *getCompany* WS			
? WS input parameters are correct			
? Receives WS response			
! Reply			Success
OTHERWISE			Failure
OTHERWISE			Failure

In the same way as the standard mismatch scenario validation, in the previous section, once the ESB-CII is invoked, it receives the resulting exception as an input parameter. If the exception is known to the component, then the latter can properly approach the cause of error as long as it is in fact prepared to do so. Otherwise, an initial solvable scenario is not possible and the alternative provider solution must be implemented. Concerning the test results during normal operation, in each situation that the process fails to succeed, then the ESB-CII is invoked and executed.

Table 6.5 describes the test results that were obtained in regard to the first case scenario, in which the password input parameter has been removed from the remote WSDL and thus prevents the process from consuming the *getCompany* WS. Because the ESB-CII had been previously enhanced with the solution for this specific situation, the WS-BPEL could be successfully adapted and re-deployed, further validating the methodological steps previously proposed in chapter 4.

Table 6.5 – Service description mismatch scenario functional test with the ESB-CII.

Test Case			
Test:	Service Description Mismatch Scenario with the ESB-CII		
Group:	Service Description Mismatch Scenario Validation		
Purpose:	Test the proposed methodology within the given scenario.		
Comment:	N/A		
Behaviour		**Constraints**	**Verdict**
! Start ESB-CII			
? Exception is provided			
? Exception is the *WSInvocationException*			
? Retrieves remote WSDL			
? Input parameters differ			
? Adapts BPEL			
? Re-deploys BPEL			Success
OTHERWISE			Failure
OTHERWISE			Failure
OTHERWISE			Failure
OTHERWISE			Failure
OTHERWISE			Failure
OTHERWISE			Failure

As mentioned, in order for the test result to be successful, the ESB-CII had to be adapted concerning this specific scenario. This means that to maintain the testing process simplified, this adaptation merely consisted in checking and comparing the three initial input parameters for the respective WS on each of the WSDL files and further modify the WS-BPEL process code accordingly, by introducing or removing the XML lines respective to the specific mappings as depicted by Figure 6.7. As it is possible to observe, the WS-BPEL XML code allows an understandable and simple presentation of the mappings currently in use for a given operation, enabling to easily modify it according to new requirements. Although the XML code concerning the mapping of the password parameter was removed in order to properly adapt it accordingly to the remote WSDL, the password variable itself continues to exist. This means that, by being possible to modify variable mappings without affecting the variables themselves, it is also possible to reverse previous mapping modifications as long as the exiting variables can be linked to the required parameters. Thus with this is mind, a second test case was performed concerning the given scenario, in which the remote WSDL was modified once again and the previously removed password parameter was re-introduced as a required input parameter.

Figure 6.7 – WS-BPEL XML code modification according to the new WSDL.

Because the WS-BPEL only removed the parameter mapping but maintained the respective variable during the previous iteration of the test case, it was then possible for the ESB-CII to associate both entities by comparing their names and easily re-introduce the previously disregarded mapping into the process's XML code, further validating the methodology in two different situations concerning the same scenario. The test results for this second test case are identical to the ones described in Table 6.5. The verdicts that resulted in failure concern the situations in which an automatic adaptation was not possible and thus human intervention is necessary.

Hence, this scenario presents a situation in which a design or modelling requirement is introduced into the system, since a re-design of the process is required in order to adapt, that forces the WS-BPEL to change accordingly. Because the ESB-CII was previously prepared to handle this very specific situation, the implemented tests were able to validate the proposed methodology in two different cases, one concerning the removal of a parameter from the WSDL

file and the other concerning the re-introduction of that same parameter. Thus, although this scenario concerns an event that should never take place in a CN, which is the modification of a given service description during normal system operation, it also proves the suggested methodology as a possible and efficient solution that can be used to tackle specific and previously defined cases. It should be reasserted however, that the situation concerning the second test case does not provide a reliable solution and should only be implemented in very elementary and specific situations, since the association between the required WSDL parameter and the WS-BPEL variable can only be done by name comparison. In the case here presented this basic solution was sufficient to successfully validate the methodology, since the amount of existing variables was very limited. However, in a large and complex business process where several variables may exist and large amounts of data are exchanged, this solution should be implemented with extreme care and only in situations where it is certain that the automatic association between variables is correct, since this association is simply done by name and can thus result in inconsistent connections between variables and parameters.

6.3 Scientific Validation

During the research and developments here presented, two scientific publications were made to further contribute to the validation of the achieved results, although at the time of writing one of these publications was still pending review. The publications are described as follows:

- *Raposo, C., Agostinho, C., Ferreira, J., & Jardim-Goncalves, R., 2014, "Automatic Detection of Harmonization Breaking in SOA-based Enterprise Networks", 21st ISPE International Conference on Concurrent Engineering (CE2014), pp. 726-735, Beijing, China.*

- *Agostinho, C., Raposo, C., & Jardim-Goncalves, R, 2015, "Sustaining Interoperability in the (Re)Engineering of Industrial Service-Based Systems", 35th Computers and Information in Engineering Conference (ASME15), Massachusetts, USA.* (Pending review)

6.4 Industrial Acceptance

To validate the proposed methodology within a real case scenario, a living lab, i.e. research environment, concerning the furniture sector was used, the Furniture Living Lab (FurnitureLL) within the IMAGINE project (IMAGINE Partners, 2013), described in section 5.2. With collaboration as an important aspect of industrial networks, the furniture sector is characterised by the wide variety of products and resources it provides, alongside short production lifecycles and a short time-to market. The i-platform eases this essential process of collaboration by

providing the means to quickly search for available partners, according to the wanted characteristics, and generate an optimised DMN with the best available solutions. To better understand how the IMAGINE concept integrates within the furniture sector, a brief fictional scenario is illustrated in Figure 6.8 and described as follows.

A table manufacturer in Europe wishes to fulfil a request to produce 100 customised tables. To achieve this, the manufacturer needs to produce both the table legs and the table top itself. If it were to use traditional means, the manufacturer would have to take a considerate amount of time to properly search for the specific producers for each requirement, evaluate the available options, and finally handle all the paperwork necessary to issue the production order.

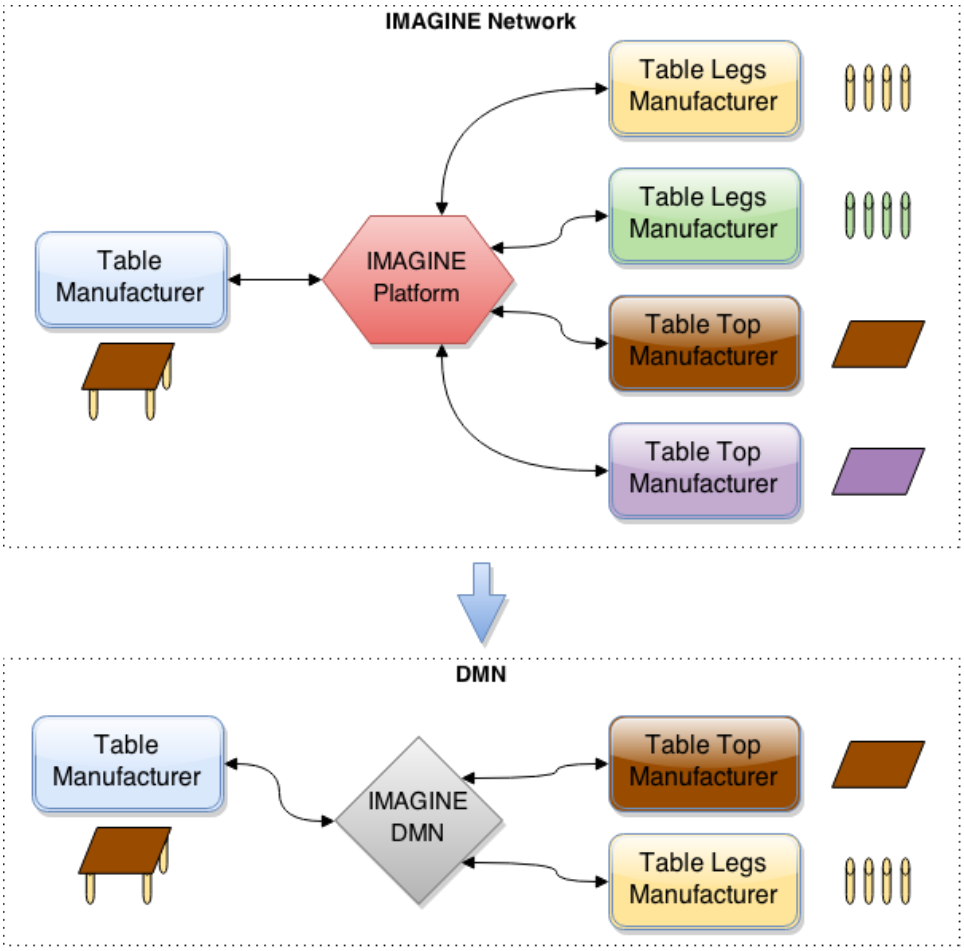

Figure 6.8 – IMAGINE Furniture Living Lab scenario.

By connecting to the i-platform however, the table manufacturer needs only to register its company and fill in the platform's provided forms with the pertinent information, such as the market region, legs and table top characteristics, etc., in order to automatically obtain a list of all

the relevant manufacturers available in the network, alongside detailed information concerning their production capabilities, prices, time-to-market, and others. After choosing the optimal partners, the platform then generates a DMN between the collaborating parties in order to fulfil the production request, allowing to consult detailed information about the state of operation, timelines, performance values and the like. In this way, the i-platform can effectively reduce the time-to-market while improving production speed and capabilities, allowing at the same time for a greater control and detailed network monitoring, being a most beneficial tool in the manufacturing domain.

6.4.1 Contribution of this dissertation

As mentioned in section 5.2.3, several implementations concerning the IMAGINE project were done during the elaboration of this master thesis. Since the latter proposes a methodological framework for the detection of harmonisation breaking in service-based networks, previously validated regarding two different proof of concept scenarios, the results achieved could then be applied to the i-platform itself, in order to further enhance its monitoring capabilities as well as to contribute to the system's ability to sustain interoperability. In this sense, the proposed methodology was effectively implemented within the platform and the ESB-CII was adapted to handle a specific harmonisation breaking situation, similar to the one described in the service description mismatch scenario in section 5.1.2, in which a single WS invocation is monitored for changes concerning the mapping between its variables and the WS's input parameters. While it remained simple, this implementation served to strengthen the legitimacy of the proposed methodology within a real world environment.

6.5 HYPOTHESIS VALIDATION

In section 1.2 the research questions concerning the work here described were presented, followed by the hypothesis comprising a possible solution to the issues raised by those questions, as quoted bellow:

- *"In a collaborative service-based network, if one can effectively monitor the interactions that occur between different entities, then one can also strive to timely intervene in the event of harmony disruptions, seeking in this manner to maintain and uphold sustainable interoperability across the network and prevent harmonisation breaking."*

Following the results and observations previously described, one can conclude that the described hypothesis has been validated. This master thesis proposes in chapter 4 a methodological framework for the detection of harmonisation breaking in service based networks. This methodology seeks to contribute to the sustainability of interoperability by enhancing SOA-

based systems with improved monitoring. Furthermore, besides its monitoring capabilities, the methodology also suggests a set of steps that may be taken in order to recover from breaks in the system's interoperability, without needing to resort to human intervention. Thus, following the implementation and validation results described in chapter 5 and 6 respectively, it can be affirmed that its purpose has been successfully achieved. After implementation, the suggested methodology was able to successfully recover from two different harmonisation breaking events without needing to request human intervention. While this does not guarantee a flawless system ready to be commercialised, it does prove its capability to provide support for the sustainability of interoperability in service-based networks and reasserts that further research and development should be considered.

Chapter 7

"The strongest arguments prove nothing so long as the conclusions are not verified by experience. Experimental science is the queen of sciences and the goal of all speculation." - Roger Bacon

7 CONCLUSIONS AND FUTURE WORK

For a dynamic enterprise network to operate efficiently, collaboration is essential. This means that interoperability must be continuous and sustained among all the involved entities. However, as different members of the network adapt to different requirements, the system experiences breaks in harmonisation. A break in the system's harmony means that a part of the system is no longer interoperable, that is, members are unable to effectively communicate and therefore, the network is no longer efficient.

Thus, in order to maintain sustainable interoperability, a proper monitoring of the network is crucial. Monitoring is usually performed alongside normal operation and consists in the gathering of data and information concerning the system state and behaviour. For this purpose, several solutions possessing different attributes and offering distinct tools exist and allow to effectively ease this process. However, several issues concerning the effects of monitoring in dynamic networks still call for the need of further research and investigation. Furthermore, few solutions offer the capability to not only monitor and detect faults as they occur, but also to automatically react to them and consequently adapt in real-time. For complex networks such as DMN which operate in a chain-like manner, this represents an inefficient and unwanted behaviour, since a disruption in interoperability between two companies may end up affecting several partners and thus harm the network as a whole. While the work performed during this master thesis elaboration does not aim to solve and fulfil all these missing capabilities, it strives to provide a strong basis of study on the domain of manufacturing monitoring as well as to contribute to the ability of automatically reacting to unexpected behaviour in service-based networks.

Hence, this thesis proposes a methodological framework for the detection of harmonisation breaking in service-based networks that builds upon the concept of a good SE process in order to

be able to adapt the system's business processes on the event of emerging requirements. Based on a simple procedure of error handling, the methodology focuses on the ability to handle exceptions as they occur and consequently invoke an external component to address the cause of error and, if possible, to adapt the erroneous business process accordingly. Its implementation is based on a centralised solution, the ESB, which offers greater control and reliability, allowing to manage and orchestrate all the interactions that occur between different partners and their respective WSs while keeping an eye on what's being exchanged and handling faults as they happen. As mentioned, by individually handling each fault, each business process is thus able to invoke an external component, the ESB-CII, which consists on an application located somewhere in the system that acts as a recovering agent capable of analysing and, if possible, automatically recovering from the issue in question.

To implement and validate the suggested methodology, the OpenESB was used to serve as the communication bus and provide a server engine to deploy business processes, which in this case were developed in the WS-BPEL language. Hence, once the fundamental architecture was installed, a single WS-BPEL process consisting of a simple interaction between a consumer and a provider was deployed within the server engine. To validate the effects of the proposed methodology on this implementation, two proof of concept scenarios were considered. The first – the standard mismatch scenario – consisted in the occurrence of a harmonisation breaking event resulting from a version mismatch between a standard or programming library shared by the two entities, while the second – the service description mismatch scenario – consisted in the occurrence of a harmonisation breaking event resulting from an invalid invocation of a given WS due to the incorrect mapping between the process's variables and the WS input parameters. Both scenarios were implemented through the WS-BPEL process and tested during normal and abnormal operation, which consisted in the simulation of the events leading to the harmonisation breaking events in test. As it was shown, in both situations the ESB-CII was able to automatically adapt and recover the faulty business process, provided that the necessary conditions were met, thus successfully validating the methodological steps proposed.

The use of specific and customised exceptions allowed to filter the cause of the error at its origin, providing the ESB-CII with detailed information concerning the possible issue and thus enabling it to properly approach it. Furthermore, because the WS-BPEL language is an XML-based language, it further facilitated the modification of the business process during the performed tests. As the first proof of concept scenario concerned a technical requirement that could be simply resolved by updating the library in question, there was no need to re-design the business process and modify its XML code. However, the same was not the case in the second proof of concept scenario, in which the mapping between variables and parameters was incorrect due to a change in the remote WS WSDL file. In this situation, a re-design of the WS-BPEL process itself was

required and a modification concerning the corresponding XML code had to be performed. Because the ESB-CII was programmed to approach this very specific situation with these specific variables and parameters, a successful validation could be achieved. However, future implementations concerning this second case scenario should be performed with extreme care and only in situations where it may be evident what the correct link between the variables and the parameters may be. While this situation in which a provider suddenly decides to modify the WSDL description of its WSs should never happen in a stable SOA-based network, the fact that it can indeed happen and consequently provoke a break in system interoperability, is a sufficient reason to prove that further testing and techniques to approach this type of scenario should be developed. During the described validation, two possible situations concerning this specific scenario were identified. On the first situation, the provider changes the WSDL of its WSs by removing a required parameter. This situation can be safely approached since there is no need to create a link between variables and parameters but rather remove an existing one. However, in a second possible situation, the provider changes the WSDL of its WSs but instead of removing a required input parameter, it introduces a new one. In order to approach this second situation, a variable that can be correlated with the introduced parameter must already exist in the process, which may be the case if this situation happens after the previous one and the introduced parameter is the same that was previously removed, since the removal of mappings between variables and parameters does not imply the removal of variables and their data. Nevertheless, when a new input parameter is introduced, the ESB-CII can only look at the existing variables within the WS-BPEL process and, if it finds one which name matches the name of the required parameter, automatically create the mapping between the two entities. As mentioned before, because this association is made by name comparison only, it should only be implemented in very specific scenarios when it is certain that the existing data will not be sent to the wrong place. In summary, while the first proof of concept scenario proved to be a reliable example in which the proposed methodology may serve as a useful solution, the second proof of concept scenario should mainly be considered when the WSDL modification consists on the removal of an existing input parameter of one of the described WSs, and not on the introduction of a new one. In either way, the methodology hereby proposed was validated in both the proof of concept scenarios considered. Moreover, the second proof of concept scenario was further validated by being implemented within a real world environment by means of the IMAGINE project.

By observing the results previously illustrated, one can conclude that while it may be simple to monitor a business process and effectively handle occurring exceptions, automatic recuperation is another matter that should be addressed individually. In the case here described, this concerns the ESB-CII component, which is the responsible process for accessing the cause of error and promptly correct it. Without it, the methodology would merely be an error handling

process with little use whatsoever. In the first proof of concept scenario, the ESB-CII implementation was simple to achieve as the solution merely consisted in copying a library file from one location to the other before re-compiling and re-deploying the process once again. However, the second scenario was more complex and the solution was not that straight forward, since it required the modification of the process's own XML code. This proved to be a much more complicated situation that was only possible to address by specifically programming the ESB-CII for this specific scenario. This means that while the proposed methodology may prove to be very efficient in some scenarios, in what concerns the automatic recuperation of business processes, in other situations it may be difficult to intervene without human intervention. In this manner, this capability concerning the ESB-CII of automatically addressing design and modelling requirements should be considered for future research, besides other remaining characteristics such as an improved ability to relate local with remote information, the ability to predict the impact of a suggested solution on the current process and on the involved entities (e.g. whether the process can continue from the point where the fault occurred or if it has to restart completely), the ability to integrate with the different features offered by the ESB, between others.

While the proposed methodology was successful concerning the previously described scenarios, further tests and developments have to be implemented in order to fully validate its capabilities, namely those concerning the secondary solution of searching for an alternate service provider in case the process adaption does not succeed. Due to timing restrictions and other limitations, this methodological procedure could not be implemented and it also remains for future work and research, posing a different set of problems such as the differences that may exist between different providers, the capability to seamlessly connect with a provider other than the current one amidst a running process, or the limitations that may exist when automatically modifying the WS-BPEL code.

While further research and development needs to be performed, the results obtained during the elaboration of this master thesis present a step forward in the approach to automatically handle harmonisation problems in dynamic enterprise networks after interoperability is first achieved, thus reducing the transition time between equilibrium states and providing additional support for current and future SOA-based systems in maintaining interoperability and efficiency.

8 REFERENCES

Agostinho, C. (2012). Sustainability of Systems Interoperability in Dynamic Business Networks. *Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa (FCT-UNL)*, Available from: <http://hdl.handle.net/10362/8582>.

Agostinho, C., & Jardim-Goncalves, R. (2009). Dynamic Business Networks: A Headache for Sustainable Systems Interoperability. *On the Move to Meaningful Internet Systems: OTM 2009 Workshops*, pp. 194-204.

Agostinho, C., Malo, P., Jardim-Goncalves, R., & Steiger, A. (2007). Harmonising Technologies in Conceptual Models Representation. *International Journal of Product Lifecycle Management*, 2 (2), pp. 187-205.

Ahmed, Z. (2006). A Middleware Road towards Web (Grid) Services. *In the proceedings of Blekinge Institute of Technology Student Workshop on Architectures and Research in Middleware (BITSWARM)*, pp. 67-74.

Atzori, L., Lera, A., & Morabito, G. (2010). The Internet of Things: A Survey. *Computer Networks*, 54 (15), pp. 2787-2805.

Baresi, L., Guinea, S., & Plebani, P. (2006). WS-Policy for Service Monitoring. *Technologies for E-Services*, pp. 72-83.

Barringer, B., & Harrison, J. (2000). Walking a Tightrope: Creating Value Through Interorganizational Relationships. *Journal of Management*, 26 (3), pp. 367-403.

Beca, M., Sarraipa, J., Agostinho, C., Gigante, F., Nunez, M., & Jardim-Goncalves, J. (2014). A Framework to Enhance Supplier Search in Dynamic Manufacturing Networks. *4th International Conference on Information Society and Technology (ICIST)*, pp. 16-21.

Bernstein, P. (2003). Applying Model Management to Classical Meta Data Problems. *1st Biennial Conference on Innovative Data Systems Research*, pp. 209-220.

Calisti, M., Leymann, F., Dignum, F., R., K., & Unland, R. (2010). Service-Oriented Architectures and (Multi-)Agent Systems Technology. *Dagstuhl Seminar 10021*.

Camarinha-Matos, L., & Afsarmanesh, H. (2005). Collaborative Networks: A New Scientific Discipline. *Journal Of Intelligent Manufacturing*, 16 (4-5), pp. 439-452.

Chappell, D. (2004). *Enterprise Service Bus: Theory in Practice*. California, USA: O'Reilly Media, Inc.

Chen, D., & Vernadat, F. (2002). Enterprise Interoperability: A Standardisation View. *ICEIMT*, pp. 273-282.

Chen, I.-Y., Ni, G.-K., & Lin, C.-Y. (2008). A Runtime-Adaptable Service Bus Design for Telecom Operations Support Systems. *IBM Systems Journal*, 47 (3), pp. 445-456.

Commission of the European Communities. (2005). *i2010 - A European Information Society for Growth and Employment.* Brussels, Belgium: European Comission.

Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2011). *Distributed Systems: Concepts and Design (5th Edition).* Boston, USA: Addison-Wesley.

Couture, M. (2007). *Complexity and Chaos – State-of-the-Art; Overview of Theoretical Concepts.* Quebec, Canada: Technical Memorandum DRDC Valcartier TM 2006- 453, Defence R&D Canada – Valcartier.

Curbera, F., Khalaf, R., Mukhi, N., Tai, S., & Weerawarana, S. (2003). The Next Step in Web Services. *Communications of the ACM - Service-Oriented Computing*, 46 (10), pp. 29-34.

Curry, E. (2004). Message-Oriented Middleware. *Middleware for Communications*, pp. 1-26.

Erl, T. (2005). *Service-Oriented Architecture: Concepts, Technology, and Design.* New Jersey, USA: Prentice Hall.

ETSI. (2009). Methods for Testing and Specification (MTS) - The Testing and Test Control Notation version 3 - Part 1: TTCN-3 Core Language. *ETSI ES 201 873-1*.

European Comission. (2008). *Enterprise Interoperability Research Roadmap (Version 5.0).* Brussels, Belgium: European Comission.

Ferreira, J., Agostinho, C., Sarraipa, J., & Jardim-Goncalves, R. (2011). Monitoring Morphisms to Support Sustainable Interoperability of Enterprise Systems. *In 6th Int. Workshop EI2N'2011 of OTM Conference, Greece*, pp. 71-82.

Ferreira, J., Agostinho, C., Sarraipa, J., & Jardim-Goncalves, R. (2012). Monitoring Morphisms to Support Sustainability of Interoperability in the Manufacturing Domain. *Information Control Problems in Manufacturing*, 14 (1), pp. 1264-1271.

Fielding, R. (2000). Architectural Styles and the Design of Network-based Software Architectures. *University of California*, 5, pp. 76-105.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., . . . W3C/MIT. (1999). Hypertext Transfer Protocol -- HTTP/1.1. *IETF - RFC 2616*.

Holland, J. (1992). Complex Adaptive Systems. *Daedalus*, 121 (1), pp. 17–30.

Ibrahim, N., & Hassan, M. (2010). A Survey on Different Interoperability Frameworks of SOA Systems towards Seamless Interoperability. *Information Technology*, 3, pp. 1119-1123.

IEEE Computer Society. (1990). IEEE Standard Glossary of Software Engineering Terminology.

IMAGINE Partners. (2013). *Living Lab in the Furniture Manufacturing Domain Report.* Brussels, Belgium: European Commission - IMAGINE Project (FoF-ICT-2011.7.3).

ISO. (2000). Industrial Automation Systems - Requirements for Enterprise-reference. *ISO 15704*.

ISO/IEC. (1994). Information Technology - Open Systems Interconnection - Conformance Testing Methodology and Framework - Part 1: General Concepts. *ISO/IEC 9646-1*.

ISO/IEC. (2001). Software Engineering - Product evaluation - Part 6: Documentation of Evaluation Modules. *ISO/IEC 14598-6*.

ISO/IEC. (2007). Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Quality Requirements. *ISO/IEC CD 25030*.

ISO/IEC. (2012). Systems and Software Engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Evaluation guide for developers, acquirers and independent evaluators. *ISO/IEC 25041*.

iSurf Partners. (2009). *Deliverable 8.1.1 – Functional and Non-Functional Evaluation Criteria for i-Surf Components and Integrated Platform.* Brussels, Belgium: European Commission - iSurf STREP Project (FP7 213031).

Jardim-Goncalves, R., Agostinho, C., & Steiger, A. (2012). A Reference Model for Sustainable Interoperability in Networked Enterprises: towards the Foundation of EI Science Base. *International Journal of Computer Integrated Manufacturing*, 25 (10), pp. 855-873.

Jardim-Goncalves, R., Grilo, A., & Steiger, A. (2006). Challenging the Interoperability between Computers in Industry with MDE and SOA. *Computers in Industry*, 57(8-9), pp. 679-689.

Klotz, B., & Ackermann, U. (2009). *US Patent No. US 7583613 B2.*

Kossiakoff, A., Sweet, W., Seymour, S., & Biemer, S. (2011). *Systems Engineering Principles and Practice.* New Jersey, USA: John Wiley & Sons, Inc.

Lebreton, B., & Legner, C. (2005). Introduction to Business Interoperability. *ATHENA - Training in Interoperability*.

Lee, J., Shim, H., & Kim, K. (2010). Critical Success Factors in SOA Implementation: An Exploratory Study. *Information Systems Management*, 27 (2), pp.123-145.

Lee, Y. (2010). Interoperability Management for SOA by Relative Proof and Test Suit. *2010 6th International Conference on Advanced Information Management and Service*, pp. 515-520.

Markaki, O., Panopoulos, D., Kokkinakos, P., Koussouris, S., & Askounis, D. (2013). Towards Adopting Dynamic Manufacturing Networks for Future Manufacturing: Benefits and Risks of the IMAGINE DMN End-to-End Management Methodology. *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pp. 305-310.

Menge, F. (2007). Enterprise Service Bus. *Free and Open-Source Software Conference*, pp. 1-6.

Mirashe, S., & Kalyankar, N. (2010). Cloud Computing. *Journal of Computing*, 2 (3), pp. 78-82.

Murray, R. (1999). The Scientific Method. *Analytical Chemistry*, 71 (5), pp. 153A-153A.

Mylopoulos, J. (1998). Information Modeling in the Time of the Revolution. *Information Systems*, 23 (3-4), pp.127–155.

Newman, H., Legrand, I., Galvez, P., Voicu, R., & Cirstoiu, C. (2003). MonALISA: A Distributed Monitoring Service Architecture. *Conference for Computing in High Energy and Nuclear Physics*, California, USA.

Nicolis, G., & Prigogine, I. (1989). *Exploring Complexity: An Introduction.* London, UK: W.H. Freeman and Company.

Noran, O. (2012). Achieving a Sustainable Interoperability of Standards. *Annual Reviews in Control*, 36 (2), pp. 327-337.

OASIS. (2004). UDDI Version 3.0.2. *UDDI Spec Technical Committee Draft*.

OASIS. (2007). Web Services Business Process Execution Language 2.0. *OASIS Standard*.

Ogren, I. (2000). On Principles for Model-Based Systems Engineering. *Systems Engineering Journal*, 3 (1), pp.38-49.

OMG. (2003). MDA Guide Version 1.0.1. *Object Management Group*.

OMG. (2011). Business Process Model and Notation (BPMN) Version 2.0. *Object Management Group*.

Ou, T., Sun, W., Guo, C., & Li, J. (2008). Visualized Monitoring of Virtual Business Process for SOA. *IEEE International Conference on e-Business Engineering (ICEBE)*, pp. 767 - 770.

Owen, M., & Raj, J. (2003). *BPMN and Business Process Management: Introduction to the New Business Process Modeling Standard.* New York, USA: Popkin Software.

Papakostas, N., Efthymiou, K., Georgoulias, K., & Chryssolouris, G. (2012). On the Configuration and Planning of Dynamic Manufacturing Networks. *Logistics Research*, 5 (3-4), pp. 105-111.

Peltz, C. (2003). Web Services Orchestration and Choreography. *IEEE Computer Society*, 10, pp. 46-52.

Pine, B., & Gilmore, J. (1999). *The Experience Economy.* Massachusetts, USA: Harvard Business Press.

Pooley, R., & King, P. (1999). The Unified Modelling Language and Performance Engineering. *IEE Proceedings - Software*, 146 (1), pp. 2-10.

Psiuk, M., Bujok, T., & Zielinski, K. (2012). Enterprise Service Bus Monitoring Framework for SOA Systems. *IEEE Transactions On Services Computing*, 5 (3), pp. 450-466.

Ray, S., & Jones, A. (2006). Manufacturing Interoperability. *Journal of Intelligent Manufacturing*, 17 (6), pp. 681—688.

Sampson, S., & Froehle, C. (2006). Foundations and Implications of a Proposed Unified Services Theory. *Production and Operations Management*, 15 (2), pp. 329–343.

Schafersman, S. (1997). *An Introduction to Science: Scientific Thinking and the Scientific Method.* Miami, USA: Online Whitepaper.

Schmidt, D. (2006). Model-Driven Engineering. *IEEE Computer Society*, 39 (2), pp. 25-31.

Spohrer, J., Maglio, P., Bailey, J., & Gruhl, D. (2007). Steps toward a Science of Service Systems. *IEEE Computer Society*, 40 (1), pp. 71-77.

Sundaram, D., & Wolf, E. (2009). Enterprise Model Management Systems. *EOMAS '09 Proceedings of the International Workshop on Enterprises & Organizational Modeling and Simulation*, Article No. 5.

Taxen, L. (2012). Sustainable Enterprise Interoperability from the Activity Domain Theory Perspective. *Computers in Industry*, 63 (8), pp. 835-843.

Ten-Hove, R., & Walker, P. (2005). Java Business Integration (JBI) 1.0. *Sun Microsystems, Inc., Specification: JSR 208*.

Thoben, K., & Jagdev, H. (2001). Typological Issues in Enterprise Networks. *Production Planning & Control*, 12 (5), pp. 421-436.

Tien, J., & Berg, D. (2003). A Case for Service Systems Engineering. *The Journal of Systems Science and Systems Engineering*, 12 (1), pp. 113-28.

W3C. (2007). SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). *W3C Recommendation 27 April*.

W3C. (2007). Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. *W3C Recommendation 26 June*.

W3C. (2007). Web Services Policy 1.5 - Framework. *W3C Recommendation 04 September*.

W3C XML Core Working Group (WG). (2008). Extensible Markup Language (XML) 1.0 (Fifth Edition). *W3C Recommendation 26 November*.

Wang, Q., Shao, J., Deng, F., Liu, Y., Li, M., Han, J., & Mei, H. (2009). An Online Monitoring Approach for Web Service Requirements. *IEEE Transactions on Services Computing*, 2 (4), pp. 338 - 351.

Wooldridge, M. (2009). *An Introduction to Multi-Agent Systems.* New York, USA: John Wiley & Sons, Inc.

Wooldridge, M., & Jennings, N. (1995). Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review*, 10 (2), pp. 115-152.

Wymore, A. (1993). *Model-Based Systems Engineering.* Florida, USA: CRC Press.