AN AGENT-BASED SELF-CONFIGURATION METHODOLOGY FOR MODULAR ASSEMBLY SYSTEMS

Pedro Ferreira

Thesis submitted to the University of Nottingham for the degree of
Doctor of Philosophy

April 2011

To my mother, Maria Adelaide Monteiro dos Santos, without whom I would not have the opportunity to do this work, and to my girlfriend, Filipa Antunes, without whom I would not have completed it

Para a minha mãe, Maria Adelaide Monteiro dos Santos, sem a qual eu não teria tido a oportunidade de desenvolver este trabalho, e para a minha namorada, Filipa Antunes, sem a qual eu não o teria completado

Abstract

Assembly systems today are exposed to market trends that have become increasingly more dynamic and unpredictable, requiring product changes and adjustments which emphasise de need for more flexible systems. The requirement for increased responsiveness has led to the development of new modular concepts which provide the bases for achieving higher system adaptability through increased component/module interchangeability and reusability. The modularization of physical and control infrastructure does, however, only address one aspect of the issue and there is still a lack of appropriate tools and methods to support the rapid configuration and reconfiguration of such systems for changing sets of requirements.

This work proposes a new distributed methodology for the configuration and reconfiguration of Modular Assembly Systems (MAS) through the use of agent technology. The new methodology defines a comprehensive model for the structured description of the MAS requirements, equipment modules and the configuration results.

This thesis proposes a new agent architecture for the self-configuration of equipment modules into systems based on a given set of requirements, as the core of the self-configuration methodology. This architecture introduces the overall behaviour of the methodology through the definition of agent types, roles and overall interactions. Furthermore this work describes the development of the specific models and methods for the local behaviour of each agent. These enable the actual decision making method for the agents to achieve configuration solutions.

This work also reports on a new methodology for the early performance simulation of MAS characteristics that can be used in conjunction with the configuration methodology.

Acknowledgments

I would like to take the opportunity to thank a few number of people without whom this work would not have been possible. The first two people I would like to thank are my supervisors Svetan Ratchev and Niels Lohse, for the opportunity to pursue this venture, for their time and patience to support me along the way and most importantly for their belief and trust in my abilities.

I would like to convey my deepest thanks to my Mom, Maria Adeleide Monteiro dos Santos, and my girlfriend, Filipa Antunes, to whom I also dedicate this work. Without them, this endeavour would have surely failed.

I also would like to thank all my colleagues in the Precision Manufacturing Centre, for all their support across the years, I am very proud to have you all as colleagues. I would like to add an extra thank you to my colleagues in C32 (the PhD office), for their motivation and useful discussions; this work, in a way, is from all of us.

I would like to thank the members of the EUPASS project for their support, guidance and useful discussions, this work could not be completed without their input. Furthermore I would like to thank the European commission for funding this work through the EUPASS project.

Finally, I would like to convey a warm thanks to my girlfriend, Filipa Antunes (her support is beyond words), and Eirini Panagiotidou, for their friendship, patience, support and for proof reading this thesis. Furthermore I would like to add a big thanks to my supervisor, Niels Lohse, for his dedication in providing detailed critical reviews for this thesis.

Table of Contents

Table of Figures

1 Introduction

Current manufacturing systems require an increasingly higher responsiveness due to the market demand for increasing product diversity leading to mass customization, shorter product lifecycles and lower times to market while maintaining the cost to the minimum and quality to a maximum. Nowadays, markets are truly global and are characterized by an intensive global competition which is conditioned by socio-economic aspects that influence the manufacturing systems. In addition to this, market trends have become increasingly more dynamic and unpredictable, requiring product changes and adjustments which emphasise the need for more flexible manufacturing systems.

The issue of flexibility in manufacturing systems is not new and has been one of the main research topics in the field of manufacturing, namely Flexible Manufacturing Systems (FMS), which provided the first concepts to introduce bigger flexibility through mainly the increase of the systems capabilities "just in case" and adding cost to the system (Shen et al. [1]). This concept provides extra flexibility, but it is restricted to what can be predicted to be needed in the future. This raised other research questions on how to have a more flexible system that is able to deal with the market needs, without adding to it redundant equipment that might never be used.

Moreover, the market's volatility has led the systems' lifecycle to be shortened and also to the need for a rapid system design and configuration to cope with quicker speeds to market, thus reducing the system profitability. To cope with this, the issues of system reusability, rapid configuration and reconfiguration were introduced in the research.

The concept of Reconfigurable Manufacturing Systems (RMS) was introduced focusing mainly on the control reconfiguration of the systems to increase their lifecycle (Koren et al. [2]). Other approaches were also developed to deal with these issues, namely using the concept of "Plug & Produce" to create a holonic production system (Arai et al. [3]). These approaches provided some solutions to respond to the market needs, although these were very control driven and provided limited application on global systems.

Manufacturing is a very wide research topic which increases the complexity of solving all its inherent problems. To deal with this complexity there are several topics within manufacturing that are addressed individually. Assembly is one of these aspects and plays a key role in the problems previously identified and as such a lot of the research effort has focused on it.

The concept of equipment modularity is not a recent issue, but it is a quite complex issue due to the inherent nature of global competition between equipment suppliers. Nevertheless, this has been identified as a key aspect towards achieving the full concept of "Plug & Produce" (EUPASS [4]). Several issues have been identified as needs by roadmaping activities (**Figure 1.1**).

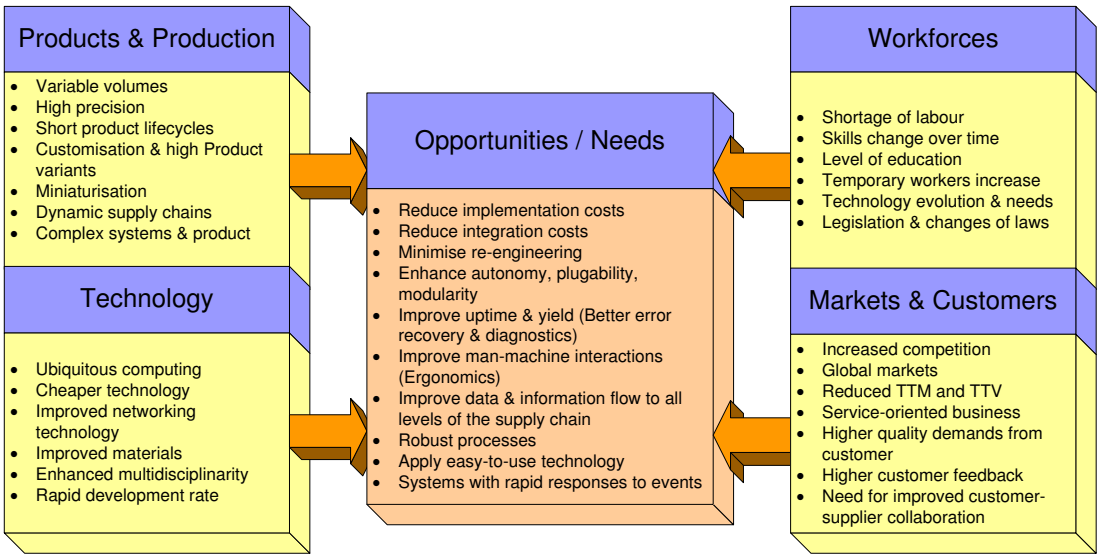


Figure 1.1.- Trends and needs within the domain of assembly systems (based on roadmap (EUPASS [4]))

Other roadmaps also highlight the need for a shift to software based on configuration and reconfiguration allowing functionality to be changed in ways not anticipated in the original system design (NACFAM [5]).

Moreover, the question of component reusability, rapid configuration and reconfiguration to enable "Configure to order" of assembly systems has become increasingly more important due to ever decreasing product life-cycles and rising process complexity. General purpose assembly machines, equivalent to CNC machine tools, are only available in specialist domains such as printed circuit board assembly where the components are highly standardised. The assembly of most other products demand custom made systems which address the specific requirements for these products. Today, these are mostly "Engineered to Order" making them cost and time intensive to design or reengineer. Increased modularisation of assembly equipment, rapid integration and design tools are considered fundamental for the move towards cost and time effective configuration and re-configuration of complex assembly systems (Koren et al. [2]; Kratochvíl and Carson [6]; Onori et al. [7]). **Figure 1.2** provides an overview of this trend and its business drivers.

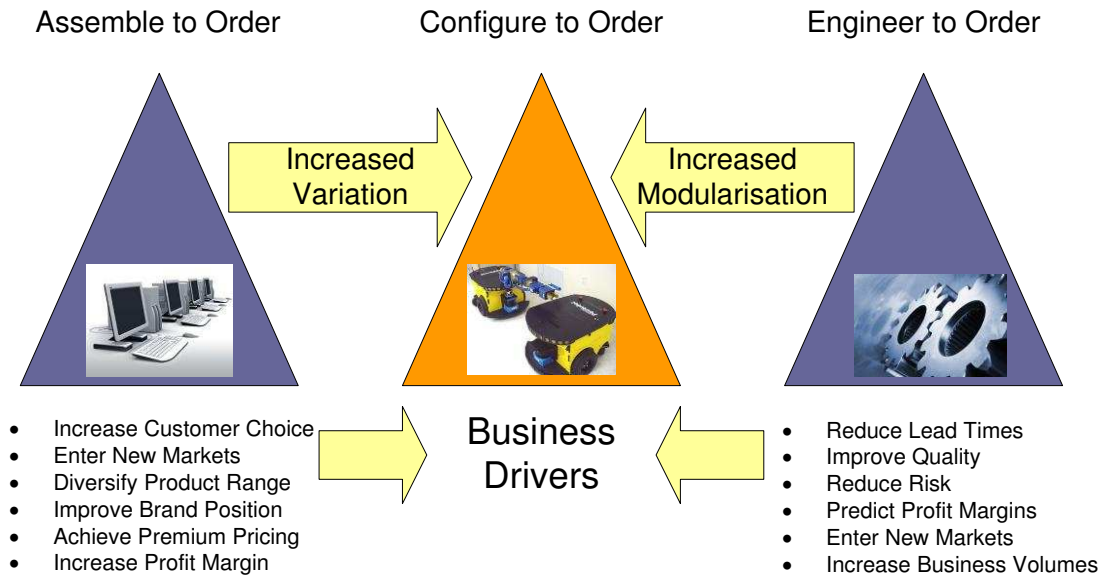


Figure 1.2 - Three basic concepts for developing producing and marketing complex services and products (based on (Kratochvíl and Carson [6]))

Significant effort has been directed towards creating modular assembly system (MAS) architectures for physical equipment and control interchange ability. The EU project EUPASS has created a framework for rapid integration for ultra-precision

assembly modules defining hardware interfaces, control interfaces, and module description formats (EUPASS [4]). Several other modular assembly system platforms have been proposed (Hollis and Quaid [8]; Alsterman and Onori [9]; Gaugel et al. [10]). While the number and completeness of underlying industrial applicable standards is still limited, there is a clear drive to overcome this barrier.

Standardisation of hardware and software interfaces is, however, only one aspect of rapid assembly system configuration. Effective tools and methods for the requirements driven selection, integration and validation of complex assembly system solutions are also needed to drastically reduce the time and effort required for the development of highly dedicated assembly systems. Most configuration methods reported today adopt a top down approach providing either methods for stepwise decomposition of the given set of requirements and subsequent solution synthesis or methods for the adaptation of similar system solutions. These approaches are often limited in their scalability and extendibility making them inappropriate and too complex for most MAS configuration problems.

In this context, the vision of this work is to provide the means to support this configuration and reconfiguration process through the creation of a methodology that is able to analyse and provide a set of viable solutions to the system integrator. This will reduced the time required for the analyses of the problem while increasing the speed and quality of configuration and reconfiguration solutions.

1.1 Research Scope

The objectives of this research are the enhancement of modular assembly systems by developing a Self-Configuration Methodology. This development will result in rapid system configurability and reconfigurability based on a set of requirements using a bottom-up approach. This approach is enabled by current advances in the MAS field with the maturity of knowledge models for requirements definition and the equipment modules standardization, which can be used as the basic building blocks of a Self-Configured modular assembly system.

The number of possible combinations of modules required for an assembly system solution depends on the number of available modules, their connection constraints

and the complexity of the given assembly process requirements. The number of combinations becomes quite large, even for relatively small problems, making configurations based on exhaustive enumeration practically infeasible. For this reason, an appropriate MAS configuration methodology needs to be more goal-oriented. Furthermore, any method should be able to exploit the specificities of the MAS configuration problem to reduce the search space.

The use of a bottom-up approach simplifies the system description, but it requires the development of methods that ensure the overall system required capabilities. To deal with this the development of negotiation methods that enable the module representations to interact and establish relations that enable the Self-configuration are proposed. This includes the need for a collaboration model, coalition rules and conflict resolution methods.

The use of agent technology has been indicated in the literature as one of the best ways to address bottom up problems (Jennings and Wooldridge [11]). Moreover, it provides the right structuring methods and capabilities to model modular systems without the need to model the complete system which in complex systems is very difficult. It also allows for the modelling of low level rules for different actors (agents) that can interact with each other creating complex models supported by these simple rules. To use agent technology the conceptualization of agent shells for different types of actors is proposed to act as complete representation of the modules. This includes the different models that use the negotiation methods towards building coalitions of modules that are able to fulfil the requirements, which will require organizational models and knowledge models for agent decision making capabilities. This raises the need for intelligent and decision making capabilities that ensure validation to guarantee system consistency.

The decision making methods for MAS configuration solutions without early performance assessments would result in the exploration of potential solutions that are not optimal. However, assessment tools and methods in the domain are not fully matured. Therefore, this work also intends to provide a method that is able to be enhanced in the future through the introduction of new agent types, which will contain the ability to assess the MAS configuration solution using different methods.

The existence of a methodology that is able to be enhanced in the future is viewed as a good decision in a constantly evolving domain.

In essence, the proposed self-configuration methodology targets the automatic MAS configuration driven by a set of process requirements, through the use of agent technology. This is a new approach for MAS configuration solutions, providing a step change from the current manual configuration process, which restricts the growth of the MAS concept, and is expected to allow quicker and more effective configuration even when large sets of modules are available.

1.2 Aim and Objectives

The aim of this work is to provide a configuration and reconfiguration methodology that enables the automatic configuration of MAS given a set of requirements. To reach such a methodology it is necessary to capture the configuration relevant information as well as the MAS equipment. This highlights the need for a clear model that is able to structure, in a transparent manner, all the aspects related to the MAS configuration process. Once this model is created, the analysis can be focused on the specific methods and models for the configuration process. This work proposes the creation of a bottom up configuration methodology using distributed decision making that enables configuration solutions to emerge based on predefined set of configuration rules and constraints. Later on, an agent architecture is proposed as the overall model which executes the distributed decision making methods and is able to achieve configuration solutions based on a set of requirements.

The following more detailed objectives provide a clear structure of the aims of the work:

- Development of methodology formal description models
    - Overall model of dependencies between concepts
    - Provide a uniformed terminology
    - Provide a clear assembly process taxonomy
    - Establishment of a MAS Requirements model
    - Establishment of a Equipment Module description model
- Development of distributed decision making framework

    - Development of Agent Architecture
        - Definition of agent types
        - Agent definition
        - Definition of agent roles
        - Definition of organizational model
        - Definition of overall agent behaviour
- Development agent behaviour models for distributed decision making
    - Definition methods that enable the fulfilment of agent roles
    - Definition of communication protocols
    - Definition of distributed configuration assessment methods

1.3 Approach and Structure of the Thesis

The research work was motivated by the European project EUPASS (Evolvable Ultra-Precision Assembly Systems) which targets the development of innovative micro-assembly modules and processes, accompanied by the standards and Industrial Technical Agreements, underlying technologies, business concepts, and methods to build up and promote radically new ultra-precision assembly solutions and support infrastructures (EUPASS [4]). This project provides the initial support as well as good contact with industrial partners which provide their best practices on the topics. Furthermore, it provided the necessary background and advancements in modular assembly systems to establish an automatic configuration methodology.

A literature review has been done on the relevant topics which are summarized in **Chapter 2** of this document. It is an interdisciplinary literature review that covers aspects of manufacturing, assembly, agent technology, modularity, configuration and reconfiguration while exploring their interconnectivity. On the basis of this literature review the knowledge gaps are defined and described based on the current state-of-the-art.

The detailed research methodology for this work is presented in **Chapter 3**. It establishes the formal problem definition as well as the research hypotheses.

Chapter 4 reports on the development of methodology formal description models. This provides insight into the creation of the models that enable the capture of MAS

requirements and also the formalization of configuration solutions. The knowledge contained in this chapter enables the creation of the MAS configuration methodology.

Chapter 5 reports on the development of distributed decision making framework, which consists in the description of the agent architecture that establishes the base of the self-configuration methodology. This chapter takes advantage of the model and respective information contained in **Chapter 4** to propose a comprehensive overall approach for agent environment that is able to provide a bottom up configuration solution.

Chapter 6 reports on the development local behaviour models for distributed decision making. This Chapter builds on the definition of **Chapter 5** where the overall system behaviour is defined, but not the local behaviours of the individual agents. The establishment of the local methods that enable emergence of configuration solutions are therefore explored in this chapter.

Chapter 7 reports on the validation scenarios of the proposed self-configuration methodology by presenting and analysing the results obtained. Each core contribution chapter is covered using a different set of validation scenarios to establish an individual validation of each knowledge contribution.

Finally, **Chapter 8** provides an overview of the knowledge contributions described in this work while providing insight into future enhancements to this work. In this chapter the final concluding remarks are also presented.

2 Literature Review

2.1 Introduction

This chapter will present a state-of-the-art review on the topics that serve as foundations for the work developed in this thesis. This review will be broken down into two main aspects, namely the analyses of the state-of-the-art related to modular assembly system and the state-of-the-art of the application of agent technology to the manufacturing domain.

In order to understand the specificities of modular assembly systems it is important to analyse the origin of such systems. To do so, it is important to understand the concept of reconfigurable manufacturing systems (RMS). This topic has been extensively researched for several years providing several approaches to deal with the issue of reconfiguration. This is a field with a high degree of influence from different research disciplines making prime candidate for multidisciplinary solutions.

There are two approaches to achieve RMS, either through highly flexible systems or modular systems (Bi et al. [12]). It is clear that flexibility increases by adding more equipment to the system, however adding equipment that is not in use can be very costly. Modular systems offer the structure to add new equipment as it is needed, providing flexibility as it is needed.

A RMS is designed at the outset for rapid change in structure, as well as in hardware and software components, in order to quickly adjust production capacity and functionality within a product family, in response to sudden changes in market or in regulatory requirements (Koren et al. [2]). This concept requires several enabling characteristics shown in **Figure 2.1**.

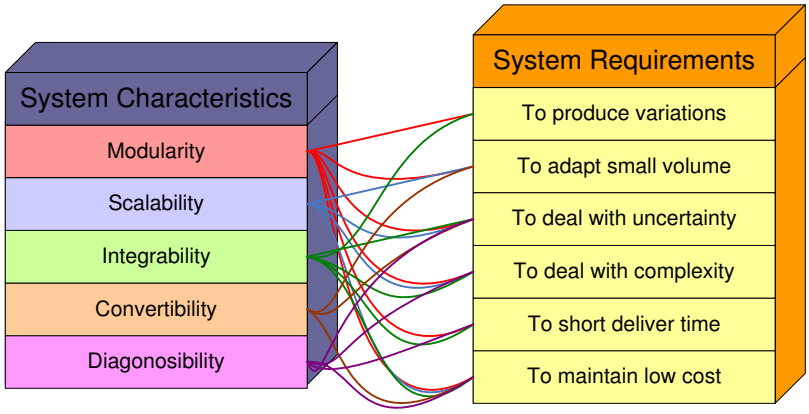


Figure 2.1 - Characteristics to meet system requirements (Based on (Bi et al. [13]))

The reconfiguration efforts have been mostly focused in the control aspects of the assembly systems producing reconfigurable software which is able to change the control of the assembly systems yet falling to address the physical reconfiguration and system enhancement. These approaches are also not related with the systems design and requirements specifications.

The literature has identified the typical difficulties for the development of RMS (Bi et al. [13]):

- The identification and generalization of design requirements, because they are not process oriented
- The automated programming of reconfigurable machines or robot systems
- The systematic methodologies for system reconfigurations
- The standardization and modularization
- The development of a heterogeneous system consisting of different types of reconfigurable machines

The manufacturing tetrahedron shown in **Figure 2.2** identifies the key issues in assembly, which are the bases for comparing reconfigurable assembly systems with conventional assembly systems (Chryssolouris [14]). This provides a visual understanding of the interrelations between these assembly system's requirements and the challenge of increasing flexibility while at least maintaining similar results on the other attributes.

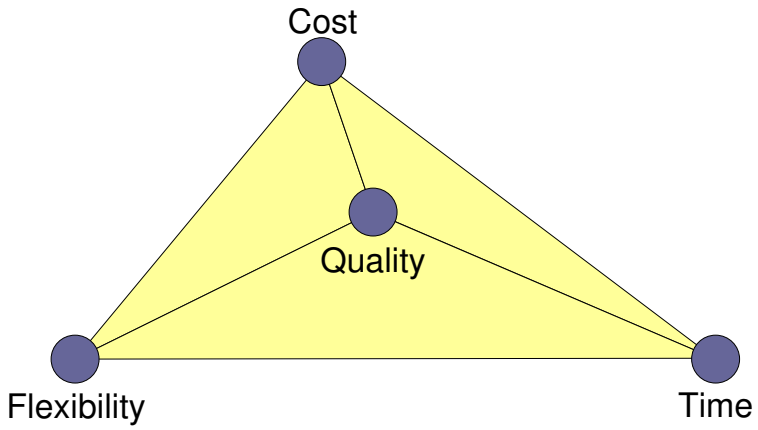


Figure 2.2 - The Manufacturing Tetrahedron (Chryssolouris [14]).

The understanding of these attributes is crucial for the assessment of assembly solutions (Chryssolouris [14]). Attributes such as time and cost are simple to understand, since they have straightforward definitions. Cost is the money spent for the creation and operation of a system. Time is the time required to set up a system and produce products. For quality, on the other hand, it is harder to provide a definition. Normally this attribute is related to the product, but it can be quite complex to define since it can range from product features, which establish quality acceptance criteria, to dismissal of product based on a human assessment of the product quality. The final attribute is not as complex but it is definitely more complex to establish a definition that is generally accepted. Nevertheless the ability of a system to deal with changes despite being quite hard to quantify is generally accepted as a definition.

The analysis of assembly systems is required to understand the importance of the focus on reconfigurable assembly system. The majority of current systems are dedicated production systems for a given product. This means that they are "data rigid" because the product and the process control data are programmed in advance. A shift from this rigid environment into a more dynamic and flexible environment will reduce the cost. However, the most adaptable system possible is achieved through the use of manual processes, since human beings can easily adjust to new tasks and processes. **Figure 2.3** provides an overview of an assembly system under the scope of changes and level of automation.

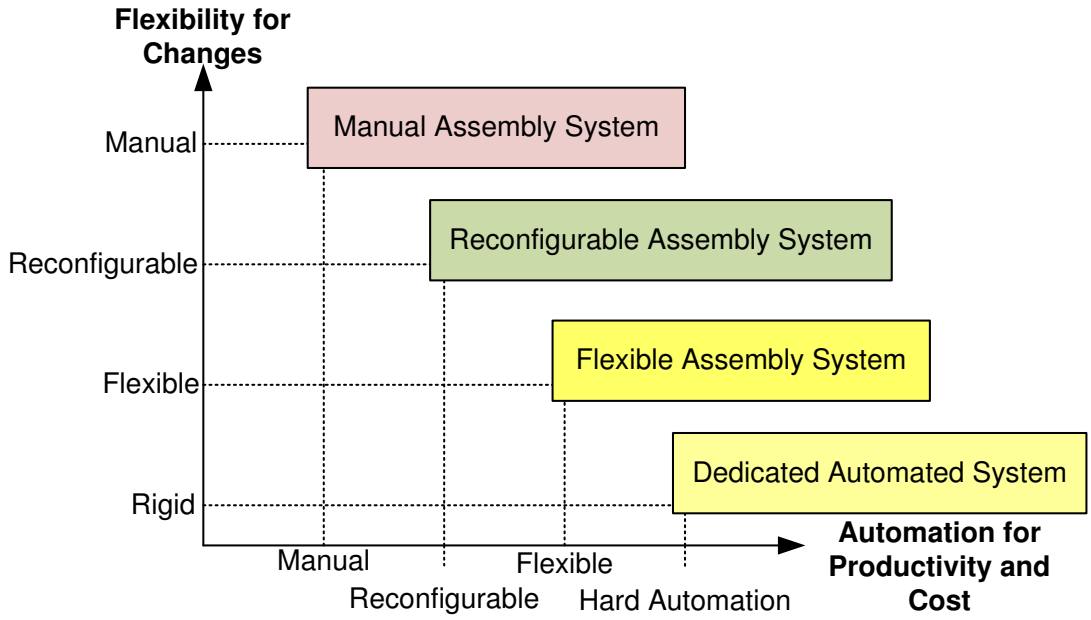


Figure 2.3 - Assembly System Paradigms versus Changes and Automation (base on (Bi et al. [15])).

It is clear that the two options for higher flexibility while looking at cost are reconfigurable assembly system or flexible assembly systems. However, as it was previously stated, flexible systems have the drawback of having redundant equipment built in to the system which has an added cost.

This chapter will firstly focus on the analyses of the state-of-the-art of reconfigurable assembly systems. This will be followed by a review on modular assembly systems which are closely linked to RAS. This review will feature the specific aspects required to achieve an automatic methodology for the configuration of such systems.

The other domain covered in this literature review is agent technology and its characteristics. The review of agent technology will be performed using current solutions within the manufacturing domain to ensure that context specific issues are also covered. This is viewed quite positively since agent technology incorporates a lot of context specific attributes. **Figure** 2.4 shows a mind map some of the different aspects covered in this literature review while highlighting the concepts used for the development of this work.

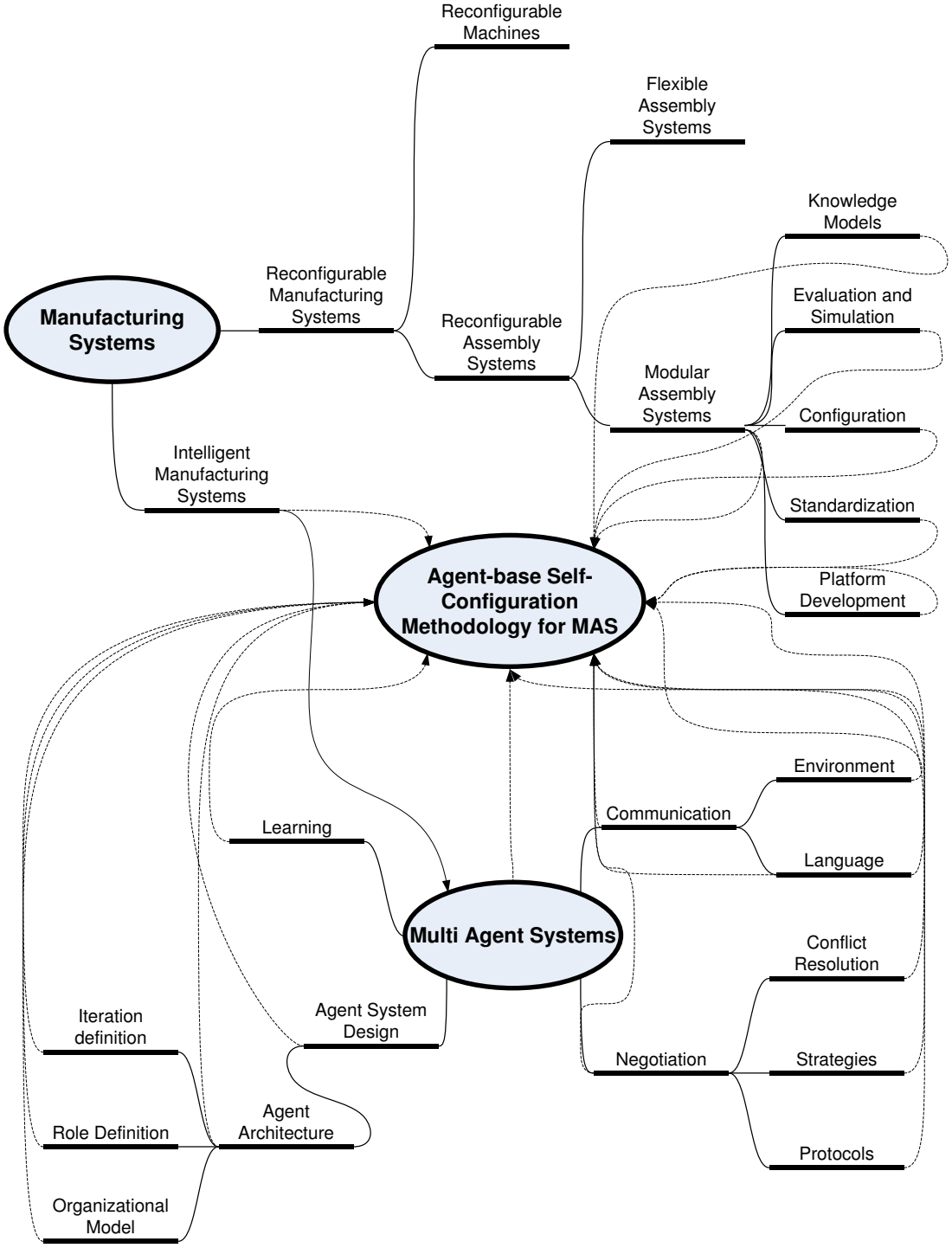


Figure 2.4 - Research domain mind map

2.2 Reconfigurable Assembly System

The understanding of the concept of Reconfigurable Assembly Systems (RAS) requires a definition of assembly. Assembly consists of all assembly processes and equipment required to bring together, configure, align, orient and adjust components and materials to form an end product (Bi et al. [15]). Assembly is a crucial part of the whole manufacturing process, taking up typically 25% to 50% of the total manufacturing cost (Bi et al. [15]). Therefore the processing value is considered to be significantly high compared to other manufacturing processes (Bellgran and Johansson [16]).

The classification of assembly system using the system reconfiguration concept has identified that assembly systems can be dedicated, flexible and reconfigurable (Koren et al. [2]; Bukchin et al. [17]; Onori and Oliveira [18]). Dedicated systems are designed for the production of a specific product with fixed tooling and automation (Mehrabi et al. [19]). Flexible systems are designed for the production of a product family, therefore having fix hardware and fix but adjustable software (Bi et al. [15]). A reconfigurable system is designed for the rapid change of its structure to deal with sudden market changes, which require changes in a product (Bi et al. [15]).

The development of RAS has been supported by several researchers and is becoming more promising due to its capability to deal with changes and uncertainties (Koren et al. [2]; Edmondson and Redford [20]; Yusuf et al. [21]; Michelini et al. [22]; Weber [23]). However, few available systems demonstrate the potential of RAS (Bi et al. [15]). Nevertheless, many companies still use manual system or hybrid systems to deal with an uncertain market (Edmondson and Redford [20]).

The key drivers for the development of RAS have been identified in the literature as the need for reducing cost and improving productivity through automation and the changes and uncertainties in the market (Tichem [24]). The analysis of these drivers provides insight into the issues that impact RAS such as, product variants increase, product volume becomes lower and fluctuates, product lead-time becomes shorter, product proves become more competitive, needs to reduce cost, achievement of high an constant quality, among others (Tichem [24]; Feldmann and Slama [25]; Bodine [26]).

The concept of modularity can be seen as a subset of RAS (Bi et al. [15]). A modular system is capable of generating different configuration through the addition or removal of modules. This means that the system topology can change according to the changes made to the system. Therefore the enhancement of such system can be made nearly infinite (Ulrich [27]).

2.3 Modular Assembly Systems

Modular assembly systems (MAS) are one of the leading approaches to deal with system reconfiguration. The modularity might occur in different levels of the system, from the control to the physical equipment, nevertheless the key aspect for such systems is the need for a high standardization of the module, regardless of its level.

System modularisation provides significant advantages, namely adaptability for product changes, scalability for capacity changes, simplicity due to decoupled tasks, lead-time reduction, maintenance, repair and disposal, among others (Martin and Ishii [28]; Gunnar [29]; Blackenfelt and Stake [30]).

The modularisation of a system involves the analysis of the similarities among system components to establish modules, which should be kept as independent as possible from each other (Bi et al. [15]). Once modules are defined under the context of a modular architecture, a finite set of modules can potentially deal with an almost infinite set of changes (Bi and Zhang [31]).

In MAS there are two types of modules, equipment modules and software modules. By definition modules are interchangeable and are connected by the flow of materials and information (Feldmann and Slama [25]; Heisel and Meitzner [32]).

Recent research has been conducted to enable this approach as a viable industrial solution (EUPASS [4]; IDEAS [33]).

The MAS builds up the concept of “Plug & Produce” (Arai et al. [34]) based on the concept “plug & play” which intends to create highly adaptive systems through a high level of standardization of system’s components and processes.

The standardization of an open architecture for the next generation of distributed control and automation through the IEC 61499 Standard has produced several

advances on the level of control programming (Vyatkin [35]). This provides support tools for reconfigurable control of modular assembly system. The standard provides a good shell structure to describe the control aspects of assembly systems; however a detailed shell including all the relevant control aspects of assembly processes is still missing.

The concept of MAS is highly dependent on modular equipment, and although there has been a lot of research on specific equipment (robots, grippers, fixtures, etc) (Bi et al. [13]) there has been very little standardization and the solutions are very equipment specific.

The EUPASS project brought together several equipment suppliers to produce a standard for modular equipment supported by a modular control structure which opens the door for self configuration systems. The definition of MAS requirements using a common description with the modular equipment is the key enabling factor to achieve self-configurable assembly processes. However the standardized description of the equipment modules supplies all the required information for physical system configuration (EUPASS [4]).

2.3.1 Platform Development

The modular system concept requires the development of a system architecture that can be modified simply by assembling different modules together (Bi et al. [13]). There have been several developments of MAS platforms with different levels of granularity (Alsterman and Onori [9]; Gaugel et al. [10]; Boër et al. [36]; Chen [37]; Giusti et al. [38]), although this technology has not yet been applied widely in industrial environments.

The EUPASS project is one of the attempts to provide a MAS platform with standardized equipment modules, modular processes, open architecture control and standard interfaces that enable the “plug & produce” concept.

2.3.2 Requirements Engineering

Requirements engineering is traditionally defined as “the elicitation and formulation of requirements to produce a specification” (EasterBrook [39]), so it can be inferred

that requirements engineering is the gathering and organization of customer requirements and system specifications describing them in an explicit manner.

The requirements engineering is a broad research topic, therefore this literature review was narrowed to requirement engineering for modular assembly systems. In this context a comprehensive knowledge model was found that claims to target the specific definition of the reconfigurable assembly systems requirements (Hirani [40]). However this does not provide a structured model that caters for the automatic configuration and reconfiguring of MAS, and it lacks the specific definitions for performance and simulation assessment of such systems.

2.3.3 Standardization

The definition of standards is normally associated with maturity of a technology being the key aspect to ensure the interoperability, integration and acceptance of the technology. The existence of standards allows for compatibility of different equipment which is of extreme importance to achieve adaptable systems. It is important to highlight that the standardization effort is not limited to any individual company, but rather a conjoined effort. Furthermore, the majority of the systems combine subsystems of different vendors, which highlight the need for a conjoined standardization effort (Faulkner et al. [41]). Standardization will provide the ability to cater for any equipment supplier by assuring the common communication protocols for all assembly equipments and operations (Grondahl and Onori [42]).

This research topic is quite wide and complex, therefore this literature review will concentrate on the recent advances in modular assembly systems standardization. In modular assembly systems the lack of standardization has been identified as one of the major issues to overcome in order to implement such systems (Bi et al. [13]).

Recent developments in this field have been introduced by research projects bringing together diverse industrial partners towards finding standards for modular assembly systems. The results from this were a standardized assembly processes library that entails the required descriptions to be used in a modular fashion and also the standardization of the physical aspects of the modules through a standardized emplacement and module blueprint description containing the generic characteristics of modules, such as interfaces, capabilities, constrains, etc (EUPASS [4]).

2.3.4 Knowledge Models

Knowledge is a term with no single agreed definition, nonetheless the Oxford English dictionary states that it is: “expertise and skills acquired by a person through experience or education; the theoretical or practical understanding of a subject; what is known in a particular field or in total; facts or information; awareness or familiarity gained by experience of a fact or situation”. This definition helps the understanding of the complexity of this topic since to define a knowledge model one requires the full understanding of the topic.

The definition of knowledge does give focus to information; there is no knowledge without information. The way we structure and deal with information leads us to our own knowledge definition. To capture this there are knowledge representation techniques and knowledge modelling techniques that allow us to formalise knowledge models.

Knowledge-based engineering (KDE) is one possible way to use these models to assist in the decision making processes through established rules based on acquired knowledge (Hirani [40]; Gardan and Gardan [43]).

Within the modular assembly systems domain a very extensive ontology-based knowledge model has been proposed allowing for the description and formalization of system requirements using a standardized language (Lohse [44]).

2.3.5 Evaluation and Simulation

The evaluation and simulation of an assembly system is a quite complex topic due to the specificities of each assembly systems. In the market there are several software tools that support some simulation of assembly systems. With modular assembly systems the simulation issues are simplified due to the concept of modularity, however current systems do not deal with the issues of modularity. Nevertheless, simulation is viewed to have a huge impact in the design stage of systems, where the evaluation of potential solutions is used as a permanent aid to assure the best choices are made (Michelini et al. [22]).

Combining modules and their capabilities is not as simple as adding them together, thus to evaluate and simulate such a system it is required the ability to extrapolate the combined capabilities and behaviours (EUPASS [4]).

Roadmaps on this field have identified that a computer representation of capabilities and behaviour of the system and all its components could potentially be a big game changer that would allow the test of alternative approaches providing the tools to make changes in the early development rather than later when the cost of change is significantly higher (NACFAM [5]). Furthermore, the literature also provides a breakdown of the required efforts to advance in this field (Bi et al. [15]), namely:

- quantifying and evaluating reconfigurable requirements
- analysis and synthesis of system solutions
- modelling and simulation of reconfigurable processes
- modelling of system design and optimization for high RAS
- modelling and simulation of human roles in RAS

2.4 Assembly System Configuration

The assembly system architecture provides the conceptual model, or blueprint, that defines the system structure, behaviour and boundaries of the available types of assembly options of the system components, thus determining the configurations variants of the system (Bi et al. [12]). The configuration design constraints and objectives are derived from task specifications and business strategies.

Configuration design consists of design analysis and design synthesis. The design analysis establishes the mappings from the design variables to the design constraints and from the design variables to the design objectives. The design synthesis finds an optimal solution from all configuration candidates. In reconfigurable systems, the configuration design is repeated once the task requirements are changed (Bi et al. [12]).

Reconfigurable systems can be classified as an uncoupled system, loosely-coupled system, or strongly-coupled system. The establishment of methodologies for configuration design depend on the complexity of the reconfigurable system (Bi et al. [12]).

In uncoupled or loosely-coupled systems the components can be determined individually based on their corresponding requirements. This might require some adjustments to individual components. Configuration design of these systems is comparable to the design of modular products (Bi et al. [12]). Therefore there are many methods that can be applied such as feature-based methods (Perremans [45]), modular-based methods (Tsai and Wang [46]), combinatorial synthesis methods (Levin [47]), entity-based methods (Hong and Hong [48]), and case-based methods (Watson [49]). Research conducted at the University of Michigan provided methodology for reconfigurable machine tools, where the task requirements of a machine tool are represented by matrices of motions, and the screw theory is employed to identify appropriate components (Bi et al. [12]).

In strongly-coupled systems the design variables should be considered together towards validating if the configuration fulfils its requirements. The combination of different variables can fulfil a requirement, thus there is no one-to-one relation between design variable and design requirement (Bi et al. [12]). Early works suggested a sequential design procedure and most of them have considered the portion of system behaviours (Paredis and Khosla [50]; Chen and Burdick [51]). However, the coupling of design variables produced a concurrent consideration of design variables, constraints and objectives towards finding global optimal solutions (Bi and Zhang [31]).

Concurrent design can increase the problem dimension, which increases the computational efforts. To cope with this, two approaches have been proposed: parallel computation (Sims [52]; Parunak [53]) and space reduction approach (Bi [54]).

The configuration design at a system level is usually made through system simulation where an approximate solution is found in a time-consuming iterative process. Mathematical formulation for the system level would be too complex and it is used only in specific sub-problems (Bi et al. [12]). Deterministic models where the system variables are constant have been used in configuration design (Son [55]; Spicer [56]; Tang et al. [57]), however these limit the system adaptability. Stochastic models arise as a solution to this problem since they provide at least one uncertain

variable. Some configuration design methodologies have used stochastic models in order to deal with the configuration problems (Zhao et al. [58]; Ohiro et al. [59]).

Although a lot of research on the topic of configuration methods has been done, there is not any systematic configuration design methodology. Most of the research efforts in this field have been conducted on the machine level, while the systems have been designed intuitively (Bi et al. [12]).

2.5 Multi-Agent Systems for Intelligent Manufacturing

Agent technology is widely recognized as a promising paradigm for the next generation of manufacturing system (Shen et al. [1]). It has already been applied in several manufacturing domains such as: concurrent engineering, collaborative engineering design, manufacturing enterprise integration, supply chain management, manufacturing planning, scheduling and control, material handling, etc (Shen et al. [1]; EUPASS [4]; Onori and Oliveira [18]; Oliveira [60]; Maturana et al. [61]). This underlines the importance of this technology and the relevance of its underlying concepts to perceive its applications.

Several definitions can be found in literature for “agent” yet there is no global accepted one. Computer science defines generically an “agent” as a software abstraction similar to object oriented programming terms such as methods, functions and objects. The concept of an “agent” is referred as a convenient and powerful way to describe a complex software entity that is capable of acting with a certain degree of autonomy in order to accomplish tasks. But unlike objects, which are defined in terms of methods and attributes, an agent is defined in terms of its behaviour (Nwana [62]).

Agents are classified for their characteristics such as mobility which determines if the agent is static or mobile. This classification can depend on a combination of characteristics as shown in **Figure 2.5.**

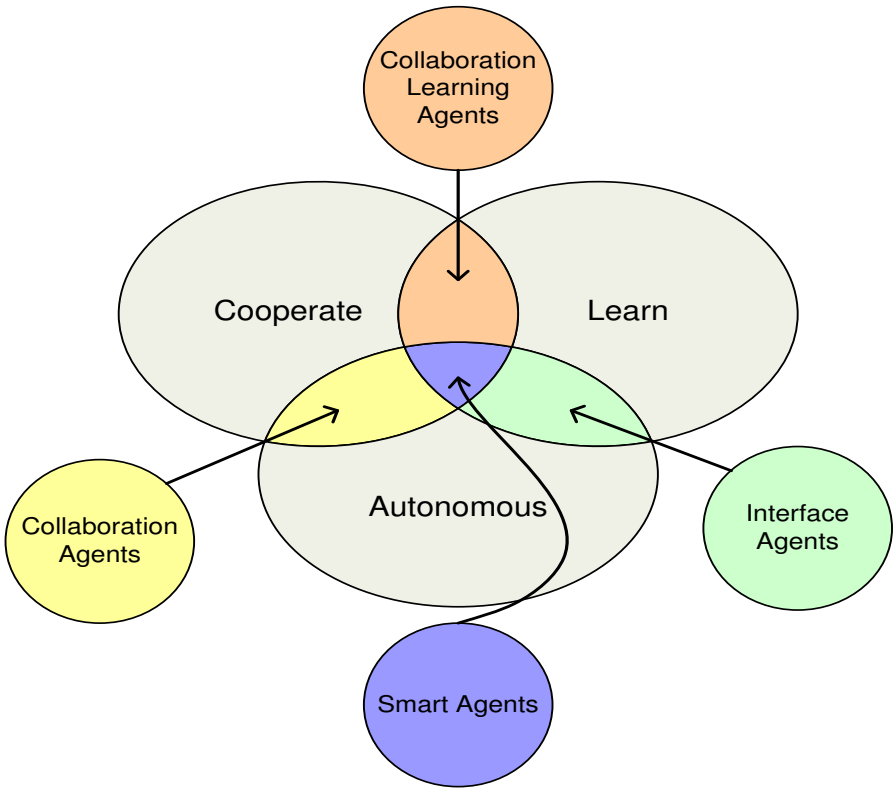


Figure 2.5 - A partial view of agent topology (Based on (Nwana [62])).

A minimal common definition established by (Ferber [63]) states that an agent is a physical or virtual entity:

- Which is capable of acting in an environment
- Which is able to communicate directly with other agents
- Which is driven by a set of tendencies (in the form of individual objectives or of a satisfaction/survival function which it tries to optimise)
- Which possesses resources of its own
- Which is capable of perceiving its environment (but to a limited extent)
- Which has only partial representation of this environment (and perhaps none at all)
- Which possesses skills and can offer services
- Which may be able to reproduce itself
- Whose behaviour tend towards satisfying its objectives, taking account of the resources and skills available to it and depending on its perception, its representation and the communications it receives

Agents are able to perform actions going beyond reasoning abilities which makes them an enhancement of conventional artificial intelligence (AI). This is a key characteristic of agents that in conjunction with their ability to communicate enables multi-agent systems (Jennings and Wooldridge [11]; Ferber [63]).

The fact that agents can be autonomous enables the definition of different tendencies adjusted to whom they represent. The agent follows these tendencies within its environment producing complex results out of the collaboration with agents with different tendencies. This result is obtained without defining very complex models of interaction (bottom-up approach).

In sum, an agent can be described as some sort of "living" entity which has a certain behaviour that can be recapitulated as communicating, acting and even reproducing, aiming at satisfying its needs and obtaining its objectives and using the available elements (perceptions, representations, actions, communications and resources).

Multi-agent systems can be roughly defined as environments where different agents interact with each other. The complexity of multi-agent systems can vary based on the complexity of the agent's behaviour. Further in this chapter it is discussed how the agent's behaviour affects the complexity of the multi-agent system. The agent organizational structures in multi agent systems also play a key role on the complexity of the system (Ferber [63]).

In manufacturing systems, agent technology is seen as the natural way to address the problems presented by traditional approaches and that limit the expandability and reconfigurability of such systems (Shen et al. [64]). Furthermore, agent technology has recently been considered as a paradigm for developing distributed industrial systems (Jennings and Wooldridge [11]; Jennings et al. [65]). It has been highlighted as a promising concept for the next generation of manufacturing systems (Shen et al. [1]; Shen and Norrie [66]). Moreover, agent technology has been widely applied within the field providing solutions for manufacturing enterprise integration, enterprise collaboration (including supply chain management and virtual enterprises), manufacturing process planning and scheduling, shop floor control, and to holonic manufacturing as an implementation methodology (Shen et al. [1]; Jennings and Wooldridge [11]; Parunak [53]; Wooldridge and Jennings [67]; Deen [68]).

The use of agent technology in the intelligent manufacturing context has been implemented in several ways, providing distinct approaches to the use of agent technology under this context (Shen et al. [64]). Agent technology has been used as a wrapper for manufacturing activities in a distributed environment using functional decomposition approach. Examples of this include product design, engineering analysis, process planning, production scheduling, simulation and execution (Azevedo et al. [69]; Barry et al. [70]; Fox et al. [71]; McEleney et al. [72]; Peng et al. [73]; Sadeh et al. [74]; Shen et al. [75]; Yen and Wu [76]). These solutions provide a significant improvement of the integration of heterogeneous software and hardware systems (Shen et al. [64]).

The implementation of agent technology in the intelligent manufacturing domain has also used a representation approach, which consists in the representation of physical resources (e.g., machines, robot, tools, fixtures, etc.), as well as parts, operations and processes (Butler and Ohtsubo [77]; McDonnell et al. [78]; Parunak et al. [79]; Shen and Norrie [80]; Lu and Yih [81]; Usher [82]; Wang et al. [83]). The concept of representation under the agent technology domain also opened the possibility for agent deployments as representations of negotiation partners to facilitate enterprise collaboration (Sadeh et al. [74]; Bremer and Molina [84]; Nigro et al. [85]; Hao et al. [86]). Furthermore, this enabled research on agent based architectures for manufacturing systems design (Shen et al. [75]; Parunak et al. [87]).

The literature on agent technology provides extensive sources of information for agent models, negotiation models and agent environments, among others that range from shop floor control (Oliveira [60]) to virtual enterprises (Camarinha-Matos [88]), however these models are mostly application specific. Nevertheless the key advantage of agent technology is its adaptability, which enables it to be applied to different levels guaranteeing an overall integration. Therefore it is useful to analyse the solutions given by the literature for establishing best practices for the use of agent technology.

The MetaMorph II is an agent based architecture for distributed intelligent design and manufacturing with the objective of integrating the manufacturing activities (e.g., design, planning, scheduling, simulation, execution, etc) with the activities of the suppliers, customers and partners within a distributed system (Shen et al. [75]).

This project builds on the MetaMorph I which addressed system adaptation and extended-enterprise issues at four fundamental levels: virtual enterprise, distributed intelligent systems, concurrent engineering, and agent architectures (Maturana et al. [61]). The projects provided an overall architecture for collaboration which included some reconfiguration methods namely on the control side. The projects also defined agent organizational and collaboration models which followed different architectures (Federation, hybrid) (Shen et al. [75]). The projects provide a good global approach to the problem but do not really address the issues of system configuration and reconfiguration, targeting more the system adaptability.

The AARIA (Autonomous Agents for Rock Island Arsenal) agent architecture also provided an agent-based system design presenting another agent organizational and collaboration model which was more requirements driven (Parunak et al. [87]). Although this project describes an interesting requirements' driven approach, it is very case specific, providing a limited scope of requirements. Nonetheless, this project provides valuable guidelines for the developing agent solutions for a non case specific system.

Agent-based approaches are mainly used to provide agility and reconfigurability of manufacturing systems. However, optimization is also one of the most important objectives of such approaches. This approach to optimization is quite different from the mathematical approaches that target global optimization through mathematical formulation of industrial problems which for complex systems can be quite difficult. On the other hand, agent approaches attempt to achieve optimization through efficient coordination mechanisms (Shen et al. [64]).

2.5.1 Agent Organization

The organization of agents is a crucial aspect in any multi agent environment. It provides the basic rules for the interactions between agents. There are three distinct approaches for agent organization in manufacturing systems, the hierarchical approach, the federation approach and the autonomous approach (Shen et al. [64]).

The hierarchical approach takes advantage of the existing structure of manufacturing environments, where there is a workstation that contains equipment units that execute certain operations. Examples for the use of this approach are described in the

literature (Butler and Ohtsubo [77]; Leeuwen and Norrie [89]; Bussmann [90]; Burke and Prosser [91]; Fischer [92]), although this approach is criticized due to its centralized appearance.

The federation approach has some variations but in essence consists on the establishment of clusters where a special agent is created to operate on behalf of the group. In the literature, the special agent is identified as a facilitator, broker and mediator. The facilitator consists of an agent that assumes all communication between agents. It provides the means for communication between local and remote agents usually by providing services such as routing outgoing messages to the right destination and translating incoming messages (McGuire et al. [93]; Petrie et al. [94]).

The brokers are quite similar to the facilitators, since they execute the same things but have extra functionalities of monitoring and notification (Oliveira [60]). The functional difference between the two is that the facilitator is responsible for a given agent cluster, while in the broker approach any agent may contact a broker for the execution of a given service (Peng et al. [73]).

The mediator is an agent that assumes the role of system coordinator by promoting cooperation among other agents and learning from their behaviour (Maturana et al. [61]; Shen et al. [75]; Ouelhadj et al. [95]).

The use of federation as the core concept of the agent architecture provides the means to coordinate multiple agent activities via facilitation as a way to reduce overheads, ensuring stability and providing system scalability (Shen et al. [64]).

The autonomous agent approach has a lot of different definitions, but in essence it is a multi agent environment where agents are individuals that are not controlled or managed by other agents or human operators. All agents are able to interact with each other without any preconceived rules. To have such a system, the agents need to posses knowledge about the environment and other agents that are contained in the environment and also a set of goals that drive their operations in the environment (Azevedo et al. [69]; Shen et al. [75]; Shen and Barthès [96]; Babayan and He [97]).

2.5.2 Agent Negotiation

Multi agent systems are populated with agents with different behaviours and objectives. So what happens when agents have both cooperative and conflicting interests at the same time? In such situations the agents have the problem of defining how to cooperate to obtain the associated benefits, thus emphasising the importance of negotiation in multi-agent systems which enables the agent to resolve conflict situations through reasoning and communication (Kraus [98]).

The topic of negotiation is by itself a complex research topic which has been widely investigated in several fields, and broadly speaking one can define negotiation as an interaction of influences. A more complete definition of an agent based scenario was given by Lesser: "Negotiation, the process of arriving at a state that is mutually agreeable to a set of agents, is intimately related to coordination. The negotiation process can be used as part of a multi-agent coordination algorithm that implements, for instance, a contracting mechanism for getting one agent to commit to solving a sub problem for another agent"(Lesser [99]).

The establishment of the negotiation model can be broken down into four components:

- The negotiation protocol
- The negotiation strategies
- The information state of agents
- The negotiation equilibrium

Negotiation between agents uses a premise that they can communicate and understand each other. This is achieved by establishing public rules that allow agents to achieve agreements, which are commonly designated as protocols. The protocols define the kind of interaction that can be made, as well as the allowed offers and counter-offers sequences. The protocols do not deal with the mechanisms of communication, simply address its content, thus protocols are very specific to the targeted domain (Rosenschein and Zlotkin [100]). Protocols establish the restrictions imposed to the agent's interactions, these restrictions have a direct impact on the reduction of the required communications to achieve a beneficial agreement (Kraus [98]).

A strategy can be defined as the approach that the agent should take to maximise its success, thus it is the definition of the agent's next move in an interaction. The interactions are constrained by the protocols; nevertheless the deals proposed by the agent are based on its strategy. A simple example of this, is an agent with the objective to maximise quality that will negotiate with an agent that wants to maximise cost, both have their own strategies but require a common protocol for negotiating. Therefore, there are usually many strategies compatible with a particular protocol, thus different strategies can be present that achieve different outcomes. The definition of strategies is not obligatory to solve conflicts since it can be avoided by the existence of a centralized algorithm that deals with all possible conflicts, although this is not possible in systems with no agreed hierarchal centralized structure and dynamic systems (Kraus [98]).

The information state of an agent describes the information it has about the negotiation. In a nutshell, agents can have complete information when they are aware of all relevant information about the rules of the game and other agent's preferences, or they have incomplete information where information may be lacking, thus agents may have private information about their own situation that is unavailable to other agents (Kraus [98]). This obviously has a big impact on the definition of agent's strategies.

The negotiation equilibrium is the point where all agents and respective strategies have no motivation to change the status quo. This is quite an important characteristic in multi agent systems since this is the point that negotiations end until the equilibrium is disturbed (Fatima et al. [101]).

The negotiation has the principle that it requires a topic to discuss, thus the establishment of the topic or topics to be discussed is the first step towards having a successful negotiation process.

Within the domain of manufacturing, negotiation has been used to enable decision making capabilities towards achieving the systems design objectives (Shen et al. [64]). The main concern of negotiation in the literature is the resulting behaviour of the multi agent environment in terms of stability. The stability of the system often in literature is associated with the term coordination, which comes in play for complex systems. In a simple system, the stability of the system given a set of negotiation

strategies can be foreseen, however in a complex system this task is quite complicated (Shen et al. [64]). The organizational approach here takes a central role, since the easiest way to guarantee that the multi agent environment does not degenerate is through the creation of a coordination agent (Shen et al. [64]).

However the creation of such an agent centralizes the decision making of the system, since this agent would gather information, create plans, and assign tasks in order to ensure the normal operation of the system. This is in fact the traditional centralized manufacturing system approach that establishes controllers that are hierarchically above other controllers which they regulate (Shen et al. [64]). The problem is that the reconfiguration of such systems is quite complex and involves a lot of effort. This goes against the current need for more reconfigurable system due to market changes, which is the main factor of the current interest in multi agent system. Furthermore, the use of a central controller for large groups of agents raises an issue of system scalability. The larger the group of agents under the controller the more complicated it is for the controller to be informed of all things happening in the system. In fact, the controller under these conditions becomes a communication bottleneck which brings problems to the system performance.

The issue of having a central coordinator does not imply that all coordination involves a completely centralised approach. Actually, there are several different coordination mechanisms in the literature, namely mutual adjustment, direct supervision, coordination by standardization, mediated coordination and coordination by reactive behaviour (Shen and Norrie [80]). These can be used depending on the agent organizational approach.

2.5.2.1. Negotiation Protocols

As was said before, the negotiation protocols are domain specific, nevertheless there are guidelines towards defining negotiation protocols using formal languages that enable communication (Finin et al. [102]; FIPA [103]; FIPA [104]).

The establishment of negotiation protocols requires a clear definition of the parties involved in this process, thus the agent architecture needs to be defined before defining the negotiation protocols. The negotiation protocols need to be defined taking into account the knowledge domain which they are addressing, this allows for

a better definition of the negotiation rules, which in turn reduces the negotiation effort.

Within manufacturing systems some guidelines for the design of negotiation protocols have been defined (Krothapalli [105]). However, due to the close relation between the agent architecture and protocol definition, the proposed negotiation protocols are very specific and do not cover modular assembly systems, nonetheless these provide a good support for the definition of new protocols. Furthermore, for a better understanding of the definition of negotiation protocols it is important to analyse other solution within the manufacturing domain.

The usual negotiation protocols used within the manufacturing domain are mostly Contract Net protocols (Smith [106]), or variants of this (Shen et al. [64]). Examples of this can be found in literature, however they tend to be problem specific solutions (Butler and Ohtsubo [77]; Shen and Norrie [80]; Ouelhadj et al. [95]; Duffie and Piper [107]; Parunak [108]; Ow and Smith [109]; Shaw [110]; Saad et al. [111]). Despite the general use of this protocol, other market-based approaches are becoming more popular. Market-based protocols are building using the principle of auctions, which make them quite simple to define and use. The use of this type of protocol in the manufacturing domain is mainly on scheduling systems (Baker [112]; Lin and Solberg [113]).

2.5.2.2. Negotiation Strategies

In agent technology the negotiation strategy is the approach that agents take to find a compromise that suits all parties trying to maximise their objectives, thus the strategy defines what the agent is willing to compromise and in return of what. In multi-agent systems the negotiation strategies are of extreme importance since they should be able to cope with a diverse agent environment (Kraus [98]).

In the literature there are classifications of negotiation strategies (Shen et al. [1]) that have been used within the manufacturing domain, namely:

- Game theory based negotiation
- Contract based negotiation
- Market based negotiation

- Plan based negotiation
- AI based negotiation
- Other approaches

The literature does not identify the best approach for the design of negotiation strategy. Analyses of different strategies in specific domains have shown that no strategy dominates over another and that combining strategies constitutes a good approach (Matos et al. [114]).

The game theory based negotiation has been indicated to produce optimal strategies and predict outcomes. However, within complex domains most strategies are designed resorting to intuition and experience of the designer. This happens because in complex domains there are no clear optimal strategies, thus the definition of strategies uses heuristic approaches (Rahwan et al. [115]). Nevertheless, there are some examples in the literature where the similarity between the characteristics of the problem and game theory have been found and explored, namely in the context of independent schedule decisions (Guan et al. [116]).

The choice of negotiation strategy is highly dependent on the analysis of the problem. In fact, it is not possible to create solution without some adjustments to the strategies to cater for the specificities of the problem domains (Shen et al. [64]). The definition of an agent negotiation strategy is also highly dependent of the choice of organization approach and the definition of the agent roles in the wider context of an agent architecture (Henderson-Sellers [117]), since these have a high impact on the definition of agent behaviours which in turn are implemented using a negotiation strategy.

2.5.2.3. Conflict Resolution

Conflict resolution, in very simple terms, is the attempt to resolve a conflict or a dispute. The first step in conflict resolution is the identification of the conflict situation so that the negotiation mechanisms can step in (Kraus [98]).

A conflict is a state of discord caused by the actual or perceived opposition of needs, values and interests. A conflict can be internal (within oneself) or external (between two or more individuals). Extrapolating this definition to the agent world, a conflict

occurs once there is an inability to achieve an objective or a belief, either because of oneself or a combination of agents (Fatima et al. [101]).

The use of some sort of hierarchical structure in the development of solutions using agent technology under the manufacturing context, provide very little information for distributed conflict resolution. In fact, the possibility of the existence of conflicts is minimized on the design of the agent environment as way to ensure the rapid response and stability of the solutions (Shen et al. [64]).

2.5.3 Agent Architecture

An agent's architecture is roughly its internal design, covering aspects from the knowledge it possesses to their reasoning abilities and thus the way they behave. On a multi agent environment this can become a quite complex definition depending on the taken approach. The architecture of a multi agent system generically follows a organizational approach of hierarchical (low flexibility), autonomous (low scalability) or hybrid architectures. Regardless of this, it has to define the types of agents present, their organizational clusters, their roles, their goals, their tasks and interactions (related to the negotiation protocol definition). The definition of an agent architecture and subsequent agent system requires a structured approach as for any problem solving activity (Henderson-Sellers [117]).

The literature provides several methodologies for the definition of agent systems (Bernon et al. [118]; Cossentino [119]; Garijo et al. [120]; Iglesias and Mercedes [121]; Padgham and Winikoff [122]; Zambonelli et al. [123]). Each of these has its own unique perspective and approach on the definition of an agent system, and there is not a clear choice in methodology (Henderson-Sellers [117]). The literature suggests that problem analysis and the definition of the agent system requirements is the key aspects in the choice of the right methodology for the definition of an agent system. In fact, a common denominator of all methodologies is the need for the definition of clear requirements for the agent environment, and not just the requirements for the solution outputs (Henderson-Sellers [117]). The Gaia methodology provides a good generic approach that is broken down into four stages, the requirements specification stage, the analysis of the requirements, the design stage and the implementation stage. The requirements specification consists on the

formalisation of the problem, the definition of the objectives and assumptions of the environment. The analysis of the requirements stage consists on analysing the problem and identifying the necessary agent roles and interactions that establishes the overall agent architecture. The design stage consists in detailing the individual agent's behaviours and strategies (Zambonelli et al. [121]).

The assessment of existing systems which have used a methodology is seen as an important factor for the decision on the suitability of a methodology for a similar problem. However, the use of these structure methodologies for the definition of agent architectures in manufacturing has not been widely used. Additionally a review of agent architectures in the domain of information technology provides evidence that the majority of the systems use these methodologies as guidelines for the system design, sometimes merging different concepts as it is suitable to solve a given problem (Sugumaran [124]).

Within the manufacturing domain there have been developed some agent architectures which provide useful solutions and hints for future development in the manufacturing domains, although these do not follow the concepts from the structured methodologies they are still quite important for the understanding of the problems in the domain that are solved using agent technologies (Shen et al. [1]; Inohira et al. [125]; Ryu et al. [126]).

2.5.4 Communication

Agent technology uses the assumption that agents communicate with each other in an understandable manner. The communication between agents has been subject of investigation since the creation of agent technology which provided two leading agent communication languages: KQMP (Finin et al. [102]) and FIPA ACL (FIPA [103]; FIPA [104]). Both these languages provide basic specifications and structures for communication, knowledge and ontology guided communication (Finin et al. [102]; FIPA [127]).

FIPA is the most widely used in the literature, and it provides an open and quite flexible way of defining the language for an agent environment. Its wide use and its openness provide great means to ensure interoperability between existing system and newly developed systems.

2.6 Knowledge Gaps

The current state-of-the-art provided an overview on the current advances in MAS. The focus has been predominantly on the development of equipment module definitions which provide the information required for creating automatic solutions. Moreover, it revealed that **the current MAS have insufficient automatisms and support tools to be more reactive and flexible producing solutions**. In fact, one of the gaps in current MAS is the **lack of a formal configuration and reconfiguration methodology to define a MAS based on system requirements and available equipment**; current practices for such systems rely on human experience and judgment to provide solutions for configuration and reconfiguration. Furthermore there is a **lack of bottom-up approaches for configuration problems**, the majority of the configuration methods reported use top-down approaches, which provide very rigid solutions.

The literature identifies agent technology as one of the best ways to solve distributed problems. Additionally the concept of modularity has characteristics that enable distributed approaches. Therefore it is a good approach to establish a self-configuration methodology for MAS.

Currently **the existing agent architecture in the field of manufacturing does not fully cover the issues of MAS self-configuration**. This presents a clear gap since there are **no appropriate agent architectures to support configuration of modular assembly systems**.

The development of the methodology for self-configurability of MAS through agent technology provides a simple structure to deal with a complex problem, yet **there is currently little formalisation of the interaction and negotiation protocols applied during the configuration of a technical system**. Particularly for automated distributed systems.

Currently MAS optimization is based on human expertise, partly because of t**he absence of the behaviour and capability models, but also due to the absence of an appropriate structure to simulate and evaluate the systems.** The literature provides the main optimization aspects for MAS (Flexibility, Quality, Cost, Time), yet currently the existing models lack the appropriate structure to support decision

making driven concurrently by these aspects. Furthermore there is an **insufficient availability of early MAS configuration assessment methods that** would have a huge impact on the MAS configuration decision making process.

The **scarce availability of suitable decision making support tools for the system design** has also been highlighted in the literature. In fact, this has been identified as one of the potential game changes in the assembly domain. The gaps in this are quantifying and evaluating reconfigurable requirements, analysis and synthesis of system solutions, modelling and simulation of reconfigurable processes, modelling of system design and optimization for high RAS, modelling and simulation of human roles in RAS.

2.7 Chapter Summary

The literature review has provided the general background of concepts used in this thesis. It includes a general review of RMS and RAS, and a detailed review of modular assembly system and the concepts it involves. Furthermore, this review includes a review on assembly system configuration process which is significantly important for the work developed in this thesis.

The second part of this state-of-the-art review focused on agent technology under the manufacturing domain. This covered aspects such as agent architectures design, agent organization models, agent negotiation protocols, agent negotiation strategies, among others.

3 Research Approach

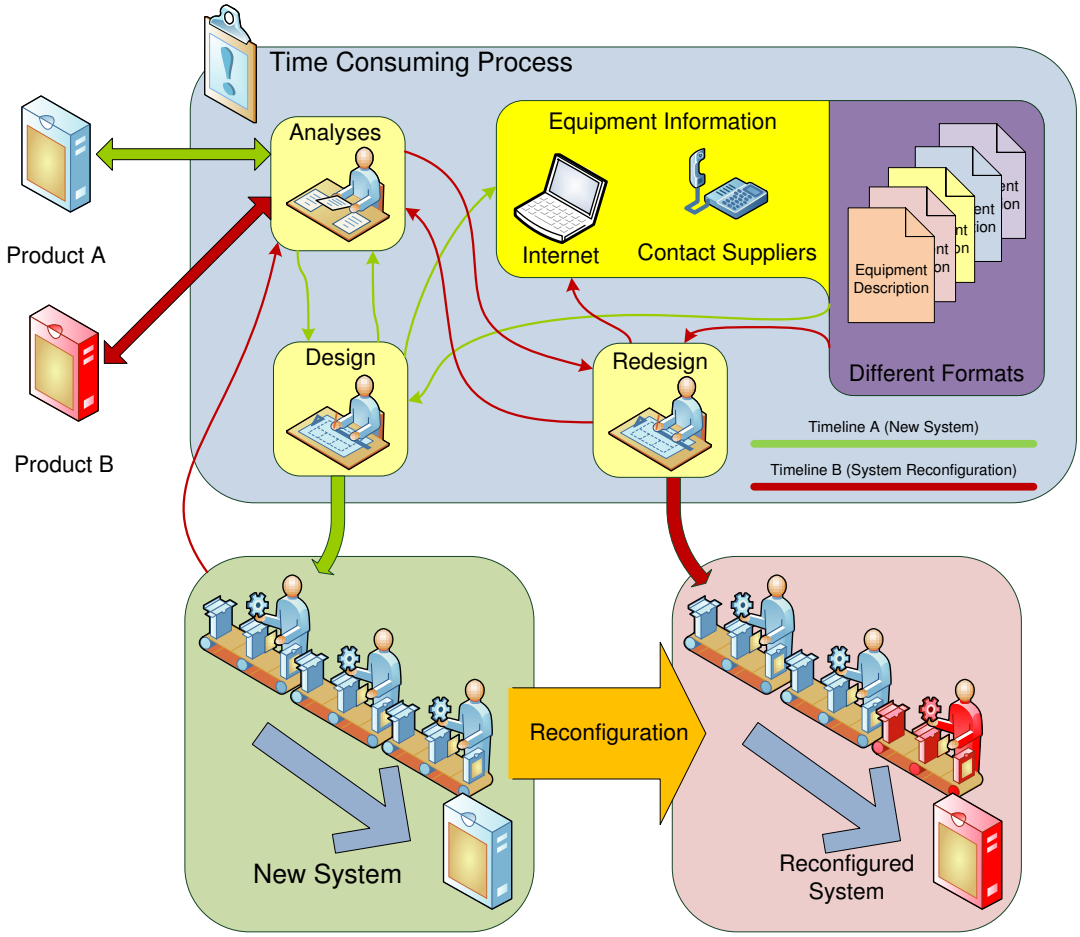


Figure 3.1 - Current MAS Configuration and Reconfiguration Process

3.1 Introduction

This work was motivated by the current trend towards Modular Assembly Systems (MAS) and the analysis of the current state of the art of these systems. The MAS paradigm has a series of objectives which were covered in the previous chapter. However, the development of MAS is still in its early stages and it raises a clear problem of scalability in the future. The work presented in this thesis intends to provide answers to allow the automatic configuration and reconfiguration of MAS in a constantly evolving domain.

In order to understand the aims and challenges of this work it is important to analyse the current state-of-the-art of the configuration and reconfiguration of MAS. The configuration and reconfiguration of MAS is currently a very manual process which requires a lot of analysis from the system integrator. This alone would not be a problem, however, with a growing solution space, as a result of many different modules being available, the system integrator would struggle to reach good solutions. **Figure 3.1** provides a simplified overview of the configuration and reconfiguration process. In this conceptual view it is clear that the analysis effort lies mostly with the system integrator. By increasing the number of equipment modules by any factor produces a big impact on the analysis time, if the system integrator considers all the possible configuration and reconfiguration solutions. Or it makes it much less likely for the system integrator to intuitively choose optimal solutions.

The vision of this work is to provide the means to support this configuration and reconfiguration process through the creation of a methodology that is able to analyse and provide a set of viable solutions to the system integrator. This will reduce the time required to analyse while increasing the speed and quality of configuration and reconfiguration solutions(Onori et al. [7]; Lohse [44]).

The aim of this work is to develop a configuration and reconfiguration methodology that enables the automatic configuration of MAS given a set of requirements. To achieve this aim it would be necessary to create a complete implementation of the whole theory and carry out substantial validation work across the whole domain. Consequently, the work involved to create a complete domain theory goes far beyond the scope of this research. Therefore, the proposed self-configuration methodology is

not intended to provide a complete solution but rather to build a suitable foundation that can evolve with the domain changes while also providing the basis for further enhancements of the approach.

In this chapter, the details of the research methodology followed in this work are presented.

3.2 Problem Definition

The problem definition for this work was design with a foundation on three pillars, the literature review presented in **Chapter 2**, the industrial input provided by the involvement in collaborative research project and finally the current state-of-the-art of the MAS domain.

The question of component reusability, rapid configuration and reconfiguration to enable "Configure to order" of assembly systems has become increasingly more important due to ever decreasing product life-cycles and rising process complexity. General purpose assembly machines, equivalent to CNC machine tools, are only available in specialist domains such as printed circuit board assembly where the components are highly standardised. The assembly of most other products demand custom made systems which address the specific requirements for these products. Today, these are mostly "Engineered to Order" making them cost and time intensive to design or reengineer. Increased modularisation of assembly equipment, rapid integration and design tools are considered fundamental for the move towards cost and time effective configuration and re-configuration of complex assembly systems (Koren et al. [2]; Kratochvíl and Carson [6]; Onori et al. [7]).

Currently, the design of assembly systems is a human driven approach based on the expertise of system integrators. Although this process provides valid system configurations, it can be quite time consuming, often considers only a fraction of the possible solution space, and does seldom provide repeatable and transparent solutions. The MAS paradigm with its focus on clear functional decoupling of equipment module functionalities and standardised interfaces for interchange ability has opened the scope for automatic configuration methods. It becomes possible to clearly formalise the functional capabilities and connectivity constraints of the

available modules hence allowing the mapping of required against available capabilities. The design of MAS is therefore essentially a conjoint equipment and process configuration problem at several levels of granularity with equipment modules and their functional capabilities (skills) as the elementary building blocks (EUPASS [4]).

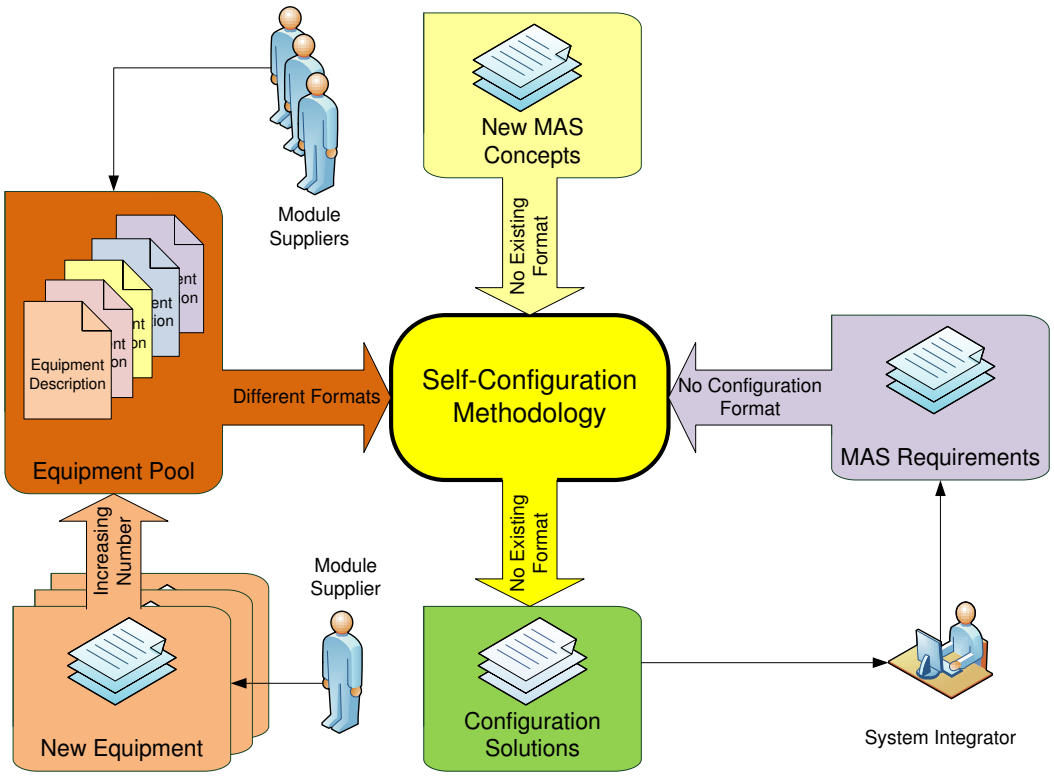


Figure 3.2- Problem Definition Overview

The MAS configuration problem can be defined as illustrated in **Figure 3.2**. A set of assembly process, system and business requirements needs to be translated into possible assembly system solutions using a given set of equipment modules. A set of methods and tools will be required to determine both the technical and logical completeness of different configurations and establish their respective performance characteristics. One of the key challenges is the concurrent solution configuration for both the process logic based on the available skills of equipment modules and the physical hardware required to execute the process logic. Another important aspect is the possibility of new concepts and paradigm shifts in the domain, as the domain is expected to evolve in the future (Onori and Oliveira [18]). This openness to new concepts allows for the solution to remain valid in a domain which has not reached the full maturity.

The analysis over the actors involved in this process is another crucial aspect of the problem. There are two major types of actors in this process, the equipment module suppliers and the system integrators. The equipment module suppliers provide the construction blocks used by the methodology to establish solutions, while the system integrator provides the requirements for these solutions. Once configuration solutions have been reached, they are passed on to the system integrator consideration and selection. It is important to note that the methodology is a support tool for the decision of a system configuration by the system integrator.

The final aspect of the problem that was taken into account is the diversity of standard formats for descriptions. The literature and industrial practices shown a quite disperse environment where the lack of standard definitions and terminologies is overcome by the users. This is obviously one of the biggest challenges to achieve a self configuration methodology, since without clear, transparent, structure and meaningful information it is not possible to establish such methodology.

3.2.1 Requirements for the Self-Configuration Methodology of Modular Assembly System

The definition of the boundary conditions of the problem domain is a crucial factor in enabling the operation of the methodology. The establishment of the set of conditions not only provides the base line of rules for which the solution is valid, but also provides good insight into the problem resolution. Thus, the first requirement for the self-configuration methodology is a clear description of the boundary conditions, namely what are the inputs and outputs of the methodology. This implies the creation of clear, transparent and well structure data models. The models will have to rely on a common terminology to enable the mapping between the information coming from different sources.

The self-configuration methodology will have to be able to deal with the models that are used for inputs and also produce outputs in the agreed formats. The methodology will be required to combine the information from the equipment modules to fulfil the set of established requirements.

A clear methodology for designing the configuration methodology should be followed to ensure a systematic approach for solving the problem.

3.2.2 Definition of Research Objectives

The aim of this work is to provide a self-configuration methodology for MAS. **The main research objective is to establish the suitable approach to solve the self-configuration problem**. The idea is to use existing technologies and methods that can contribute for the establishment of self-configuration methodology. This research objective is covered in **Chapter 2** through the literature review together with the identification of the specific knowledge gaps that prevent the existence of a self-configuration methodology.

The next two research objectives are closely related, **one is the architecture design and respective models that will enable the existence of a dynamic environment that will produce configuration solutions**. **The other is the definitions of local behaviour models for distributed decision making which will drive the methodology and provide the necessary solutions for MAS.**

However, to achieve a self-configuration methodology, one requires the establishment of the relevant models and structures for describing the information required for the operation of the methodology. **Therefore, the establishment of formal description models that enable the self-configuration methodology is also a clear research objective**.

In summary, the main research objectives are:

- Development of methodology formal description models
- Development of distributed decision making framework
- Development local behaviour models for distributed decision making

3.2.3 Definition of the Research Hypothesis

The definition of the research hypothesis is the core of this chapter and sets the scene for this thesis. In a constantly evolving domain it is expected that in the near future the available numbers of equipment modules will increase quite significantly, creating a scalability problem for the configuration of MAS using the current human driven method. Therefore the need for support tools is a clear demand. However, the definition of support tools requires clear models that provide the necessary description of the domain for the tools to be able to interpret and process. Thus the first aspect of the research hypothesis is **if a structured and transparent model can be defined which formalises the physical and assembly process constraints of**

equipment model and a model that enables the definition of MAS requirements using the same concepts, it will be possible to establish automatic configuration methods.

In the scenario that this first statement of the hypothesis, where the necessary description models exist for the purpose of self-configuration methodology, then **it is hypothesized that the self-configuration of MAS is better achieved through the use of a distributed bottom up approach.** While heuristic search and linear programming methods are able to solve these kinds of configuration problems, they require quite complex models and are difficult to define and maintain. These solutions are also very specific and non scalable, which makes their applicability not very good in a constantly evolving domain (Onori and Oliveira [18]). Furthermore, they apply a top down approach which only takes limited to no advantage of the hierarchical nature of the problem. Therefore, this work proposes a distributed bottom up solution for solving of this configuration problem.

The use of agent technology is viewed in the literature as the natural approach for bottom up problem solving (Jennings and Wooldridge [11]). Therefore, it is hypothesised that **by creating a multi-agent solution for the bottom up solving of this configuration problem maximising the parallel computation and taking advantage of the latest negotiation protocols to achieve a goal oriented behaviour of the overall configuration environment.** The choice of agent technology is also supported because of the modular nature of the problem, with the added advantage of providing scalability option and future enhancements. It also provides the basis for distributed computing built in, which in computer intensive processes is crucial for viable solutions. Therefore, it is proposed that the development of an agent architecture and respective models, will provide the basis for solving the configuration problem, while providing means to deal with future advancements in the MAS domain. In addition to this, it also provides the ability for different equipment module vendors to define individual equipment module rules (which can be shielded from other vendors) for actively seeking for participation in MAS solutions.

The final step in the hypothesis definition is the emergence of solutions through the agents interactions supported by simple lower level rules that support their decisions.

It is therefore hypothesised that the collaboration of the agents using basic rules will enable the emergence of complex solutions.

The research hypothesis requires three major elements which are the knowledge contributions contained in this work. These are the **Requirements Model for Agent-Based Self-Configuration of Modular Assembly Systems**, the **Agent Architecture for Distributed Self-Configuration Methodology for Modular Assembly Systems** and the **Local Behaviour Models for Distributed Self-Configuration Methodology**.

3.2.4 Research Methodology

The work presented in this thesis followed a systematic methodology presented in **Figure 3.3**. The first step of this work consisted in an extensive literature review of the MAS domain. This literature review was only a partial view to establish the current state-of-the-art in the field. In addition to that, the input from the industry through a collaborative research project (EUPASS [4]) was a crucial source of information that in conjunction with the academic work in the field provided a good starting point to establish a problem definition.

The problem definition provides a clear view of the domain which enables the identification of a clear set of research requirements that provide the set of conditions for which the hypothesis will be validated. In addition to the research requirements, a set of research objectives was also extrapolated from the knowledge gaps found in the literature. The combination of these, offer the basis for the definition of the research hypothesis which sits at the core of this research methodology.

The hypothesis of this work was broken down into sections which result in the knowledge contributions contained in this work. The first contribution focuses on the means to elicitate the MAS requirements and the equipment module descriptions, proposing a model to describe these aspects. The second contribution addresses the need to achieve configurations while catering for the scalability of the MAS domain, through the use of agent technology architecture that is designed for this purpose. Finally, the last contribution is the creation of a self-configuration methodology for MAS, which consists of a set of methods and beliefs that enables the agent interactions which lead to configuration solutions.

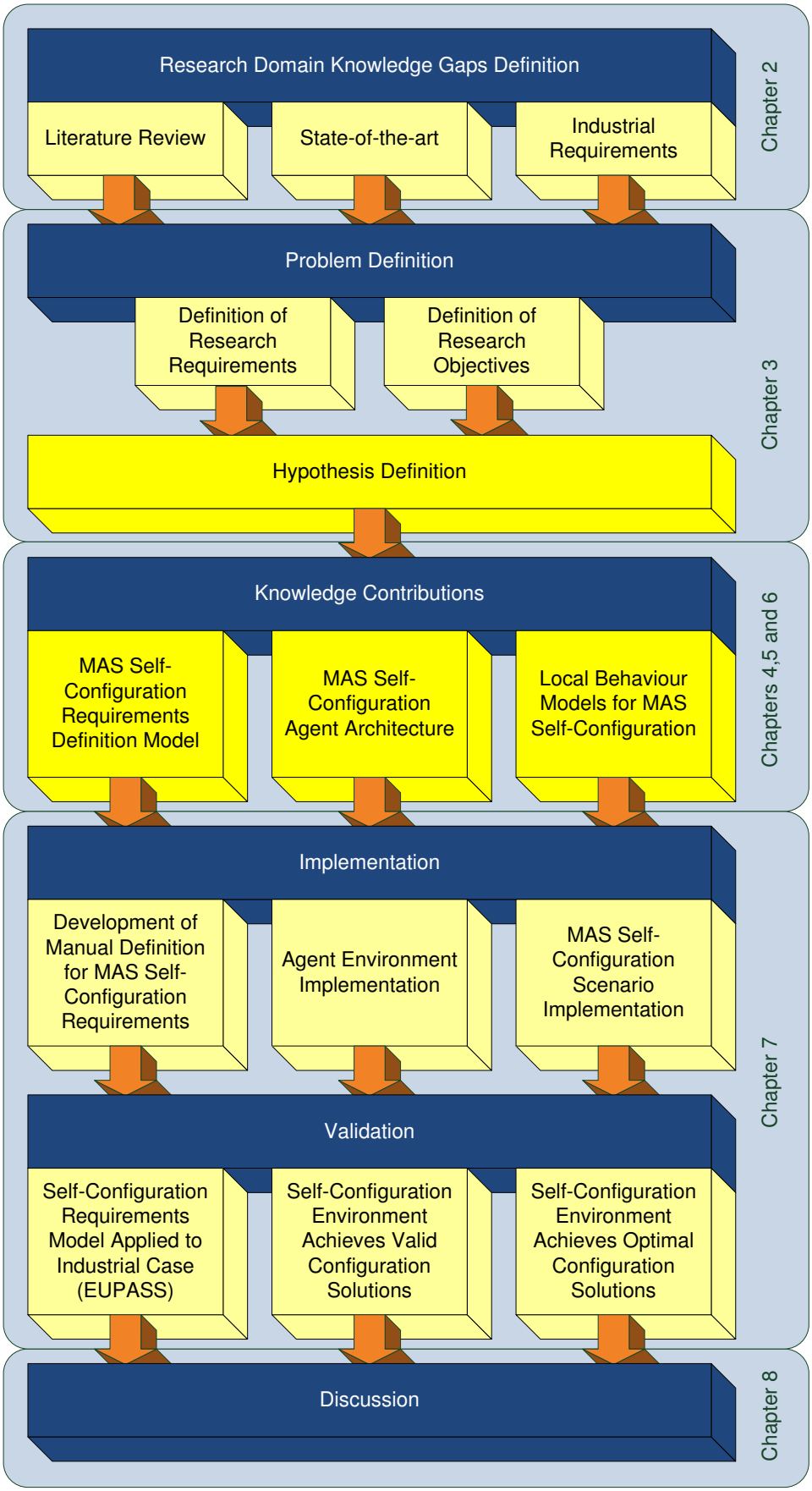


Figure 3.3 - Research Methodology Overview

Once the hypothesis was broken down into the core contributions, it was possible to establish validation procedures for each of these contributions. The definition of validation scenarios for a wide range of configurations problems is outside of the scope of this work, therefore the validations of each contribution focus on the available examples provided by the collaborative research project (EUPASS [4]). Each contribution will require its validation against their individual research objectives.

The final stage of this work is the critical discussion or conclusion where the information achievements, the limitations and future work is discussed.

3.2.5 Requirements Model for Agent-Based Self-Configuration of Modular Assembly Systems

The formalisation of models that enable the clear and structured capture of the different aspects required for the configuration of MAS is the first knowledge contribution contained in the work. The justification for its existence is simple, without a clear model that can be computer interpretable it is not possible to establish any method to support the configuration process. **Figure 3.4** provides an overview of the models required to enable the self-configuration methodology while highlighting the involved actors.

The actors identified are the equipment module supplier, the configuration expert and the system integrator. It is important to identify the actors since they are the source of all the information that is required to formalise the models. The analysis of the individual contributions in the MAS domain allows for the formalisation of these contributions. The proposed model will be established after an analysis of current state-of-the-art configuration procedures in the scope of the collaborative European project EUPASS.

In the proposed model seen in **Figure 3.4**, the equipment module suppliers will be required to supply their module description following a specific format that adheres to the common concepts and terminologies. Similarly the system integrator will also define the MAS requirements following a specific format that also adheres to the common concepts and terminologies. However, without the establishment of common concepts and terminologies it would not be possible to map the

requirements to the existing capability. Furthermore, the concepts will require updates as the domain evolves. To address these issues, it is proposed the creation of a new role of configuration expert that is able to add, change and update these concepts and terminologies. Finally a data model needs to be created for the solutions to be presented to the system integrator.

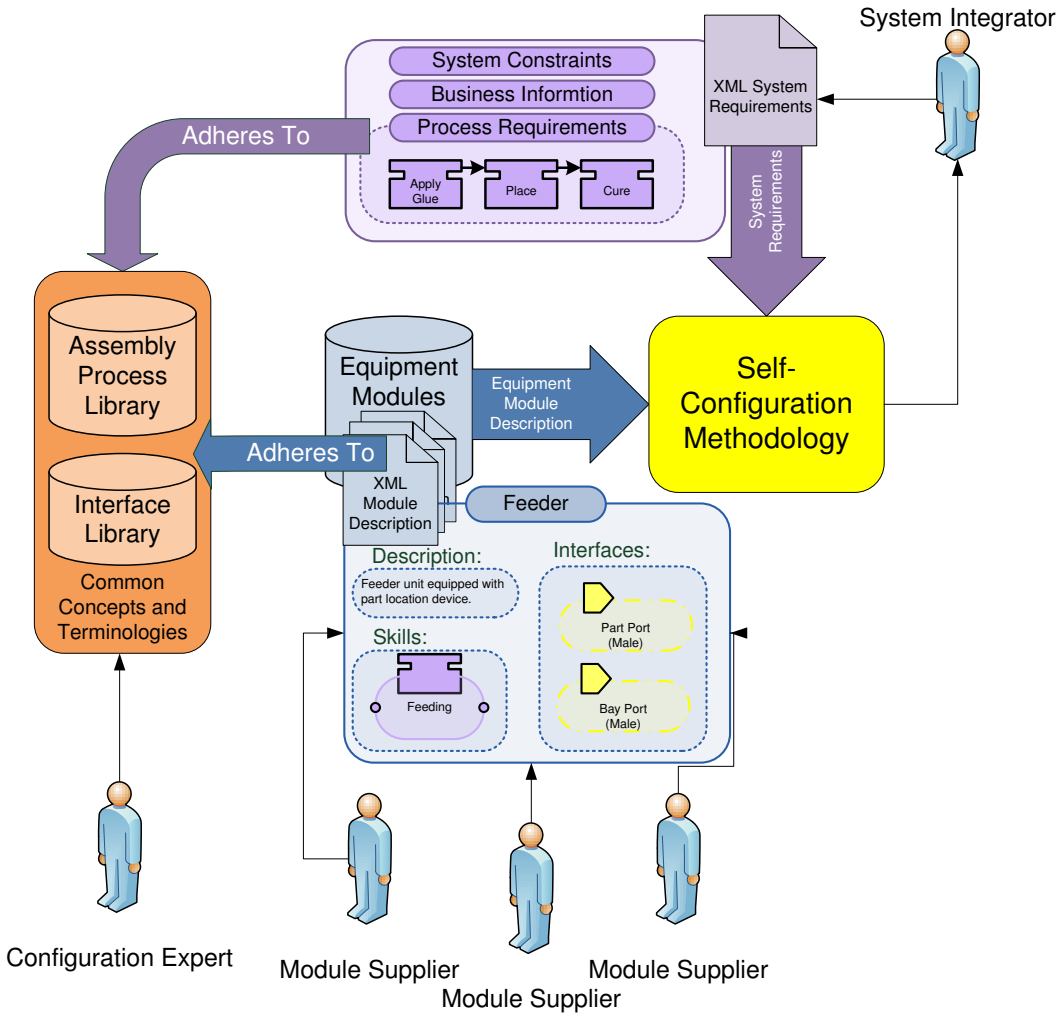


Figure 3.4 - Overview of Requirements Model for Agent-Based Self-Configuration of Modular Assembly Systems

In sum, the requirements model for the self-configuration of MAS will include models for assembly process and interface libraries, for the definition of MAS requirements, for equipment module descriptions and for the description of MAS configuration solutions.

3.2.6 Agent Architecture for Distributed Self-Configuration Methodology for Modular Assembly Systems

The creation of an agent architecture that is able to represent the aspects of the MAS configuration problem is the first step in the creation of the proposed bottom up distributed self-configuration methodology. The basic notion of this proposal is that agent technology can enable the creation of this methodology. For this, one needs to create agent types, roles and a structured hierarchy that is able to accurately structure the configuration problem.

The design of a multi agent architecture requires a structured approach. In the literature there are a couple of methodologies for the design of multi agent solutions. The majority of the existing methodologies are domain specific, however the GAIA methodology provides a good generic approach for the architecture design that has proven itself in computer science domain (Zambonelli et al. [123]). The design of the agent architecture based on the GAIA methodology requires firstly an analysis of the problem, namely the clear definition of the objectives and targets that the agent system has to address. The first step is the understanding of the requirements for such system, specifically the identification of what needs to happen and what information is required. The required information was already identified in the Requirements Model for Agent-Based Self-Configuration of Modular Assembly Systems. In this, requirements are established that have an impact on the objectives of modular assembly system configuration and reconfiguration. The main objective of the system is to provide valid solutions for the configuration and reconfiguration of MAS.

Once the objectives are defined, the next step in the design of a multi agent architecture is the analysis of the problem. This establishes the need for the definition of the agent types, roles and expected interactions.

The nature of the MAS paradigm provides a clear focus on functional decoupling of equipment module functionalities and standardised interfaces for interchange ability. This enables the formalisation of functional capabilities and connectivity constraints of the available modules hence allowing the mapping of requirements against available capabilities. This clearly identifies the two main agent types required for

the self-configuration methodology agent architecture based on the different objectives: the **Equipment Module Agent** and the **Requirements Agent**. This decoupling into two agent types, uses principles from blackboard architecture model, where two agent types come together to solve a problem, the difference being these will have a structure and common understanding of the relations between the two aspects of the configuration problem. These agents will provide the basic functionality required for solving the configuration problem. The **Equipment Module Agents** provide representation of equipment modules, which can interact with each other to establish collaborations that represent configurations. The **Requirements Agent** will provide the objectives that motivate the Equipment Module Agents interactions, while also being able to evaluate the solution based on the requirements established by the system integrator. **Figure 3.5** provides a conceptual overview of the agent architecture for the self-configuration methodology, where all the agent types and respective hierarchies are established.

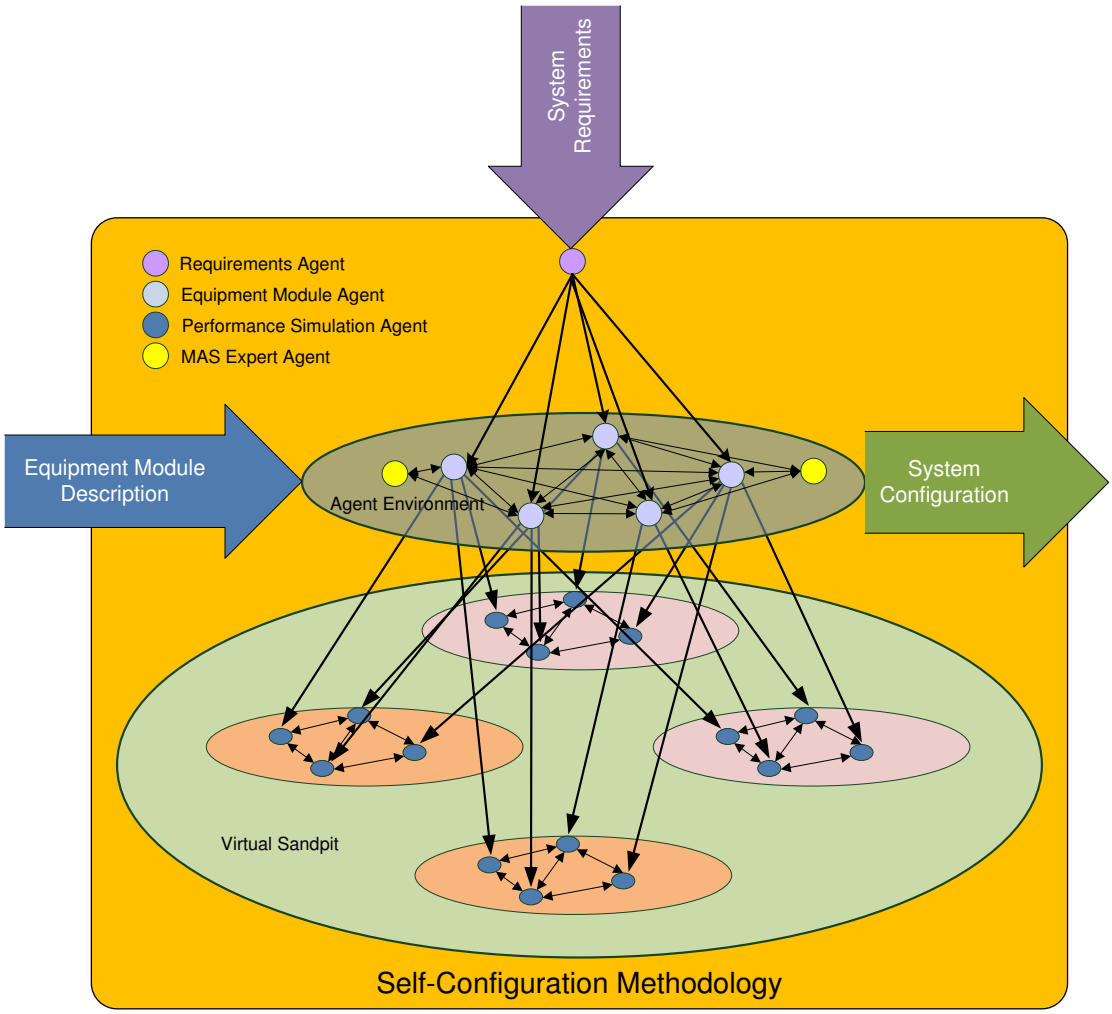


Figure 3.5 - Agent Architecture for Distributed Self-Configuration Methodology for MAS

The nature of agent technology allows for the distribution of decision making processes that would be computer intensive through the creation of child agents, therefore taking advantage of distributed computing. The **Performance Simulation Agent** is introduced into this architecture to provide a decoupling of one of the most computer intensive problems, the simulation of given set of configurations for selections. It is easy to understand how the computer processing requirements grow exponentially if simulations for the performance characteristics are required for all solution possibilities.

The final aspect of the analysis of the configuration problem has to do with assessing the logical conditions of the configurations based on its internal knowledge model. The issue is, if this knowledge was built in to the configuration methods, e.g. the internal decision making models of **Equipment Module Agents**, future changes might require a complete change of the configuration methodology. Therefore, it is proposed that this knowledge is decoupled into the **MAS Expert Agent**, which can be changed or even replaced in order to cater for the evolution of this knowledge.

3.2.7 Behaviour Models for Distributed Self-Configuration

The creation of agent architecture is followed by the detailed design and implementation according to the GAIA methodology. Thus, establishment of the behaviour for each agent is the final piece for enabling the methodology. The behaviour builds on the previous contributions providing the specific methods that enable the operation of the multi-agent solution. **Figure 3.6** provides an overview of the concepts involved in this definition, which sit at the core of the decision making process on finding MAS configuration solutions.

The definition of the agent behaviour is based on the specific roles and interactions established in the agent architecture. The interactions impose the first and most important aspect in a multi-agent solution, the establishment of the interaction protocols that define the rules and means for agent interaction. These will provide the basis for the agent behaviour in relation to other agents. It is a crucial definition since agents will only be able to deal with the protocols which are known to them. The protocols are closely related with the collaboration agreements, which are a set of

formalization rules that are triggered within a protocol sequence. The models for information exchanges between agents will provide the basic information for the agents to make decisions.

The protocol definitions will provide the overall agent system behaviour, however this is only one of the steps required to enable the distributed decision making process. For this, it is important to clearly formalised the agents beliefs since they drive their decision making process. The formalization of the decision of each individual agent will provide the missing elements of the self-configuration methodology, namely how each agent type behaves based on a set of information. The distributed decision making agent architecture raises two important issues for the individual agent behaviour, namely on how to capture and use MAS expert knowledge and how to establish a performance simulation of potential solutions. These require a set of models that enables the agent behaviour and operation that addresses aspects that have an impact on the self-configuration methodology but are in a different domain. This is the basis for the definition of the **Performance Simulation Agents** and **the MAS Expert Agent** behaviour. This will entail the creation of a new model and method to establish modular components that can be distributed with a set of rules that enables the simulations of assembly characteristics, and the means to capture MAS configuration knowledge.

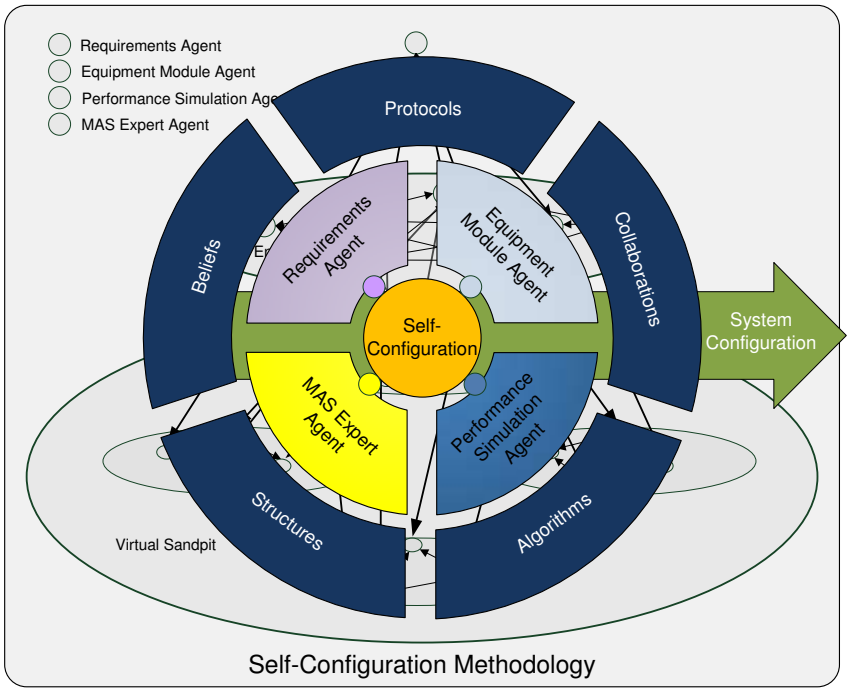


Figure 3.6 - Overview of Behaviour Models for Distributed Self-Configuration Methodology

The creation of all the agent behaviours will culminate in the establishment of the self-configuration methodology that caters for both configurations and reconfigurations of MAS.

3.3 Definition of Validation Methods

The definition of validation methods in any research requires an analysis of the domain. The proposed solution targets a domain that is quite vast, complex and expensive to validate for all existing MAS systems. Therefore, the validation of this work will focus on a set of representative scenarios that reflect the key problems and characteristics in the domain of MAS configuration. The complete validation of the proposed methodology for the whole domain is outside of the scope, and in practice could only be done in industry.

The validation of methods for the proposed agent based self-configuration methodology for MAS will be broken down into three main parts, which represent the three knowledge contributions.

The requirements model for agent-based self-configuration of MAS will be validated through the incorporation of the model in a manual configuration tool, which will be used by both academic and industrial experts to define requirements for MAS, equipment modules and system configurations in a collaborative project (EUPASS). This tool will be developed with the proposed model as its base, and will be used to perfect the model to cater for inputs from the expert users. This provides a good validation platform for this model. This data will also help to populate the equipment module library with available equipments and expert selected configuration solutions that can be used in the validation of other methods.

The agent architecture for distributed self-configuration methodology for modular assembly systems validation cannot cover all possible MAS in existence, therefore the focus will be on the operational side. The first validation is the demonstration that the designed architecture in operation works. This will entail the demonstrations of the different operational states of agents as described in the architecture. The second

aspect of validation and perhaps the most important is the demonstration that this is a good computational approach for solving the configuration problem.

The behaviour of self-configuration of modular assembly systems through agent technology validation will be achieved through the verification and logical analysis of the results derived from the proposed methods, given a set of MAS scenarios. The methods account for possible adjustments that ensure the testing of scenarios under different conditions, namely using exhaustive and heuristic approaches which provide the basis for comparing different configurations of the approaches. This will enable the identification of best practices for the operation of the method while validating that it works for all given scenarios. It is expected that the results for this validation will focus on two aspects, MAS configuration solutions and MAS performance simulation results.

3.4 Chapter Summary

This chapter gives an overview of the research methodology and highlights the motivations behind this work. The chapter presents the definition of the problem that this thesis addresses and formalises the research approach. The hypothesis for this work has been formalised and described, also detailing an overview of the knowledge contributions of this work. Finally the validation strategy for this work has been presented.

4 Model for Agent-Based Self-Configuration of Modular Assembly Systems

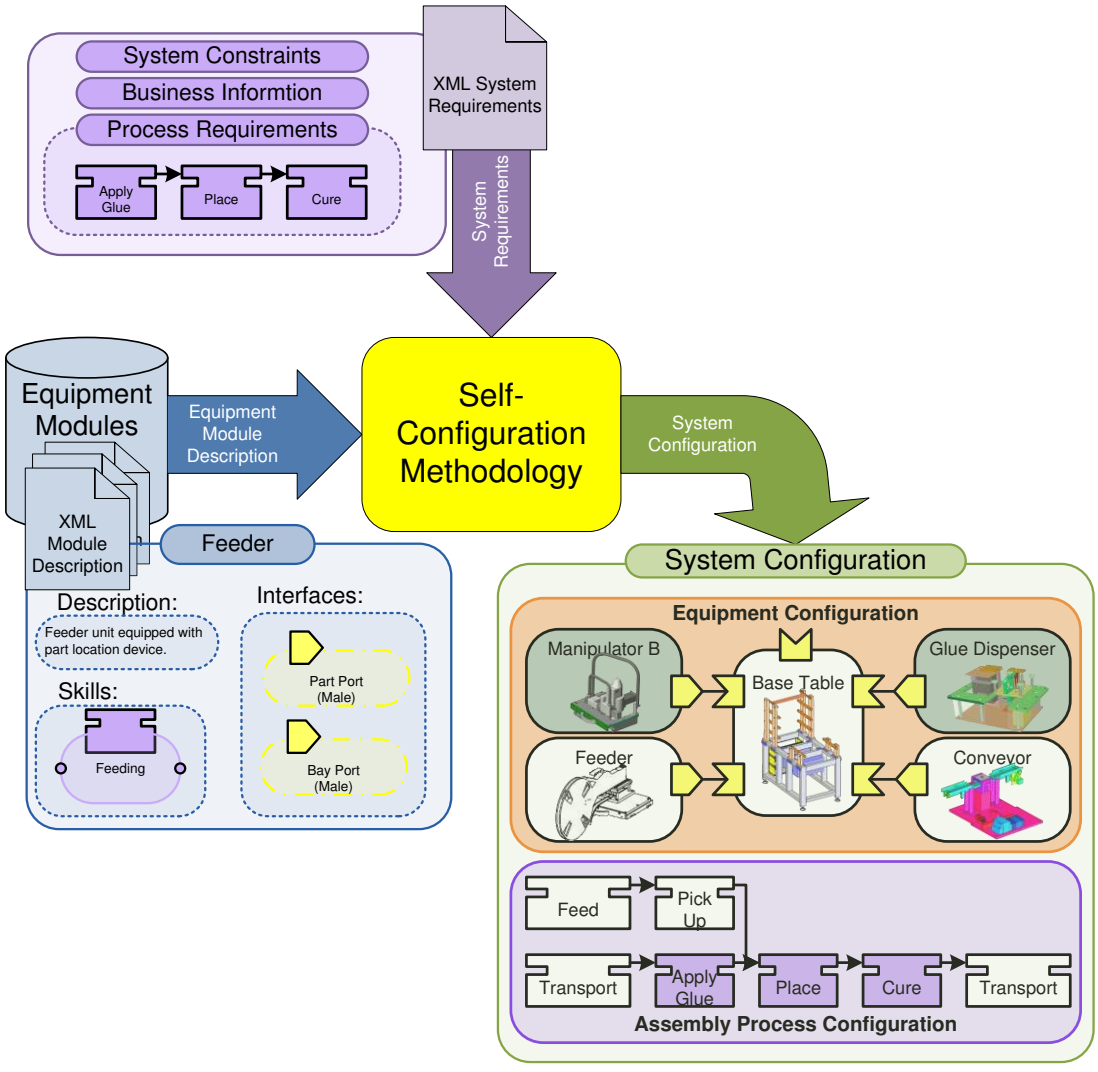


Figure 4.1 - Overview of Boundary Conditions of the Problem Domain

4.1 Introduction

In this chapter it is proposed a model that encompasses the requirements description that enables an agent-based self-configuration of modular assembly system. The chapter covers all the different sources of requirements, highlighting the need for clear and their formal definition which will enable the self-configuration of modular assembly systems.

The concept of modularity is highly dependent on the general understanding of a module. A module is a building block with certain characteristics, both physical and logical, that enable it to be combined with other modules. The question that arises from this definition is what these characteristics are, and more importantly whether they can be generalised. It is clear that different modular systems can have different characteristics. The domain of modular assembly system is quite wide and complex, and there are several different types of solutions. The challenge is to find the common characteristics and establish a clear model for their descriptions. Despite the fact that this target domain is quite wide, what can be clearly generalized is that to establish any configuration methodology for a modular system, one requires a module description, which has to contain information on its capabilities and how the module can be combined with others.

Modular assembly systems (MAS) have existed for over two decades. They focus mainly on the advantages of fast physical integration of equipment. Nowadays, system builders often use the concepts of modularity on the physical side. Standardization is a complex and lengthy process and, in the case of assembly, quite impossible to tackle. However, the need for standards does not provide an obstacle for system configuration. If one equipment only plugs in to another of the same supplier, it is not ideal for the future of MAS, but it does not pose a problem for configuring a system, since there is a set possible solutions. The real need that arises from analysing the standardization issue is the need for a storage of terminology that should be used by different module suppliers, regardless of it being shared definitions or not. The intention of this chapter is to provide a model that is module

supplier independent, catering only for the aspects relevant to the configuration of modular systems.

In the assembly domain, we should consider two aspects of modules, namely their physical characteristics and their assembly process capabilities. Assembly modules exist to perform certain activities, which can be described as its capabilities or skills (EUPASS [4]). The capabilities of the module can themselves be described as a module, a different type of module but still a module. This is important to note because these modules also have their connection issues that are crucial for the definition of the assembly process sequence (Lohse [44]).

The context of system configuration has to be driven by a clear set of requirements.

For the assembly system requirements an important aspect is to maintain certain common terminologies with the equipment modules descriptions. These have been identified as the assembly process capabilities and physical interfaces (EUPASS [4]). The physical interfaces allow for the definition of physical requirements and constraints. The assembly process capabilities allow for the identification of which modules can fulfil the capability requirements. The assembly processes descriptions should also follow a common taxonomy to enable the possibility of high level assembly process requirements, which will be complex compositions of assembly processes. This definition provides the basis for making configuration decisions based on clear hierarchical assembly process structure (Lohse [44]).

The choice of agent technology poses a few constraints on the configuration process since agents are required to communicate. Consequently, the information needs to be transparent for the agents to be able to exchange information and make decisions based on the semantic descriptions. The configuration of an assembly system is a process that involves different information types which must be modelled for the use of agents. It is important that the model is structured in a scalable manner, catering for possible modular assembly systems evolutions. Moreover, it is very likely that new assembly processes, new interfaces, and new equipment module types will be introduced over time.

The definition of clear assembly process and system requirements is crucial for the design of an agent system that will provide configuration solutions. If the

requirements are not defined there are no clear objectives to establish the configuration. The established requirements will define the information that can be inputted in the agent environment. This information also enables the decision making process of the agents, namely for establishing valid configuration solutions. In addition, it highlights the boundary condition of the problem this work intends to tackle. **Figure 4.2** provides a high level overview of all the descriptions that are required by the configuration methodology. The proposed model is supported by three types of actors, which highlight the requirements of the agent environment. The system integrator provides the definition of the system requirements, namely what are the expected capabilities and constraints for the desired system. This description will follow a complex skeleton or template which imposes the required common terminologies. The next user is the configuration expert, which provides the required maintenance to the model in terms of common terminologies for both the interfaces and the assembly process. The final user is the module provider, who is responsible for populating the equipment module library (providing descriptions that follow the established terminology). It is also important to point out that the output of the configuration methodology will be provided in a structured manner to the system integrator for validation. The details of these models will be presented throughout this chapter.

The use of Extensible Markup Language (XML) format is proposed for the instantiation of the proposed models definitions, because it allows for a clear description that is transferable and usable across different systems. XML description is transparent and understandable format for both individuals and computer systems. The use of XML format provides a transparent description which is widely accepted for transfer of information, and is also able to cater for future extensions which enables the scalability of this approach. The wide use of XML also allows for a better acceptance of industry for the use of this approach.

The use of XML means that in order to encapsulate this information, the model will be provided in a XML Schema language known as XSD. This form of description is at its core hierarchical since it is based on the definition of nodes, with certain attributes, that contain other nodes making it hierarchical form of description. Some nodes and information might be optional in some cases, and mandatory in others. The full XSD model can be found in **Appendix A**.

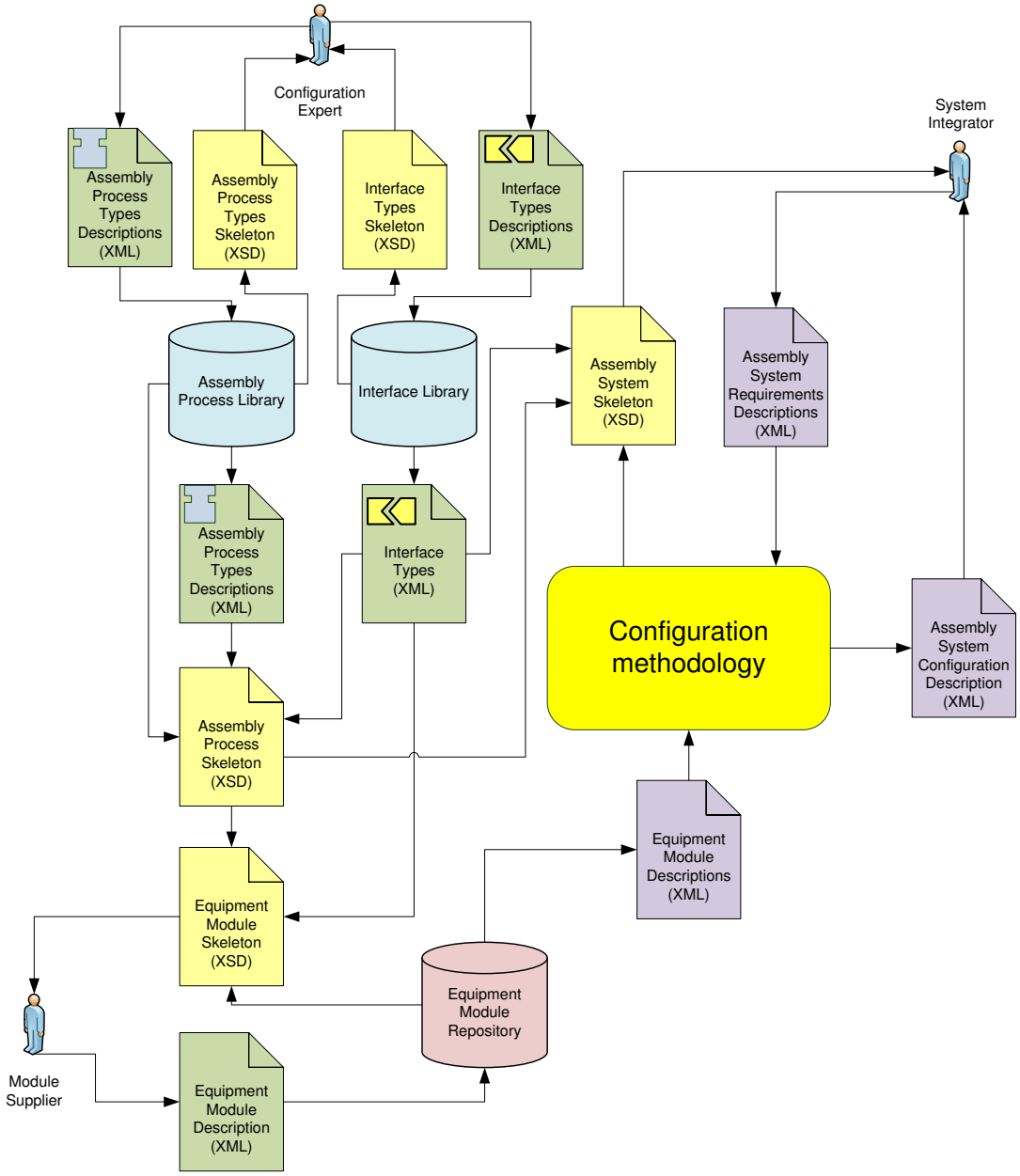


Figure 4.2 - Overview of Self-Configuration Requirements Model for Modular Assembly Systems

4.2 Agent Technology Requirements Identification

The analysis of the requirements is the first step in any methodology to define a multi agent system. Its first step is the problem definition, which will provide the boundary conditions of the agent environment. The boundary conditions will establish what

information needs to be introduced into the agent system in order for it to provide the expected results.

The configuration of MAS requires clear definition of both system requirements and equipment module description. Therefore, clear models have to be defined to tackle these aspects. However, simply providing descriptions is not enough for the agent environment. These descriptions are required to share the same common semantic model, otherwise the agent environment will not be able to correctly interpret the information. Consequently, the agent environment requires both structure and meaningful information, hence it is required to establish a common semantic model, and the structure for both equipment module descriptions and system configuration requirements.

The agent environment intends to provide the configuration of MAS. However, the form of the solution is not a trivial matter. The details for establishing configuration description of MAS are themselves quite complex. This highlights the need for a clear model for describing the configuration solution. The model will simply build on the already available terminology and structures, and provide a clear system configuration description.

4.2.1 Common Semantic Notation Definition

The common semantic model is a set of elements intended to serve as the language that will permit specification of the meanings of any domain term or concept. The common semantic model is a crucial element for building any type of distributed decision making system (Nwana [62]).

The first stage of the MAS configuration process is the identification of which modules can perform the required assembly processes. This matching activity provides the first clear need for semantic notations, the assembly processes (or Skills). The assembly processes need to be matched between the configuration requirements and the equipment modules, as such both need to use the same terminology for each assembly process. The existence of taxonomy for the assembly processes is also quite important, since it provides a wider matching based on hierarchical definitions (Lohse [44]). This also enables the concepts of elementary assembly processes and composite assembly processes(Onori et al. [7]; Lohse [44]).

The notion is quite simple; composite assembly processes are composed of lower level assembly processes. This also provides a basis for a better matching between the system requirements and the equipment modules.

The second need for semantic notation is the specification of interfaces for establishing connections between equipment modules and also between the assembly processes. The configuration of MAS will require modules to be connected with others, as well as assembly sequences which require connections between different assembly processes. The identification of what can be plugged together can only be determined if a common terminology exists for the definition of interfaces.

It is clear from the above discussion that the terminology for the skills and interfaces, needs to be defined in a uniform manner to enable the creation of the configuration methodology. As such the creation of two types of repository, an Interface Library and an Assembly Process (Skill) Library is being proposed (as showed in **Figure 4.2**).

4.2.2 Common Taxonomy and Terminology

The descriptions provided so far for the assembly processes do not cover any sort of classification. Without a clear assembly process classification it would not be possible to address elementary and composite assembly processes (Lohse [44]). An assembly process classification establishes a hierarchical view which enables high level assembly processes that can be composed of lower level assembly processes. The established hierarchy allows for semantic reasoning because it gives bigger depth to the information. If, for example, an assembly process A is hierarchically above assembly process B, then if A is required then B can also be used. This depth contributes to an easier establishment of system configuration requirements, providing the mechanism to define high level assembly processes. The high level definition of assembly processes simplifies the requirements and leaves more leeway for the configuration methodology to provide the low level solution. In (Lohse [44]), an assembly process classification is proposed for modular assembly system fitting the requirements of the configuration methodology. However, this classification has a limitation, namely it does not cater for how processes do or do not affect the product. The way the product is affected is quite important for the assessment of

certain MAS characteristics, namely repeatability and accuracy. Therefore, it is proposed in this work an extension of the characteristics for reasoning about how processes do or do not affect the product, and also the inclusion of MAS configuration characteristics. These extensions will not change the core of the classification, but a new classification is introduced in the form of an attribute that establishes information to allow for repeatability assessment by the configuration methodology. As such, the use of a high level process definition, which is able to represent a large number of sub processes with similar characteristics, has been proposed. The following classification of product relevant process types has been identified and proposed in (Ferreira et al. [128]):

- Qualifying Process: Sensing processes, vision systems, etc, that enable the compensation of stack up errors.
- Fixating Process: Processes that attach two or more components together, which will result in the inability to compensate for the current errors.
- Decision: Processes that require certain thresholds, thus providing a certain guarantees to the product characteristics.
- Other Process:
    - Compensate Process: This is a characteristic that each process has and provides the possibility of classifying the processes that are able to compensate for error based on their specificity, namely certain types of gripping.
    - Non-Compensate Process: All other processes that do not fall in the previous category are considered error stack up processes.

This work builds on the work of (Lohse [44]) in this domain, which provides the base terminologies and classification of assembly processes. This work proposed the enhancement of Lohse's classification by providing the previous extra attributes to the classifications. These attributes provide a functional meaning to the different types of assembly processes, which enables the assessment of MAS characteristics for a given configuration. The characteristics that can be assessed are the repeatability and accuracy which will require the creation of dedicated synthesis methods, which provide detailed repeatability and accuracy results which in turn can be used for the decision making process of the configuration methodology.

4.2.3 Definition of Assembly Process Skills Library

The creation of a repository with common terminology and clear taxonomy is fundamental for the operation of the self-configuration process. The need for a repository is supported by the nature of the modular concept. The nature of the concept allows for a constant update of new modules that might require new assembly processes due to new technological advances. The only way to allow for a scalable self-configuration method is to provide a way of describing new concepts in an understandable fashion. The purpose of creating a repository is to provide scalability to the self-configuration method by creating a placeholder for assembly processes that is common across all module vendors and system integrators. This is supported by the impossibility of providing a list of all existing and future assembly processes. This also highlights the importance of classification of assembly processes, since it provides a high level placeholder that facilitates the understanding of newly introduced assembly processes. It is proposed, that providing a transparent structure for the definition of assembly processes that can be used for current and future assembly process definition, will provide future scalability of this configuration approach.

The Assembly Process Library is defined as the repository that contains all possible assembly processes considered for the configuration. Thus, if a given assembly process does not exist in the repository, it is not possible to be defined as an attribute of an equipment module or include it in configuration requirements.

4.2.4 Standardized Assembly Process (Skill) Descriptions

The description of an assembly process is in line with the new IEC 61499 Standard which provides an object oriented control structure and reuse of program logic for PLC's. The standard is applicable to the system control, which consist in the operation of the MAS. This is the step that occurs after the process of configuration. The use of this function block concept in the configuration methodology allows direct mapping between the configuration of the system and the control of the system. If the same function block descriptions are used both for configuration and for control, then the setting up of the system control will be shortened. **Figure 4.3** provides an overview of the conceptual assembly process block, the main

characteristics of the Assembly Process description for the purpose of configuration are:

- The Assembly Process type – which has to be extracted from the existing list of assembly process types within the Assembly Process Library, so it is included in the assembly process template.
- The Control Ports – which provide control variables, both inputs and outputs, for operating the assembly process (namely: Start, Interrupt, Finished, etc.). These also provide the means for plugging together two different assembly processes. This will enable the establishment of the control sequence.
- The Parameter Ports – which exist for assembly processes interactions, providing complex data when present. A typical use of these is a force feedback loop, where the value of the force would be passed on to other processes via a parameter port. In other word this provides the information flow of a given assembly process configuration.

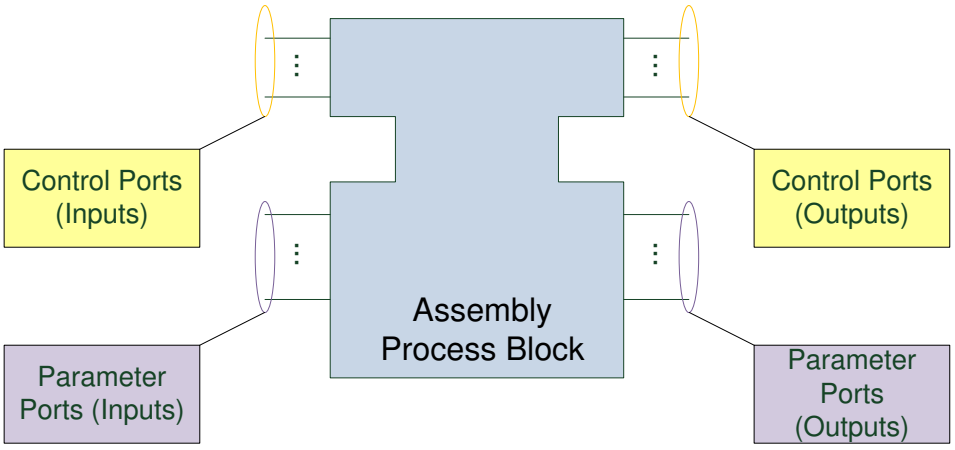


Figure 4.3 - Overview of conceptual assembly process block

The Assembly process needs to be able to encapsulate all of the characteristics, but also to be structured in a way that can be enhanced in the future to add other characteristics. The proposed assembly process description establishes a main node for the assembly process, which contains a set of attributes (Name, AssemblyProcessID and Description). The node will contain five child nodes to provide the additional information that can be common across other assembly processes, namely Assembly Process type, Control Ports, Parameter Ports,

Configuration Characteristics and Composed. These can be seen in the overview of the XSD description provided in **Figure 4.4**.

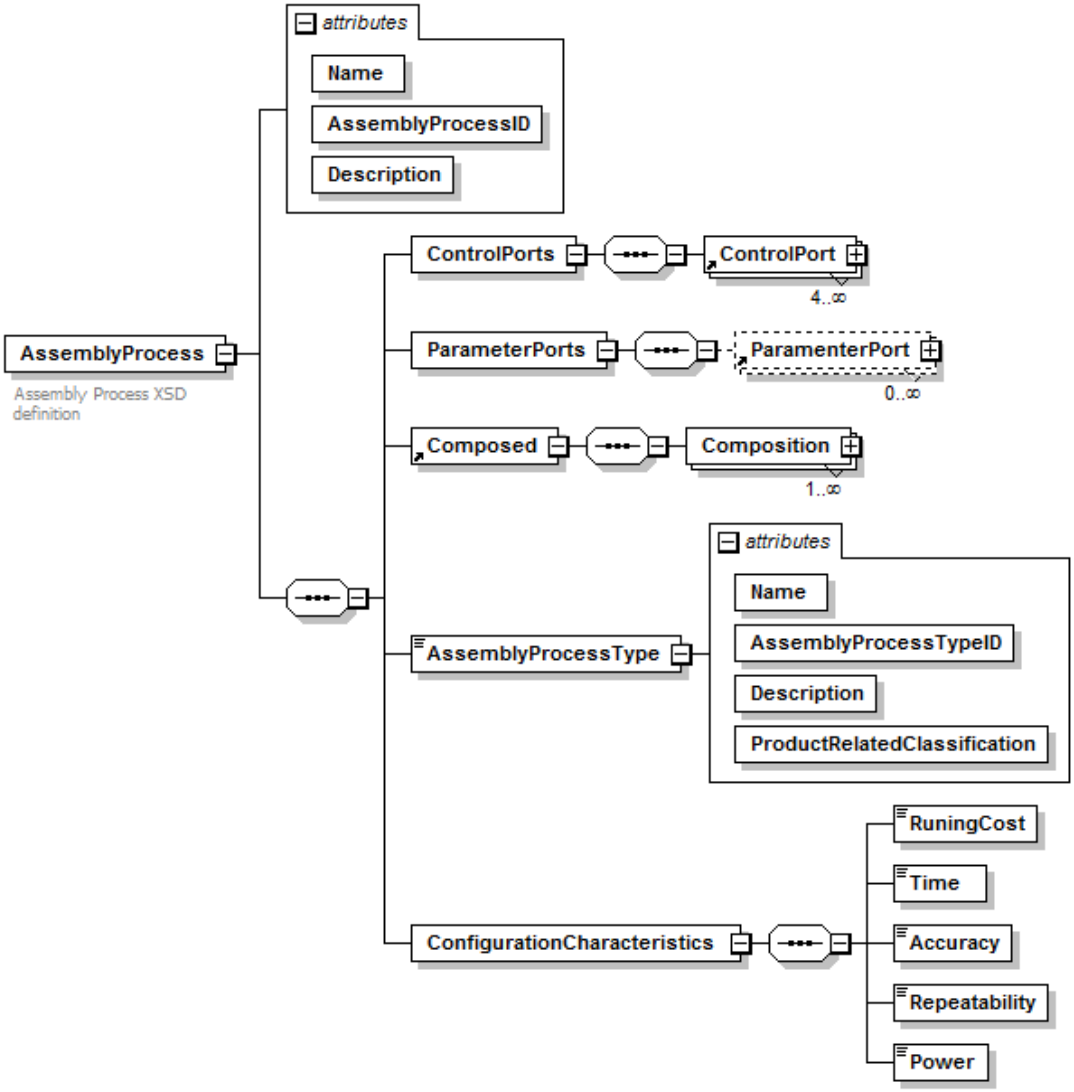


Figure 4.4 - Overview of Assembly Process Skeleton (XSD)

The Assembly Process type is linked with the Assembly Process Library, therefore the template will contain a list of possibilities, which restricts the choice of assembly process type to the types that exist in the Assembly Process Library. This enforces the use of the same terminology to define the same type of skill.

The Control Ports node contains several nodes of the type “Control Port”. This enables the possibility of an infinite number of ports. However, at least four are always required, namely two inputs, Start and Interrupt and two output, Finished and

Error. Each port will contain the interface information which provides the information for the other control ports which can be plugged into the current one.

The Parameter Ports is an optional node that contains several nodes of the type "Parameter Port". In simple assembly processes it is expected that this node will not exist. However, in more complex assembly processes this will be required. When this node exists it will also require interface information for establishing to what other ports it can be plugged into.

4.2.4.1. Assembly Process Configuration Assessment Characteristics

The assembly process description needs to contain configuration relevant information. It has been identified in the literature that the most important parameters for the configuration methodology are Time, Cost, Quality and Flexibility (Chryssolouris [14]). All these aspects could be related to the assembly process. However, it is considered that only characteristics that are intrinsic to the assembly process should be associated with it. For the purposes of this work, **flexibility has been defined as the spare capabilities within a given system**. Therefore, it is not specific to an assembly process but rather to the non used assembly processes present in the given system.

The Cost of the process could be viewed as a relevant characteristic, but the process cost would have a marginal impact on the system cost. However, a system running cost can be inferred if its cost is defined for each assembly process. This information should be an average value by a unit of time. So, the first required characteristic of an assembly process is "Running Cost" and is defined as:

"The average cost per unit of time to activate and run a given assembly process"

The execution of an assembly process always has a time constraint, as "Time" is required for performing any assembly process. Therefore, "Time" is another important characteristic that needs to be established for each process in order to determine the cycle time of a given system. Consequently, "Time" is the second required characteristic of an assembly process, and is defined as:

"The average time required to perform an given assembly process"

Quality is a concept intrinsically related to the product. However, accuracy and repeatability are assembly process characteristics which have an impact on the

quality of the product. Furthermore, these are characteristics that are related to the defined flexibility of a system. Therefore these need to be included in the assembly process description. To do so it is required to clarify what is the difference between accuracy and repeatability, which can be summarized by **Figure 4.5**, where an overview of the two concepts can be seen.

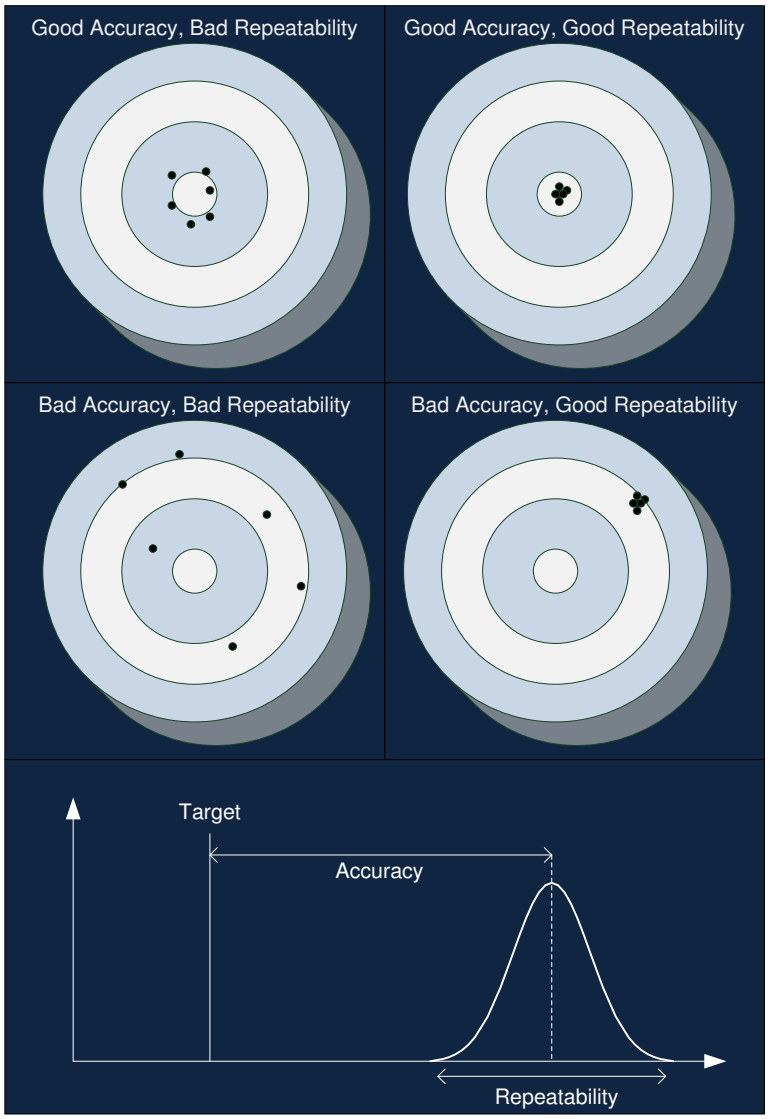


Figure 4.5 - Overview of Distinction between Accuracy and Repeatability

Accuracy tells us how close a measurement is to achieve its intended target. The difference between the target and the achieved result is the accuracy of the assembly process. Thus, "Accuracy" is the third required characteristic of an assembly process and is defined as:

"A numeric value that establishes the average difference between the achieved assembly process result and its intended target"

The repeatability of an assembly process is its ability to achieve the same target in a repeatable manner. The perfect precision is achieved if an assembly process is capable of obtaining the same result, regardless of the number of times it is executed. So the repeatability is the deviation obtained when performing repeatedly the same assembly process under the same conditions. So, "Repeatability" is the fourth required characteristic of the assembly process and it is defined as:

"The deviation of results obtain from running the assembly process under the same conditions several times"

The configurations characteristics described should appear in the format of a statistical distribution. This will provide better insight into the assembly processes, allowing a more realistic description of the assembly process. This will also allow more information for system optimization issues. The normal distribution will be used in this work, which can be seen in **Equation 1**. This distribution has two variables that enable its definition the mean value (μ) and the variance of that value (σ). Thus all values for the configuration characteristics will have the following form:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Equation 1 - Normal Distribution (Snedecor and Cochran [129])

4.2.4.2. Composition of Assembly Processes

The Composed node provides the means to define composite assembly processes. These composite assembly processes are composed of more elementary assembly processes. This allows for the high level definition of assembly processes and provides the basis for combining assembly processes into new assembly processes. **Figure 4.6** provides a schematic overview of this concept.

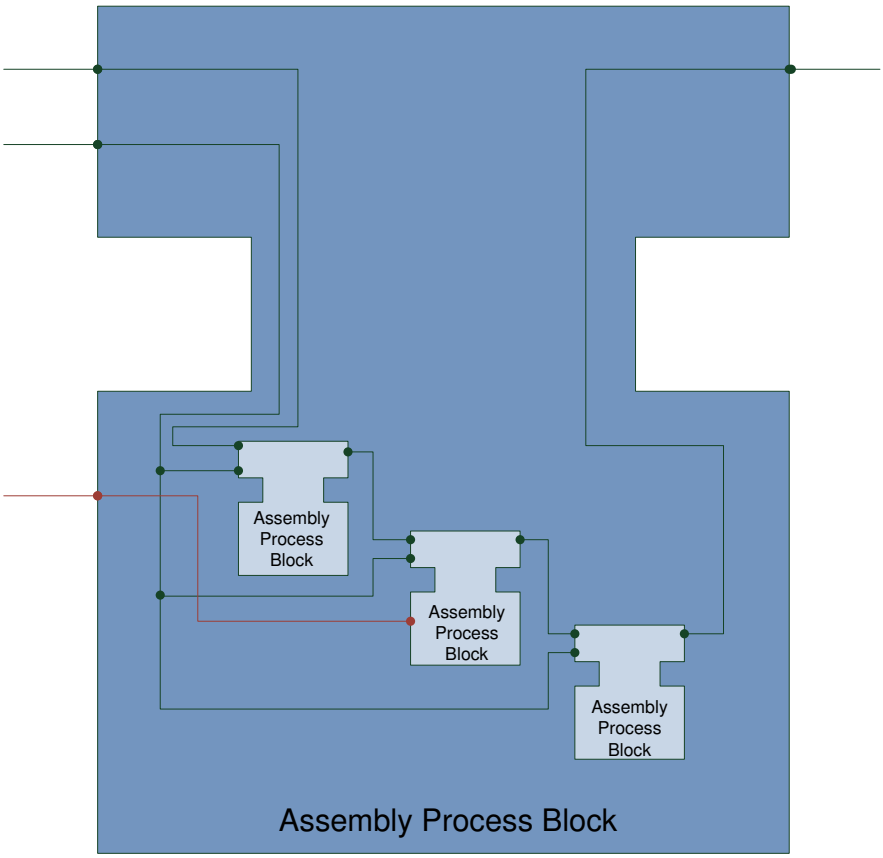


Figure 4.6 - Composite Assembly Process Block Overview

The composite assembly process also need to cater for the connectivity issues that arise from being composed of elementary assembly processes. Therefore, there is a need for establishing how to connect the control and parameter ports. Another important aspect to consider is the possibility of a composite assembly process having alternative realizations. This should be an element that the description should also cater for. The Composed node will contain several composition nodes to deal with the alternatives possible. These will in turn contain assembly process nodes, and a connection node that establishes which port nodes connect to one another. **Figure 4.7** presents that XSD structure that incorporates all the description and enables the definition of composite assembly process blocks.

This provides the model for the definition of templates that can be stored in the Assembly Process library. Once stored these can be retrieved and used by the actors involved in the configuration methodology.

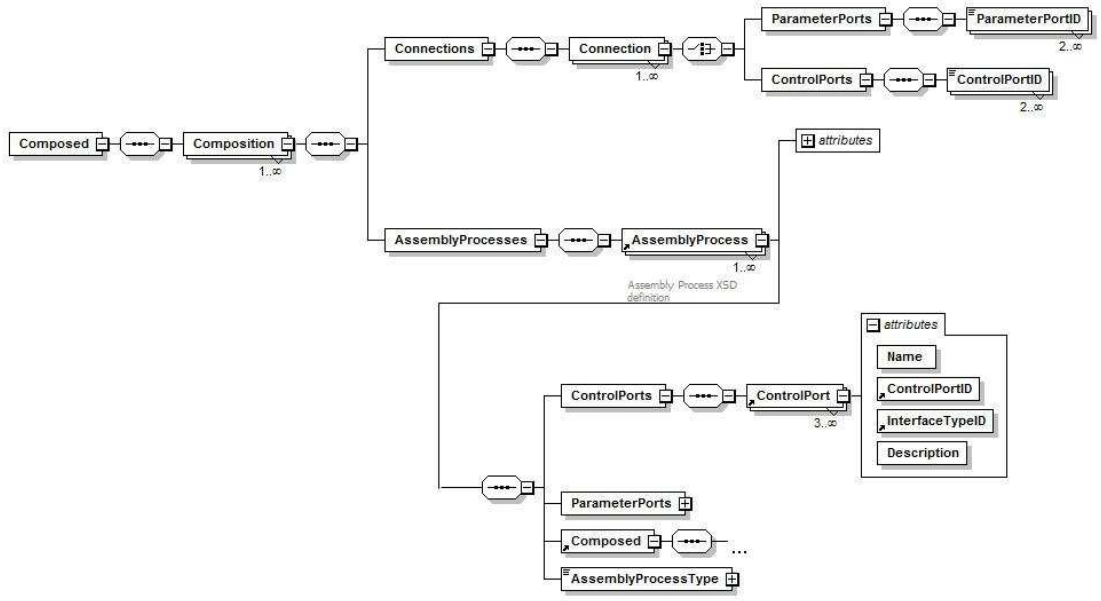


Figure 4.7 - XSD Structure that Enables the Definition of Complex Assembly Processes

4.2.5 Definition of Standard Interfaces Library

An interface is defined as a combination of two ports, either physical or logical, that can be paired together and establish a connection. The definition of interfaces stands at the core of the modular assembly system concept. Interfaces will establish what may be plugged together and what may not. Therefore, the first step is to clarify the origin of the interfaces in MAS, in order to be able to classify them. In MAS there are equipment modules, which will require physical interface descriptions, and functional blocks, which will require logical interfaces.

The assembly process is a functional block with interfaces quite distinguishable from the equipment module interfaces. The assembly process describes a capability, which description includes, as previously mentioned, ports that allow for its operation. The interface definition will consist of a pair of ports of the same type. Therefore, each port is required to have a port type which is extracted from the interface library. In addition, the interface library will have at least two port types for each interface.

The equipment module interfaces follow the same principle as the assembly process ones. These interfaces are defined as the combination of physical ports. So, each physical port will require a type which is established in the interface library with each interface having at least two ports.

The standard interface library will be a repository of defined interfaces. These interfaces will be described in a semantic relevant pre-existing XML structure which allows for its understanding by the self-configuration methodology. An interface is a logical association of two or more assembly process ports which can be plugged together. It is clear that an interface must have at least two ports to fulfil the definition. However, this does only set a lower constraint for the definition. The model allows for the definition of complex interfaces that require the existence of more than two ports. It is envisioned that in some complex cases, both physical and logical interfaces might require this possibility.

The main constraint is that an interface requires all ports to exist in order to be a valid interface. As a result, if there are more than two ports present in the definition of an interface, all must be present when considering a configuration. However, this constrain does not mean that alternative interfaces containing the same type of ports in different numbers cannot exist. On the contrary, it is expected that some port types might be combined to form a lot of different interfaces. Another constraint is that no two interfaces can exist if they contain the same port types. This is a logical constraint to prevent the explosion of duplicate interfaces. **Figure 4.8** provides a conceptual overview of possible interfaces given a set of port types.

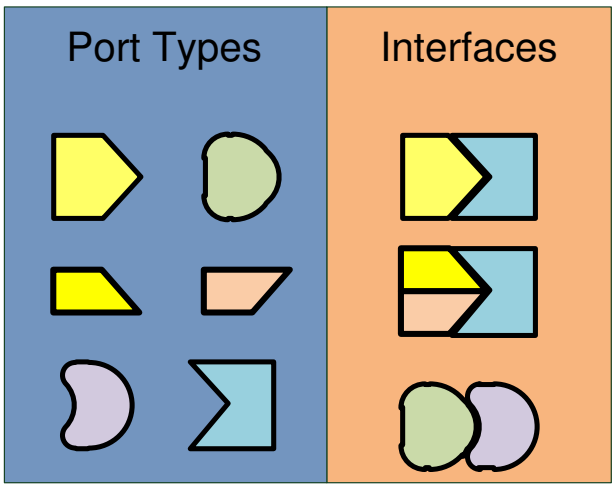


Figure 4.8 - Conceptual Representation of Port Types and Resulting Interfaces

There are two types of interfaces for assembly processes, the interfaces that target control ports and the interfaces that target parameter ports. In both cases the structure of the interface is the same. On the physical side of the modules, there is only one interface type, the equipment interfaces. The equipment interface type is the one that

contains the pair of ports related to physical equipment modules. These are the interfaces responsible for the physical connections between different equipment modules. For completeness of the model they include not only geometry connection but also equipment required interfaces, namely for electric, hydraulic or pneumatic connections. However these aspects will not be catered for in the self-configuration methodology, since for configurations purposes the interface and its port composition is enough to guarantee plugability.

The proposed XSD structure in **Figure 4.9** is composed of the same port types as the interface. The port types need to have their own characteristics, namely name, port type and description. This caters for searches on interfaces that contain a given port type. It is proposed that the interface node contains its own attributes, name, unique ID and Interface Type. The Interface type allows for a faster search within the repository. The proposed types are based on the identified connectivity requirements, which are summarized into three types of interface, namely:

- Control Interfaces – The interfaces that contain only control ports.
- Parameter Interfaces – The interfaces that contain only parameter ports.
- Equipment Interfaces – The interfaces that contain physical ports of the equipment modules creating the pair that enables their connectivity.

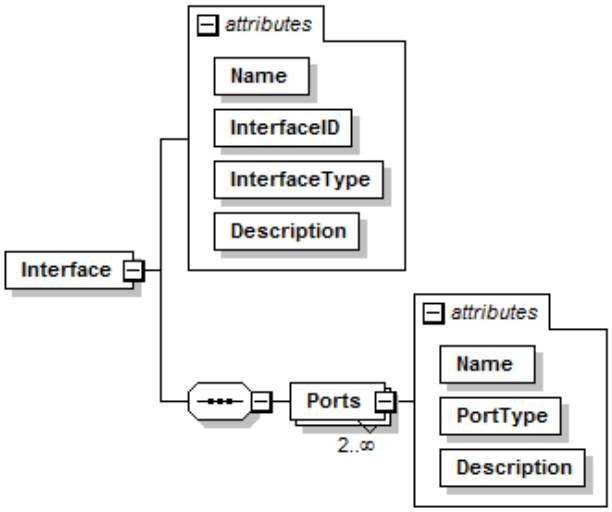


Figure 4.9 - Interface XSD description

4.3 Equipment Module Model Description

The equipment module model defines the relevant module characteristics which need to be captured. The configuration methodology requires an equipment description that is transparent and understandable. The reasoning behind this requirement is the need to identify what information is necessary in order to establish a configuration. This information also needs to be structured in order to become semantically understandable. The XML format is also used here for the equipment module description, thus its structure will be provided through an XSD file.

The definition of the equipment module capabilities is central to enable the matching of available capabilities against required ones. The definition of the module capabilities allows the configuration method to identify those set of modules which have the capability to fulfil the given assembly process requirements. Therefore, this information needs to be contained in the equipment module description. In addition, the assembly processes need to follow the established standard descriptions. For this reason, the equipment module description will simply include a node for capabilities, where you can set a number of specific assembly processes that follow the classifications and terminologies established before in section 4.2.4. The use of overall classifications and terminologies guarantees the ability to compare capabilities between different module vendors, and more importantly to map these to established set of requirements, which is the trigger of the configuration process.

The equipment module physical descriptions required for the configuration are the physical ports. This work focuses solely on the connectivity aspects, thus other physical consideration are not covered here. It follows from this that physical ports are the only physical description required which needs to be in line with the standard interface definitions. For completeness of the model it was included the ability to define reference coordinates for the physical ports. This aspect is not used in the configuration process since the assessment for plugability can be made by checking if a combination of ports has a established interface. The ports also provide characteristics of type, namely:

- Physical Fit – Ports that provide only physical connections with other ports.
- Product Fit – Ports that provide connection for interactions with the product.

These port types are quite important to maintain, since only the physical fit is considered to have a standard interface description. The product fit is an open port that is not considered as an interface in this work. Nevertheless, the model allows for this to change in a future enhancement of the configuration methodology that includes the product descriptions. **Figure 4.10** provides an example where a gripper is attached to a robot to establish a physical fit, while highlighting the importance of the product fit to achieve valid configurations.

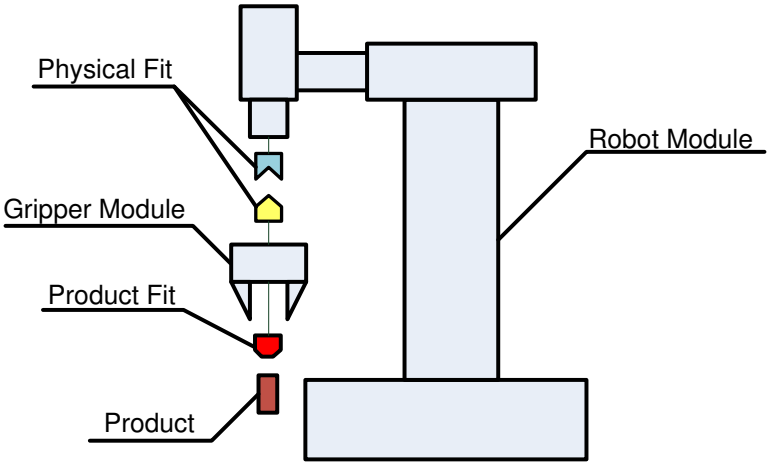


Figure 4.10 - Example of Equipment Connectivity Issues

The XSD for the physical ports is established based on what has been described and can be seen in **Figure 4.11**.

The final required description for the equipment model is business information. The crucial information required for the configuration methodology in this section is cost. The equipment modules description has to contain what the cost of the module is. However, this is not considered to be a straightforward number. The concept of MAS presents two different business solutions for cost, the buying of the modules and leasing of the module (EUPASS [4]; Maffei [130]). This needs to be described in a way that caters for the three possibilities that arise (just buying, just leasing or both). The lease option also requires an extra definition for the availability of the module. This characteristic can also be used for identifying available and unused modules for reconfiguration purposes.

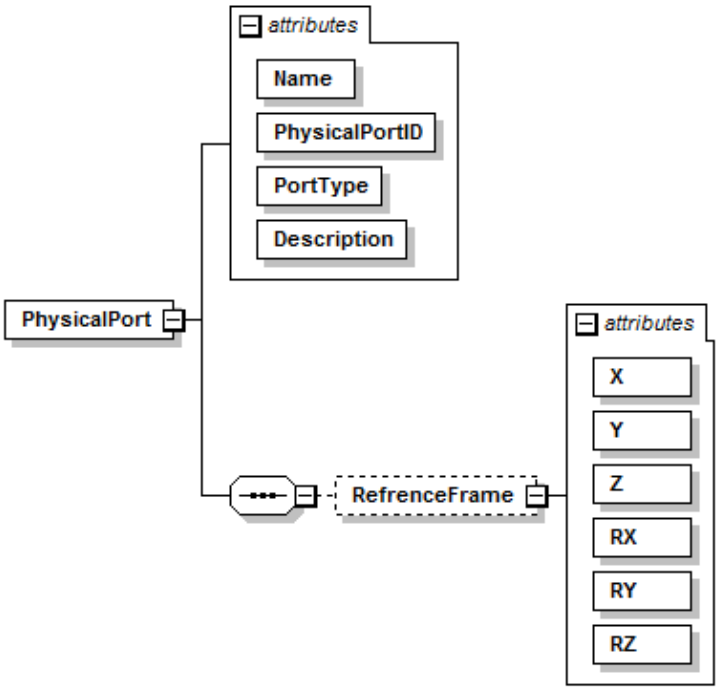


Figure 4.11 - Physical Port XSD Description

The delivery time is an important consideration for cost, especially taking into account the reconfiguration of a system. So, it is established as the average time to deliver the specified module. Another aspect is the preferable collaboration, which contains information to drive equipment modules to interact with preferred modules suppliers. This is introduced because industrialists tend to cooperate with one another within groups, which they want to maintain. Thus, when present, this definition would force configurations to use preferred supplier if the configuration solution is possible using preferred modules. The description also allows for an added value to the collaboration, meaning it is established a percentage to use to discount the cost. This is an important characteristic that is expected to be used when module supplier has several types of modules. The final attribute of the business information establishes the owner of the module. This information is important to determine the source of the module, but also in the case of reconfiguration it allows the method to readjust the values if modules are already present.

The final node within the equipment module is the configurability strategy, which is a simple weight matrix that attributes which configuration indicators are more relevant to the equipment module (**Figure 4.12**). To understand the proposal of this node one needs to look at the previously defined configuration attributes. It is clear

that some modules will be cheaper, others more accurate, others quicker, etc. Therefore it is only logical that module suppliers want to influence an automatic configuration by establishing priorities among these attributes in order to increase the chances of participating in a system configuration. This way, the module suppliers have the ability to influence the configuration strategy of the equipment modules playing to their strengths. It is clear that a cheap module might want to put emphasis on this aspect to collaborate with other cheap modules. On the other hand a very precise module might want to value this aspect more. The idea here is to provide the module suppliers with the means to influence how their module will try to fulfil its objective of being selected into a MAS configuration. The total value of the sum of all the elements of this matrix is always one.

$$\begin{bmatrix} Percentage\ for\ Cost \\ Percentage\ for\ Time \\ Percentage\ for\ Flexibility \\ Percentage\ for\ Accuracy \\ Percentage\ for\ Repeatability \end{bmatrix}$$

Figure 4.12 - Weight Matrix for the Configuration Attributes of the Equipment Module

In summary the equipment module XSD contains all the described characteristics, divided into four main nodes, namely Module Capabilities, Module Structure, Business Information and Configuration Strategy. An overview of the equipment module XSD is provided in **Figure 4.13**.

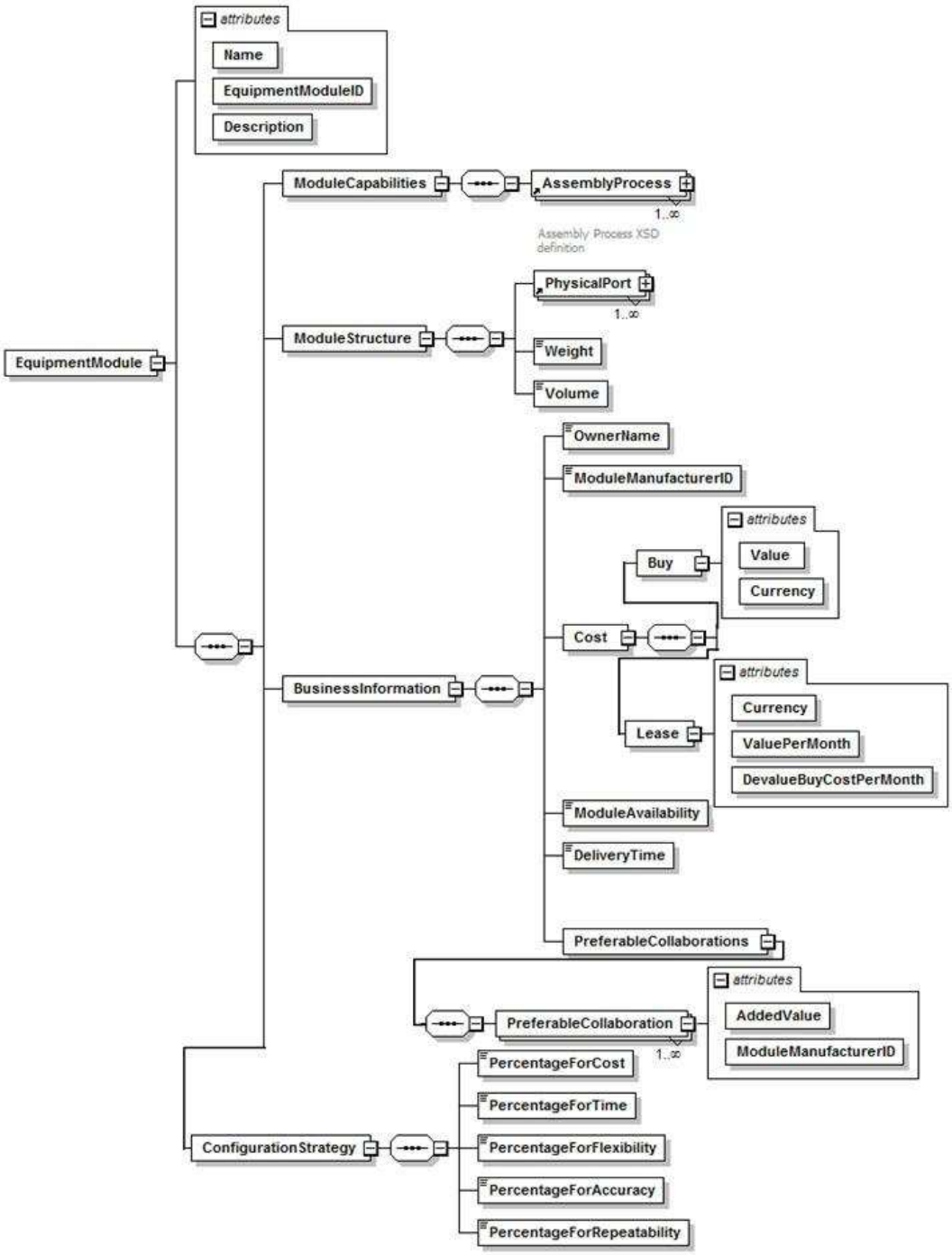


Figure 4.13 - Equipment Module XSD Overview

4.4 Assembly System Requirements

The assembly system requirements define the expectations for the assembly system. It is important to differentiate these expectations into two groups, the required capabilities and constraints of the system, and the assembly system targets that will drive the configuration process. Based on literature the main targets that the

configuration methodology will use for establishing valid configuration are Cost, Time, Flexibility and Quality (Chryssolouris [14]).

Generally speaking, the definition of the assembly system requirements is composed of three parts. The first is the definition of the values and targets for the wide system. This will contain the information for system validation, as well as the optimization information. The second is the physical requirements for the system. This will provide all the system information, namely constraints on the workstations and existing equipments. The third and final part is the assembly processes that the system is required to perform. This defines the required capabilities of the system in terms of assembly processes.

The XSD that provides this structure will contain the three main nodes, plus some attributes, namely Name, unique ID and description. **Figure 4.14** provides the overview for the assembly system requirements, which will be detailed in the following sub-chapters.

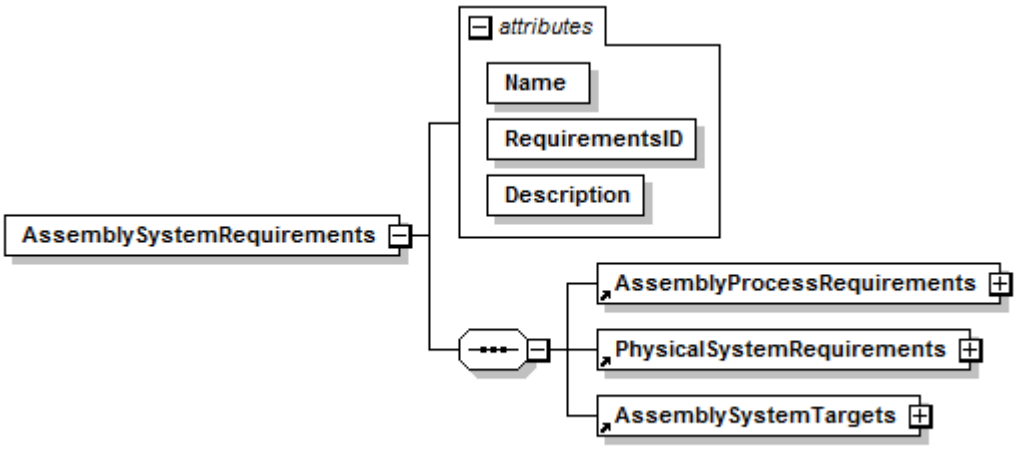


Figure 4.14 - Assembly System Requirements XSD Overview

4.2.6 Assembly System Targets

The assembly system targets node contains all the overall targets for the system. In this node the overall information of the system is defined. This information provides the basis for the assessment of valid solutions, namely the values set by the different targets. The XSD diagram **Figure 4.15** shows how the information is structured and arranged for the purpose of defining the assembly system requirements.

The system cost has been discussed previously, and it was stated that the two options of assembly system are buying a system or leasing a system. This is the first decision that the system integrator needs to supply for the configuration methodology to operate. The cost information on this node will define the configuration methodology targets. Because cost is an indicator that is expected to be minimized, any solutions that go above the target will be discarded. Therefore, the cost establishes a maximum threshold for the system cost.

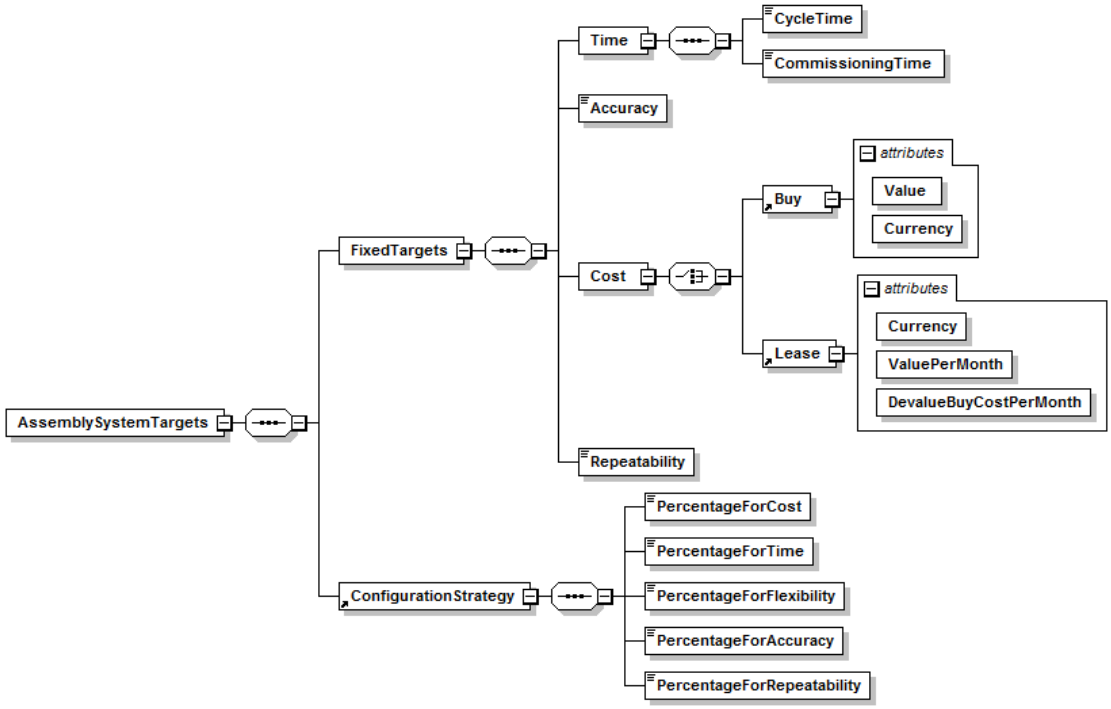


Figure 4.15 - Assembly System Targets XSD Overview

Time is another central block for the operation of the configuration methodology. Time is divided into two categories, commissioning time and cycle time. The commissioning time simply states by when the system needs to be available. This information is important since some equipment modules might be available in the near future, but not at present. The cycle time is defined as the time required for running the required processes one time. It is important to note that the possibility of defining an overall cycle time does not prevent the system integrator from defining specific cycle times for individual required assembly processes. However, it is not expected that all required assembly processes will contain this information and thus, the overall cycle time is crucial for establishing a target for the configuration methodology. In summary, time information provides two crucial requirements, one

for the commissioning time and the other for the overall cycle time. The cycle time and commissioning time are similar to cost, so they establish a maximum threshold number that cannot be broken.

The quality indicator, or as defined previously the accuracy and repeatability of a workstation, is related to the definition of the required assembly processes, and will be described in the subsequent chapters. Nevertheless, similarly to the time indicator, it is expected that an overall system accuracy and repeatability needs to be present in the overall system description. The overall system accuracy and repeatability are indicators that are defined based on the allowed difference between the system targets and the system performance. These indicators are at their best when their value is zero, so similarly to the other indicators these also establish a maximum threshold.

Flexibility has been defined as the spare capacity of the system and it does not need to be defined in the assembly system requirements as a value. This characteristic can be inferred based on its definition. Extra capabilities are viewed as positive if it has a minimum impact on other characteristics.

In the assembly targets node there is a weight matrix that defines the importance of each of the indicator for the given required assembly system. These indicators will be used for the assessment of the different configurations, providing the most suitable solutions for the required assembly system. These are the same as for the equipment module descriptions seen in **Figure 4.12**.

4.2.7 Physical System Requirements

The physical system requirements are designed to provide physical guidelines for the configuration. The configuration methodology requires the definition of workstations to predetermine the workspace for the system. Therefore there is the need to pre-establish the number of workstations. The requirements catering for such definition also allow for establishing pre-existing modules for the configuration. This is viewed to be the case in reconfiguration scenarios, where a full system can be described minus the missing capabilities.

The proposed physical system requirements model is comprised of information for both configuration but also reconfiguration. It is expected that the definitions for configuration can be quite high level descriptions, however, the system reconfiguration needs to contain the existing configuration, which needs to provide specific equipment module information. Therefore, the model needs to provide a structure that enables definitions at all levels.

It was stated above that the model follows a hierarchical structure, from system to equipment module. Systems contain workstations and workstations contain equipment modules. Each of these is considered to be a main node, which can contain several instances of its subsequent node. For each of these nodes have their own characteristics, namely their description, unique ID and name.

The second node will contain the port description, establishing the connectivity possibilities for each node level. These ports are mere lists of what the expected plugability options are. These options will be used in the third node which contains the connections. The port descriptions will have a name, a unique ID, an interface type and a description. The connections node establishes the expected connections using the established ports. This node will contain several connection nodes, which in turn contain which ports plug to each other. This enables the possibility to establish constraints for the configuration solution, if the system integrator requires certain modules to be connected in a certain way.

It is important to note that the established connections in some cases are across different levels, i.e. between system and workstation or workstation and equipment module. **Figure 4.16** provides a conceptual example of this, by describing the conceptual system level with three workstations. One of the workstations is pre-existing, and therefore has specified its equipment modules. This example is missing the necessary feeding module, which means the configuration methodology will be required to complete this workstation. It is also important to note that defining equipment modules is optional. The example also allows for a better understanding of how the outer ports of each block needs to be connected to its inner blocks to describe how the system looks. Finally, another important remark is that the described ports are only those that have an interface, meaning that the product related ports are not present in this diagram.

The physical constraints for the configuration and reconfiguration method are provided in the Requirement Level attribute in each level. This attribute determines if the node is optional, mandatory or advised. This is mostly intended for the equipment module level but it is valid for the other levels. In the equipment module level it allows for forcing specific modules, advising then, or simply inform of availability.

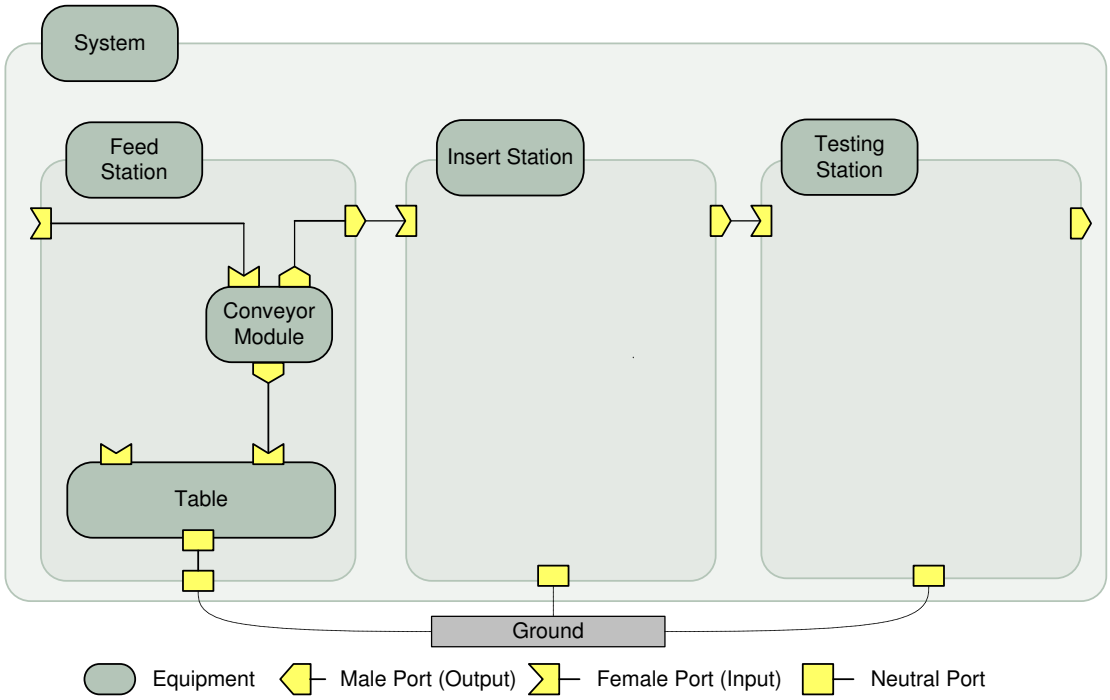


Figure 4.16 - Example of Conceptual Physical System Requirements

The last node, the spare equipment modules, provides a list of available equipment modules which targets the system reconfiguration without any constraint on where to place them. The idea is this is a list of available modules in a given site, which will obviously have low cost since they are not required to be bought. This will provide them with an incentive for use whenever possible in the configuration methodology.

Figure 4.17 provides the discussed aspects modelled into a XSD file overview of the model.

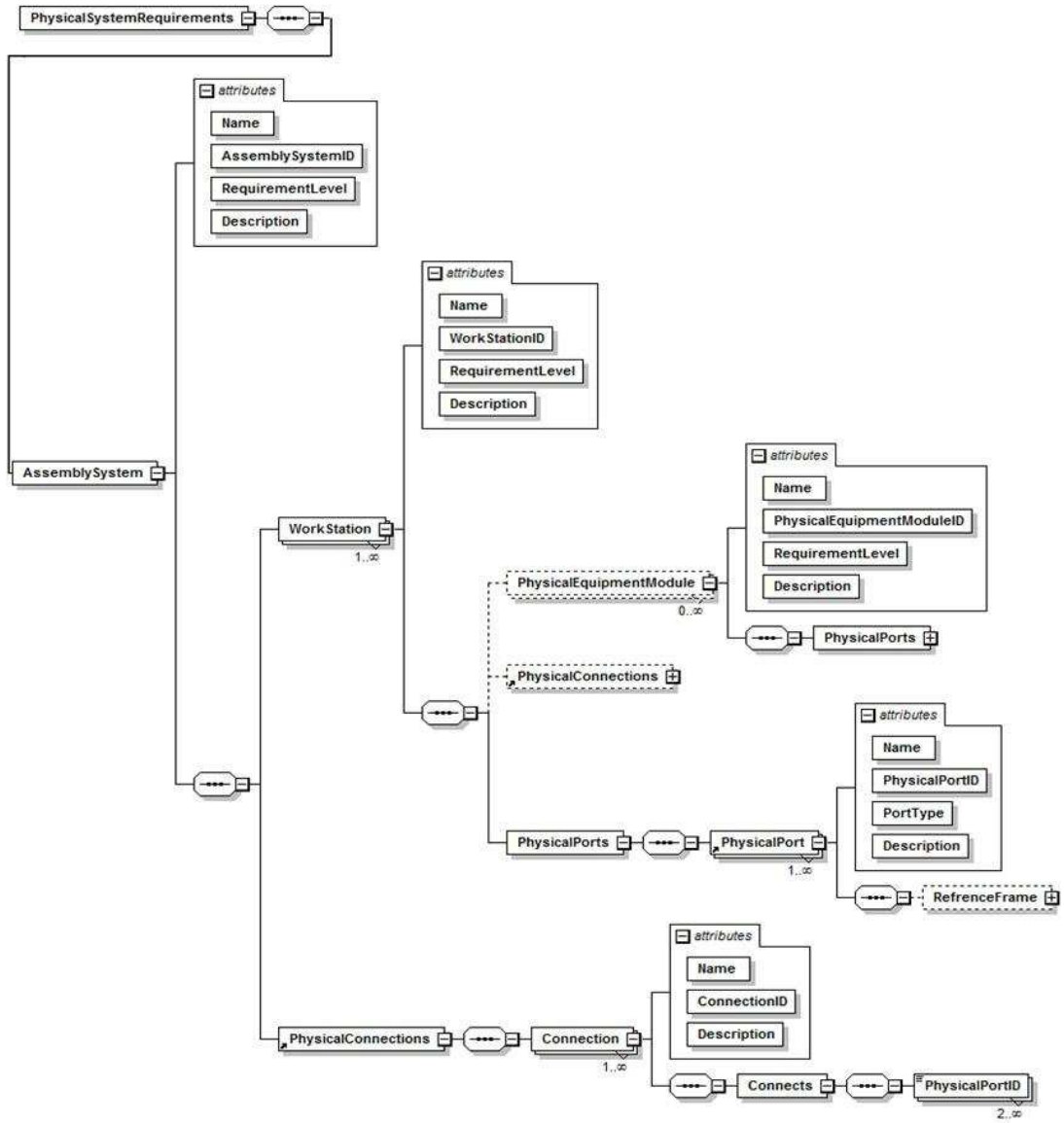


Figure 4.17 - Physical System Requirements XSD overview

4.2.8 Assembly Process (Skills) Requirements

The assembly process requirements are the representation of the system desired capabilities. The assembly processes should reflect the needs for the assembly of a given product, or set of products. This work does not focus on the product side, therefore it is assumed that all the relevant product information is included in the assembly process requirements.

The assembly process requirements provide two sets of high level information for the configuration methodology. The first is the required system capabilities which are

required for the assembly of a product and which need to be present in the assembly system. The second is the assembly processes connections, which defines the sequence of the required assembly processes. The overall XSD model view of the assembly process requirements can be seen in **Figure 4.18**.

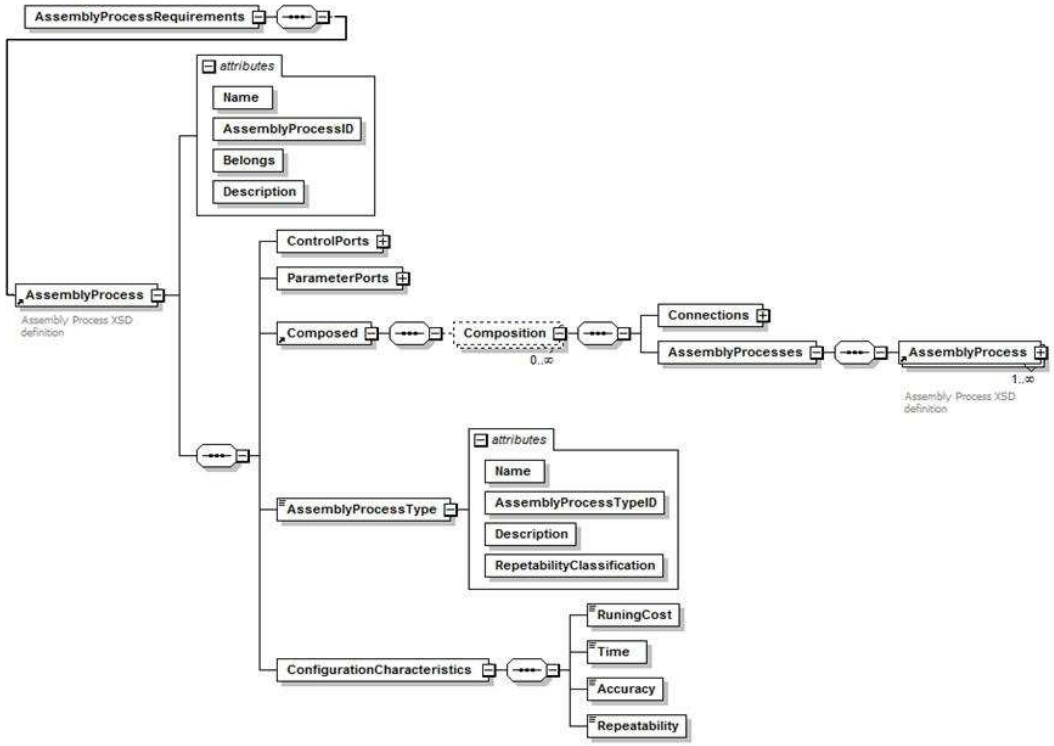


Figure 4.18 - Assembly Process Requirements XSD Overview

The required system capabilities need to be matched with the capabilities of the equipment modules, therefore these have to be instances from assembly process definitions from the assembly processes library. Therefore, this definition follows the same terminology as the one used to describe the equipment modules. The control ports and parameter ports need also to be instantiated to provide the inputs and outputs for the connection nodes. These ports in turn allow for the definition of the assembly processes sequence in the connections node. It is important to note that these port definition are expected to be the bare minimum, however the model caters for a full definition for completeness.

The assembly process requirements allow for the definition of composite assembly processes that are composed of elementary assembly processes, allowing for the encapsulation of complex assembly processes in the requirement. This is viewed as

important for the establishment of constraints on what is the composition of high level assembly processes. Toward that end the "Composed" node is proposed. This node consists of two other nodes, the assembly processes and the connections. The assembly processes node describes the composition of the assembly process by defining its containing assembly processes. The connections node establishes connections between the control ports of these assembly processes. This follows the same concept that was presented in **Figure 4.6**, where the need for a clear establishment of connections is crucial for the concept to work. Without the connections, it would be impossible to have complex assembly processes and also no process sequence. It is important to note that in all requirement definition there exists a high level assembly process that contains all other assembly processes.

Finally, this model needs to provide a relation to the physical system requirements to establish where the assembly processes are expected to be performed, so the "Belongs" attribute is introduced. This simply provides a relation through unique ID with the system, workstation or equipment module responsible for the assembly process. **Figure 4.19** shows a conceptual example of this relation.

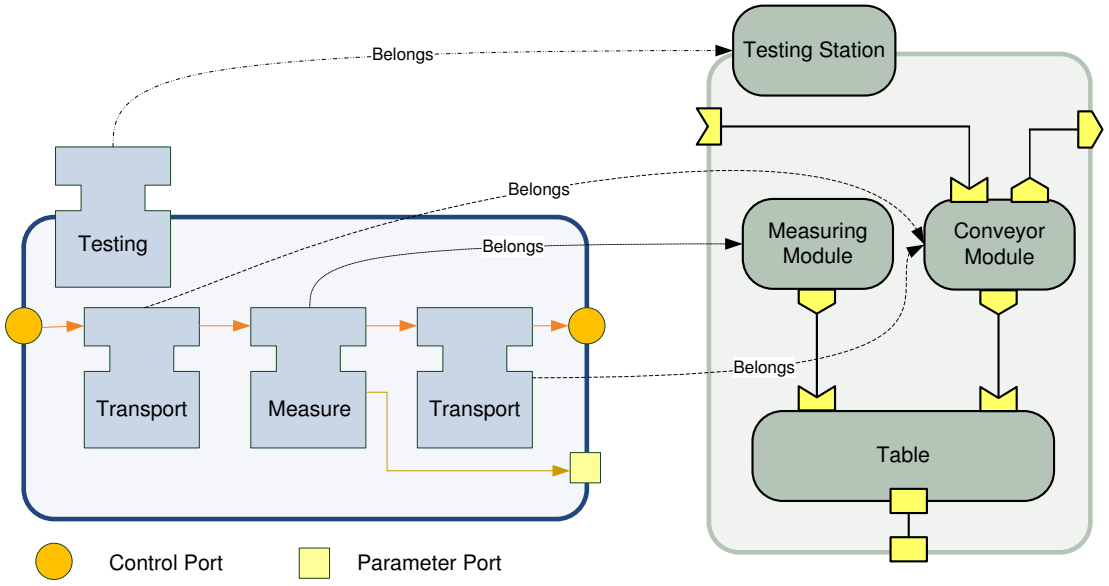


Figure 4.19 - Conceptual Example of Assembly Process and System Relations

4.5 Assembly System Configuration Output

The final piece of the model definition is driven by the need to present the results of the self-configuration methodology. However, there is no need for a complex definition of the output of the methodology since the established structures already provide all the necessary structuring for defining a configuration file.

The system requirements model presented in Section 4.4 provides the perfect base structure for this. As it was discussed in the previous chapter the structure has three main elements, the physical assembly requirements, the assembly process requirements and the assembly system targets. The physical assembly requirements allows for the definition of the system to the equipment module level, which in the requirements is optional. Nevertheless, the structure is there to define a complete physical configuration. In the case of the output configuration description this node would be the same but called physical assembly. Similarity, the assembly process requirements also allows for the definition of complete assembly process sequences, thus the use of the same structure is only logical. Again the node is renamed to assembly process for clarity of the model. Finally the assembly system targets define the information that enables the choice of assembly system configuration. Thus the same values will need to be provided for the assessment process, therefore the same structure can be used.

In sum, the assembly system requirements model is more than suitable for the definition of a configuration output file with the mentioned minor changes, since it provides the base structure for the definition of the whole system.

4.6 Chapter Summary

The focus of this chapter was on the formal definition of the model developed to enable the self-configuration methodology. The proposed model enables the description of information to be introduced in the proposed self-configuration methodology. Moreover, a common terminology is presented and a structure to maintain the commonalities across all descriptions is proposed. A taxonomy was also identified for MAS which is suitable for the configuration methodology, and it was proposed some enhancements to cater for the assessment of MAS quality

characteristics. An equipment module definition and structure is also proposed as well. Finally a clear assembly system requirements definition and the output of the methodology are presented.

This chapter also establishes some of the assumptions for the use of this model by the configuration methodology, such as the disregard of the product for the system configuration, the need for the use of a global terminology, the required inputs of the configuration methodology and their structure and the definitions of the used terminology. Finally, it offers the means for introducing information into the configuration methodology, namely through the creation of XML files that follow the described specifications.

5 Agent Architecture for Distributed Self-Configuration Methodology for Modular Assembly Systems

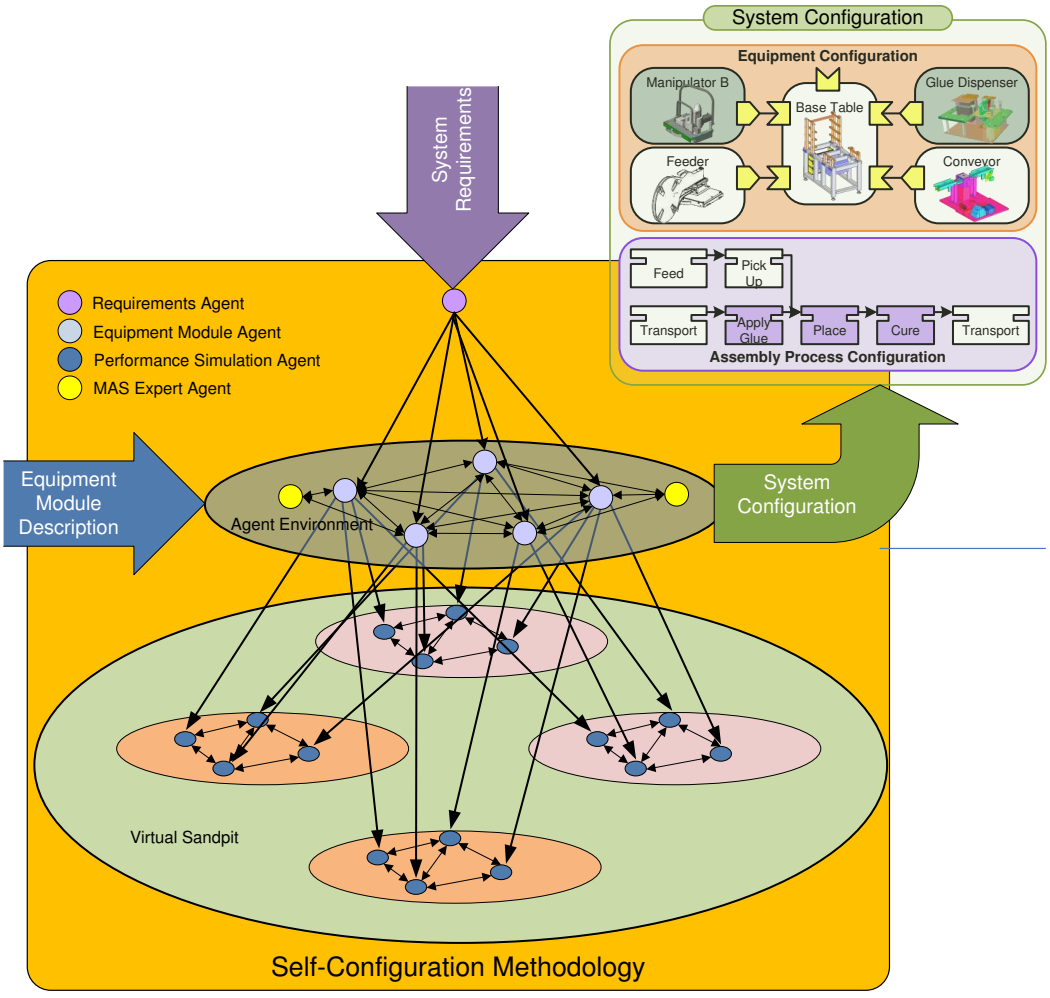


Figure 5.1 - Overview of Self-Configuration Methodology Solution

5.1 Introduction

In this chapter, a new multi agent architecture for the self-configuration of Modular Assembly Systems (MAS) will be presented. The chapter describes the followed approach to create the agent architecture, while also providing the detailed descriptions and models of the proposed architecture. The proposed agent architecture will provide the basis for the creation of an agent environment that reflects the concepts of MAS. This enables the distributed decision making necessary to enable the bottom up approach to establish an automatic distributed self-configuration methodology.

The nature of the MAS paradigm with its focus on clear functional decoupling of equipment module functionalities and standardised interfaces for interchange ability has opened the scope for automatic configuration methods (EUPASS [4]). It becomes possible to clearly formalise the functional capabilities and connectivity constraints of the available modules, hence allow the mapping of process requirements against available capabilities to synthesise suitable assembly system configurations. The design of MAS is therefore essentially a conjoint equipment and assembly process configuration problem at several levels of granularity with equipment modules and their functional capabilities (assembly processes) as the elementary building blocks.

The number of possible combinations of modules required for an assembly system solution depends on the number of available modules, their connection constraints and the complexity of the given assembly process requirements. The combinations, even for relatively small problems, become quite large making configurations based on exhaustive enumeration practically infeasible. For this reason, an appropriate MAS configuration methodology needs to more goal-oriented. Furthermore, any method should be able to exploit the specificities of the MAS configuration problem to reduce the search space. Most MAS solutions described in literature exhibit a very low level of complexity. This can be split into a number of loosely coupled sub-problems with corresponding solutions making them hierarchical in nature. Additionally, elementary equipment modules often have specific predefined roles within a solution and can be classified accordingly, reducing the possible number of candidates for specific aspect of a configuration. This chapter will reflect on all these

aspects in order to provide a coherent agent architecture which serves as the basis for the self-configuration methodology.

While heuristic search and linear programming methods are able to solve these kinds of configuration problems, they require quite complex models and are difficult to maintain. These solutions are also very specific and non scalable to a domain that is constantly evolving. Furthermore, they apply a top down approach which only takes limited to no advantage of the hierarchical nature of the problem. Therefore, this work proposes a multi agent solution for the bottom up solving of this configuration problem maximising the parallel computation and taking advantage of latest negotiation protocols to achieve a goal oriented behaviour of the overall configuration environment.

5.2 Agent Architecture Requirements and Objectives

The design of multi agent environments is a problem extensively covered in the literature, covering a wide range of problem domains (Shen et al. [1]). The variety of problem domains poses a challenge for a widely acceptable methodology for the definition of such systems. Nevertheless, several methodologies have been presented each one arguing their strengths. After a literature review the GAIA methodology (Zambonelli et al. [123]) was identified as the most promising for the configuration problem because it provides guidelines for the system definition which are flexible and provides enough leeway for heuristic inputs in the definitions.

The GAIA methodology identifies the clear definition of the system requirements as the first step for a definition of a multi agent system. In the previous chapter the MAS requirements and the equipment module description were identified and modelled as the inputs for a self-configuration methodology. These established inputs allow for a clear mapping between what exists in the equipment module repository and the requirements for the new systems. In addition, the definition of the requirements as defined in **Chapter 4**, caters for both configuration and reconfiguration problems. **Figure 5.1** clearly shows the requirements needed as inputs, namely in the form of XML files, which have been described in detail in the previous chapter.

The following steps of the GAIA methodology are analysis, architectural design, detailed design and implementation. The analysis of the inputs was presented in the previous chapter, while the problem analysis will be performed across this chapter.

5.3 Agent Architecture

The design of the agent architecture based on the GAIA methodology first requires an analysis of the problem, namely the clear definition of the objectives and targets that the agent system has to address. The first step is the understanding of the requirements for such system, namely the identification of what needs to happen and what information is required, which has been done in the previous chapter. Therefore, the established requirements are already influenced by the objectives of modular assembly system configuration and reconfiguration. The main objective of the system is to provide valid solutions for the configuration and reconfiguration of MAS. What this work proposes is the creation of an agent environment that through collaboration between agents is able to achieve this objective.

The achievement of valid solutions, e.g. logical and plausible configuration combinations, is the first objective. Its achievement alone however would not grant a big step forward compared to current state of the art solutions. Valid solutions can be achieved with the current manual configuration tools. However, these solutions tend to use only a restrict set of modules, which might not be the best set, but a valid set. Also, the modular concept is expected to produce an increasing number of modules in the future, making the manual configuration untreatable if all modules should be considered. Therefore, the need for an automatic solution that also targets the best system configuration is critical.

As the need for a solution that targets the best system configuration has been identified, it is crucial to define what constitutes the best solution. In the configuration of MAS the best solution will vary depending on the individual performance objectives for a system. In fact, it would be impossible to determine the best. Nevertheless, what can be defined is a best solution that is based on globally accepted performance characteristics for assembly systems. These have been identified and described in the previous chapter.

The existence of performance characteristics does not establish which are more relevant, and which are less relevant. In fact, the value of these parameters varies depending on the person providing an estimate. This is the reason for the non existence of an absolute best configuration. Therefore, it is proposed that the different actors involved in the system configuration can provide inputs for establishing what qualifies as the best configuration. This is catered for in the model provided in the previous chapter and takes the form of a weight matrix.

The ultimate decision for choice of the best configuration will always reside with the system integrator. What the proposed system will provide is a list of the top configurations based on pre-established weights for the performance characteristics. The system integrator defines these weights, but has the freedom to choose from any of the available solutions. This is viewed as an important aspect of the self-configuration methodology to ensure the transparency of the method for their users.

The module suppliers are the other user that might have some expert knowledge on which performance characteristics should be more valued for their specific module. The idea is to capture their past experience in what characteristics made them more successful (sold more equipment modules). As such, each equipment module description contains a weight matrix that ultimately will determine the configuration decision. It is foreseen that in the future these can be readjusted based on prior failed or successful configurations. The detailed strategies for the use of these aspects will be contained in the agent methods in **Chapter 6**.

In this chapter the agent model will be described, as well as the detailed agent definitions, the organizational model of the agent architecture and the definition of the agent interactions, which is a crucial step for the achievement of a distributed decision making solution.

5.3.1 Agent Model Overview

The main common denominator of all configuration design methodologies including MAS is the need to elicit and maintain the system requirements independent of the proposed solution alternatives (Ferreira et al. [131]). Consequently there is a need for **Requirements Agent** which is able to provide clear objectives to those agents involved in the configuration process. Furthermore, they need to be able to represent

the interests of the customer/system user to validate possible system configurations against the original requirements. The need for assessment capabilities in this agent is justified by the possibility of a big list of configuration solutions, which will be filtered and ranked by this agent based.

Another important aspect within this problem domain is the equipment modules. It is proposed that each equipment module should be represented by **Equipment Module Agents** that have a detailed understanding of the module's capabilities and behaviour, which is viewed as crucial to enable the concept of self-configuration.

The **Equipment Module Agents** have to play a key role in the bottom up approach to the MAS configuration process, because they are representations of the equipment modules enhanced with methods to enable the collaboration with other agents to achieve MAS configurations. Each **Equipment Module Agent** should only be aware of its own capabilities and only should have a very limited understanding of the surrounding world to maximise the adaptability and scalability of the framework. This provides the ability to introduce new modules, or remove them without the need to adjust the self-configuration methodology. Consequently, there needs to be a mechanism which validates the logical consistency of the agreed interactions between collaborating **Equipment Module Agents**. An agent system is in itself a modular system where new agents can be introduced as long as they adhere to the agent architecture rules. This characteristic allows the creation of interactions with expert agents that can be enhanced in the future to cater for the changes in the constantly evolving domain of MAS (Onori and Oliveira [18]).

The role of a mediator has been introduced to the agent architecture to create a configuration assessment mechanism. This role will be fulfilled by an **MAS Expert Agent** which is responsible for assessing the logical conditions of the configurations based on its internal knowledge model. The need for the introduction of these concepts into a separate agent is supported by the nature of this knowledge, which is in constant evolution. If this knowledge was built in to the configuration methods, e.g. the internal decision making models of **Equipment Module Agents**, future changes might require a complete change of the configuration methodology. Therefore, it is proposed that this knowledge is decoupled into the **MAS Expert**

Agent, which can be changed or even replaced in order to allow the evolution of this knowledge.

The **MAS Expert Agent** is expected to change, however its interactions in the agent architecture will not. This strengthens the need of clear communication and iteration protocols which are required to work in the future iterations of this agent. The general premise to establish an interaction is its need, thus the question is which agent or agents require this expert knowledge. At first glance both **Requirements Agents** and **Equipment Module Agents** can benefit from access, but a closer look shows the redundancy of establishing interactions for both. The requirements could be extended by the **MAS Expert Agent** to contain extra constraints for the MAS configuration. However, this would not include equipment module specific constraints, therefore the **Equipment Module Agents** will require an additional interaction with the **MAS Expert Agent**, regardless of prior interactions with the **Requirements Agent**. On the other hand if the **MAS Expert Agent** only interacts with the **Equipment Module Agents**, the extension of the requirements will occur as part of the configuration assessment. Therefore it is proposed that the **MAS Expert Agent** only interact with the **Equipment Module Agents** for validating configuration that fulfil the requirements as they were set.

This architecture will potentially provide a very large number of possible solutions. Some method for early evaluation of the likely success a consortium needs to be available to reduce the computation effort required. Ideally, this evaluation should be synthesised from the actual performance characteristics of the modules. To provide some bases for early comparison, it is proposed that the **Equipment Module Agents** deploys **Performance Simulation Agent** a simulation environment. These agents represent the physical and process capabilities of the modules and dynamically interact with each other to determine the resulting behaviour and performance characteristics of a consortium. The information provided by these agents is used for the final decision of which configuration is better. The decoupling the simulation process from the **Equipment Module Agents** is justified by the computer intensive task that configuration methodology already is. Furthermore this decoupling provides the means for equipment suppliers to supply this computational effort as a service, which reduces the computational requirements on their clients, the system integrators. In addiction it is envisioned that each **Equipment Module Agents** will

run multiple simulations, therefore the use of distributed computing is seen as the best option. This vision was supported and generally accepted within the EUPASS consortium (EUPASS [4]).

The organizational structure for these agents is based on the agent-roles discussed above. The **Requirements Agent** is hierarchically above the **Equipment Module Agent**, since it triggers the beginning of the collaboration process and it terminates it by making a selection.

The **Performance Simulation Agent** is hierarchically below the **Equipment Module Agent**, since this agent is the only one with the information required to deploy it. The **MAS Expert Agent** is on higher level from the **Equipment Module Agent** since it can provide a global view of the configurations. An overview of this can be seen in **Figure 5.2**.

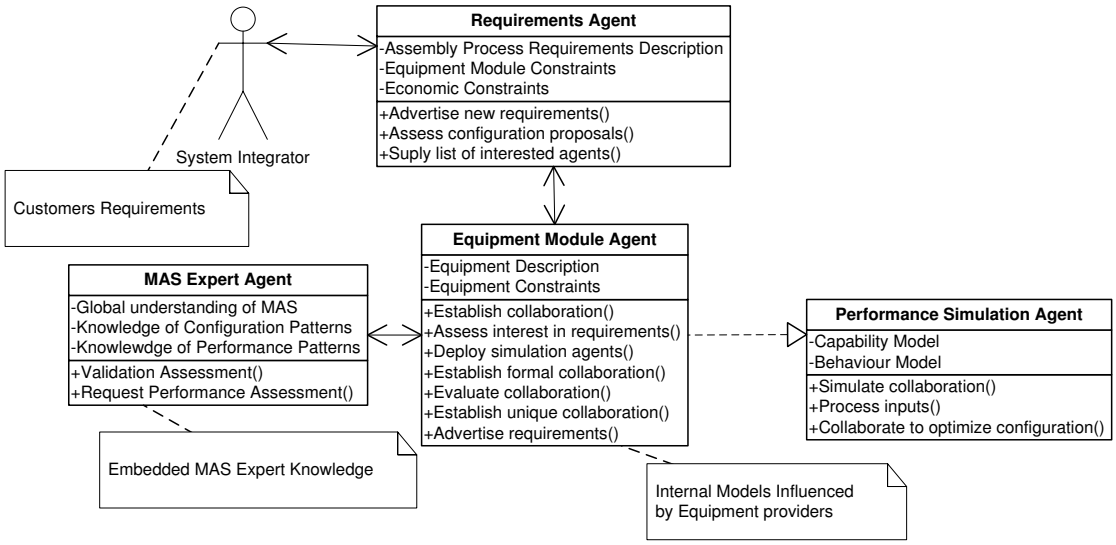


Figure 5.2 - Agent Architecture Class Overview

The existence of different agent types emphasises the need for clear separation of levels. In the proposed agent model there are three clear tiers. The first tier is the triggering configuration tier where the **Requirements Agent** sits. The second tier is collaboration establishment tier, where the MAS configurations are established. This tier is populated by **Equipment Module Agents** and **MAS Expert Agents**. The final tier is the virtual sandpit tier, where the **Performance Simulation Agent** are deployed to assess performance characteristic of given configurations. The proposed three tier structure requires some decoupling of agent roles, particularly across tiers, for a clear architecture design.

All agents have to be able to communicate, which is an intrinsic characteristic of agent technology. However, the agents will disregard communications that are not modelled in this chapter.

5.3.2 Agent Organizational Model

The proposed multi agent environment should be seen in three tiers which provide a clearer picture of what occurs during the configuration process. In addition to this clarity this three tier structure also reflects the existing hierarchies between the different actors. **Figure 5.3** provides an overview model for the agent environment.

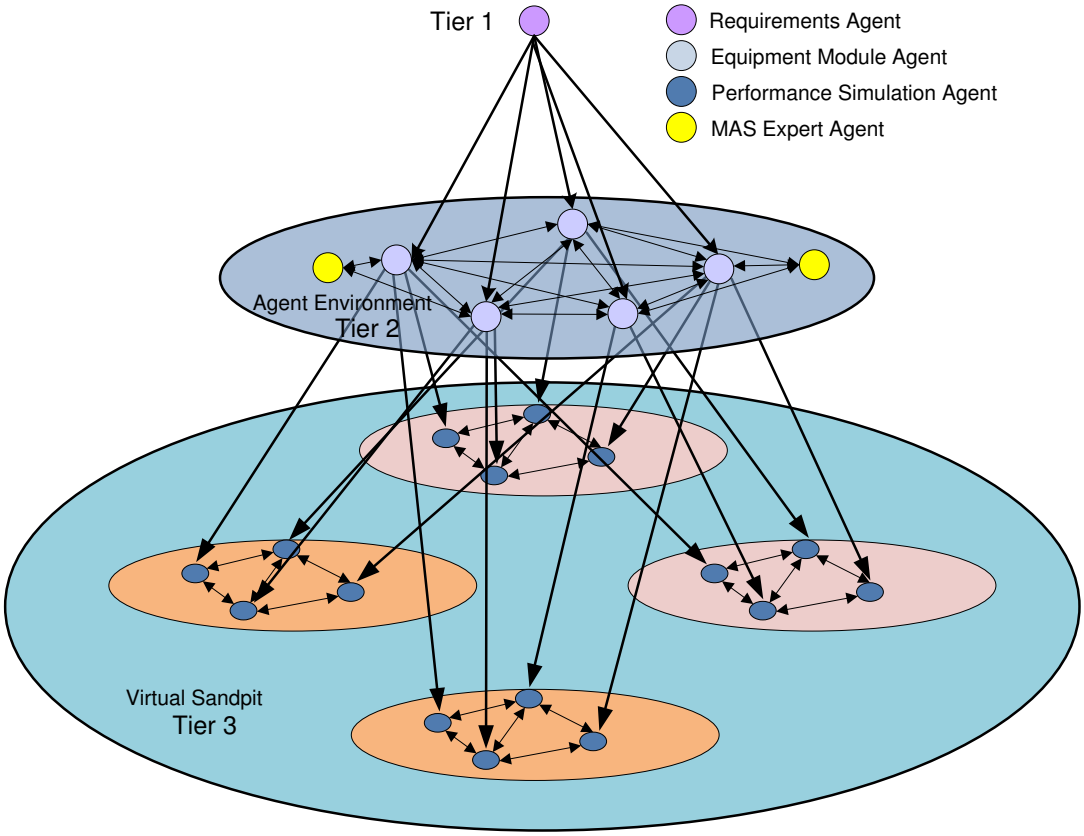


Figure 5.3 – Overview Model of Agent Environment

The first tier represents the separation between the MAS requirements established by the system integrator and the equipment modules agents (representations of the equipment modules) that will drive the configuration process. The communication from the first tier provides the configuration requirements, which are the trigger of the configuration methodology. As such this tier only contains one agent, the **Requirements Agent**.

The second tier is created to cater for the decision making processes involved in the configuration methodology. In this tier **Equipment Module Agents** react to advertised requirements and try to form collaborations that represent valid configuration that fulfil the requirements. The interaction with **MAS Expert Agent** occurs also here and is viewed as added value to establish better configurations.

The final tier is created to have clear separation of simulations and the decision making process for establishing configuration solutions. Simulations are quite computer intensive, therefore if included in the **Equipment Module Agents** the configuration methodology would be quite slow. Thus it is proposed the existence of **Performance Simulation Agents** that are able to be distributed across different machines in order to distribute the most computer intense process. This is viewed as a way to shorter decision times, since the results will appear faster.

5.3.3 Requirements Agent Definition

Requirements Agents are responsible for eliciting the assembly system requirements from a system integrator and advertising them in an understandable format to the other **Equipment Module Agents**. These agents will provide the assembly tasks description to the interested agents, which entail the basic requirements for the system.

The **Requirements Agent** also has to have the capability to assess the established collaborations in order to select the most suitable assembly system. This selection is based on the relevant assembly system aspects, namely cost, time, accuracy, repeatability and flexibility.

Finally, this agent can also negotiate with the system integrator some trade-offs between the established systems constraints and requirements.

5.3.3.1. Agent Role and Use Cases

The **Requirements Agent** sits at the first tier and stands alone, therefore a clear separation between this agent and the other agents must be set. Firstly, it should be clear that the roles that this agent performs are unique to this tier, since other agents have no access to the system integrator, or can select the final configuration. This agent is placed as a buffer between the system integrator, through the definition of

configuration requirements, and the configuration methodology which sits in the lower tiers.

The role of the **Requirements Agent** needs to reflect all these considerations, but also enable the operation of the configuration methodology. The first role of the **Requirements Agent** is to trigger the start of the configuration process. This will require the definition of a few functionalities, which will be detailed later in this work, but the general concept is the elicitation of configuration requirements and the broadcast of the requirements to the second tier agents, namely the **Equipment Module Agents**.

The second role of the **Requirements Agent** is the communication with the system integrator. This role is defined to provide methods that enable a two way communication with the system integrator, either to inform him on the current status of the configuration method, or to request changes to the requirements when solutions are not possible given the existing equipment. This role enables the system integrator to track the configuration method, and change requirements if needed. To ensure this ability the **Requirements Agent** has to provide the means for a constant update of configuration solutions by the **Equipment Module Agents**.

The final role of the **Requirements Agent** is to select a subset of solutions from the provided solutions, that better serves the established configuration requirements, and provide these to the system integrator for final selection. After the final selection is performed this agent also informs the lower level agents about the decision. This enables the **Equipment Module Agents** to update their internal decision making models based on the success or failure of the configuration solution.

The use cases for the **Requirements Agent** provide a description for the triggering mechanisms of the agent. These provide the input methods for triggering the functionalities of the agent, which in turn allow the agent to perform its role. The use cases for this agent can be grouped into two aspects, the system integrator interactions and the equipment module agent interactions. These result in the introduction of six use cases for the **Requirements Agent**. **Figure 5.4** identifies these use cases and provides the information on the actors involved in triggering them.

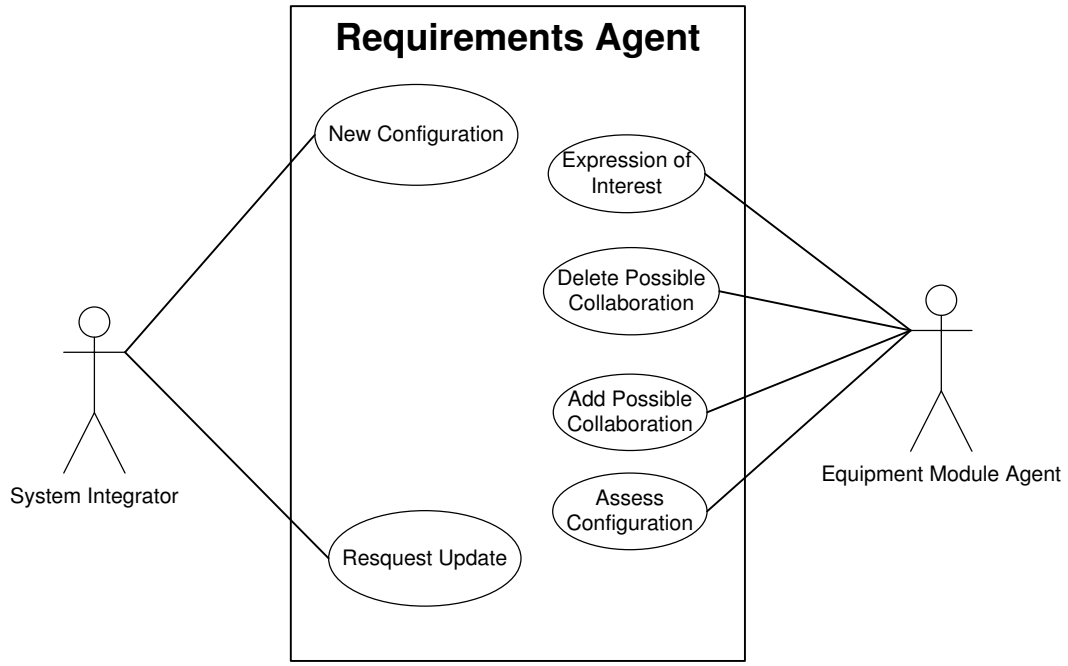


Figure 5.4 - Requirements Agent Use Case Diagram

5.3.3.2. Agent Behaviour Model

The **Requirements Agent behaviour model was** inferred from the role it has in the multi-agent environment, but also from the tasks associated with this role. **Figure 5.5** provides an overview of the **Requirements Agent** behaviour, which this subchapter will detail.

The first functionality is the ability to communicate with a system integrator. This ability enables the start of the configuration methodology, providing the means for collecting the MAS requirements from the system integrator. It is also important to note that the communication between this agent and the system integrator is required throughout the configuration process to provide updates on the configuration process. Moreover, in the case of an unsuccessful configuration, it is expected that the **Requirements Agent** may relay back to the system integrator reasons for the failure. All this can be seen in **Figure 5.5**, where the communication with system integrator is present in all procedures except for the monitor configuration process procedure that targets the **Equipment Module Agents**.

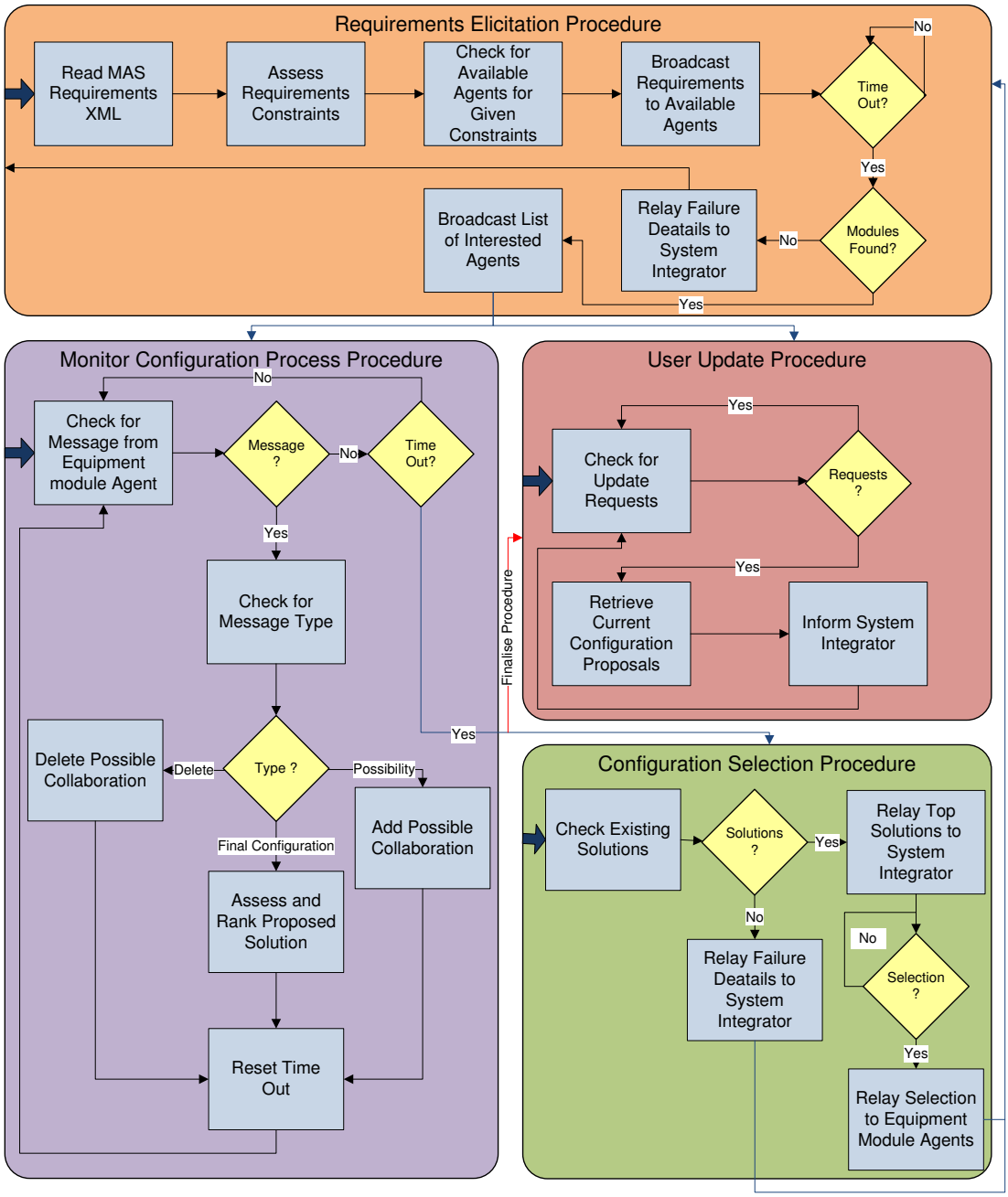


Figure 5.5 - Overview of Requirements Agent Behaviour

The second ability that the **Requirements Agent** is required to possess is inferred from its role of relaying information to the **Equipment Module Agents**. Before this happens the **Requirements Agent** needs to assess the constraints established in the requirements. This procedure of requirements elicitation provides the first filters in the configuration methodology because it prevents early on configurations that are not allowed by the requirements. The decision will be based on the business information and the system description of the requirements XML file. If a specific equipment module is required, that means that only the **Equipment Module Agents** can be involved in the configuration. If a specific equipment vendor is not allowed in

the configuration, the relevant **Equipment Module Agents** should not be contacted. It is important to note that this procedure uses only the requirement constraints provided in **Chapter 4**.

The identification of which **Equipment Module Agents** should be contacted is obviously followed by the **Requirements Agent** relaying to them the requirements. This identification involves checking which **Equipment Module Agents** are available and do not conflict with constraints defined in the requirements. This ability is seen as a broadcast functionality, where there will be a time period for expressions of interest by the **Equipment Module Agents**. If the agent denotes interest in a given configuration, then it will contact the **Requirements Agent** and establish a possible configuration thread. These threads will over time be destroyed once final collaborations between different **Equipment Module Agents** start to emerge. This can be seen in **Figure 5.5** in the monitor configuration process procedure. It can be also seen there that during this procedure open a full collaboration proposal, this agent will assess the results in relation with the weights established by the system integrator and produce a rank number which is used in the configuration selection. In sum, **Requirements Agent** will maintain a list of the configuration possibilities and solutions, allowing it to inform the system integrator of the status of the configuration at any given time.

The ability to update the system integrator on the current state of the configuration process is enabled by the user update procedure as seen in **Figure 5.5**.

The final ability of the **Requirements Agent** is to relay the ranked list of configuration possibilities to the system integrator, if one exists, otherwise the agent will relay to the system integrator the failure to achieve a configuration. The failure to achieve a configuration also entails the description for the reasons for the failure. Once the system integrator selects a configuration option, the **Requirements Agent** proceeds to inform all **Equipment Module Agents** on the decision so these can update their internal decision making models. The configuration selection procedure in **Figure 5.5** provides the illustration to enable this behaviour.

5.3.4 Equipment Module Agent Definition

Equipment Module Agents represent the equipment modules containing detailed models of their connection constraints in terms of available interfaces, capabilities in terms of assembly processes they can perform and business information. The **Equipment Module Agent**'s main objective is to participate in as many successful configurations as possible. It constantly monitors the adverts for new system requirements to identify opportunities for its own set of capabilities. Once the agent identifies an opportunity to fulfil some of the requirements broadcasted by **Requirements Agent**, it expresses interest on fulfilling the requirements and waits for the list of other interested agents. On arrival of the list of interested agents the **Equipment Module Agents** proactively engage in negotiation with other **Equipment Module Agents** to establish a collaboration which will fulfil the given set of system requirements.

The basis for negotiation is the individual capabilities of the **Equipment Module Agents** and their expected contribution to the success of the consortium. The **Equipment Module Agent** needs to find other **Equipment Module Agents** that are willing to collaborate to fulfil the set of requirements. Once a potential configuration is identified these agents will find a **MAS Expert Agent** to assess potential collaboration and identify logical faults and missing requirements. Once the collaboration is validated by the **MAS Expert Agent**, the **Equipment Module Agents** will deploy the **Performance Simulation Agent** into the virtual simulation and validation environment. After deployment the **Equipment Module Agents** will be able to interact with their counterparts and analyse the technical validity of a given configuration and its expected performance characteristics.

5.3.4.1. Agent Role and Use Cases

The **Equipment Module Agent** plays the central role in the configuration methodology, because it actively represents the equipment modules with the objective of establishing collaboration with other **Equipment Module Agents** to fulfil the established requirements. Each equipment module agent is a one to one representation of an available equipment module. Therefore, the combination of all equipment agents represents the pool of all possible configurations.

The first role of the **Equipment Module Agent** is to listen for broadcasts from the **Requirements Agent**. The architecture decision to make the requirements broadcasted requires the Equipment Module Agent to make an assessment on its interest to participate in a configuration process. This choice is considered the best approach because it eliminates the need to maintain complex tables of availability of modules. If a module is not able to participate, it simply ignores the broadcasted messages. Therefore, this agent will decide if it will actively participate in a configuration process, which is its second role.

The **Equipment Module Agent** also has the central role of broadcasting to other **Equipment Module Agents** that have expressed interest, its capabilities and negotiate collaborations with them. This iteration allows for the establishment of collaborations between different equipment modules. However, these simple collaborations are not seen as enough for valid configuration, because they are solely based on the established requirements.

The **Equipment Module Agents** also have the role of interacting with the **MAS Expert Agent**. This interaction is intended to provide an expert MAS assessment to the **Equipment Module Agents** on a given preliminary configuration solution. The fact that all agents in the collaboration interact with the **MAS Expert Agent** guarantees that each can make different decisions based on their internal models. The role is simply to provide the collaboration information to the **MAS Expert Agent**, and receive the feedback provided by this agent, namely what is missing from the configuration based on the expert knowledge contained in this agent.

The **Equipment Module Agents** are also responsible for the deployment of the **Performance Simulation Agents**, which are responsible for assessing the established collaboration. Therefore, the **Equipment Module Agents** have the role of deployment of these agents to assess different possible configuration options.

The most important role of the **Equipment Module Agents** is the decision on which collaboration it is participating. This is crucial for the configuration methodology to work. The decision to leave a collaboration will lead to an opportunity for other **Equipment Module Agents**.

The **Equipment Module Agents** are also required to continuously update the **Requirements Agent** on the current state of the configuration process. This is put in place to have a continuous stream of information for the system integrator, if he chooses to track all the configuration processes.

The final step of the configuration methodology is the proposal of a definitive configuration to the **Requirements Agent**, which is a role performed by the **Equipment Module Agents**. The definitive proposal will entail the details of the configuration, namely the equipment modules, how they are plugged together and the attributes of the configuration.

The final role of the **Equipment Module Agent** is the updating of the assembly attributes weight matrix based on the system integrator decision of selecting them or not.

The use cases of the **Equipment Module Agent** describe the mechanisms that are used to trigger this agent. Therefore, they highlight the agent iterations and how and by whom they are triggered. **Figure 5.6** provides a description of the **Equipment Module Agent** use cases and their triggering actors.

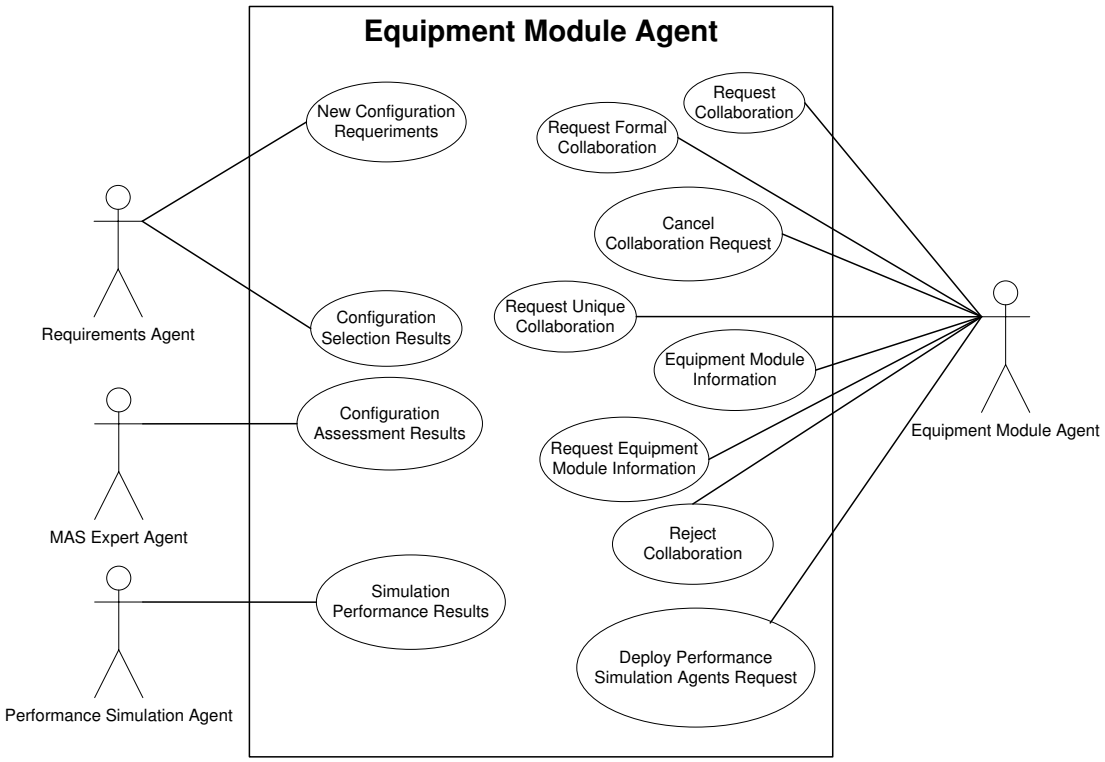


Figure 5.6 - Equipment Module Agent Use Case Diagram

The **Equipment Module Agent** is the most complex agent of the proposed agent architecture because of its central role in the configuration methodology. Being the most complex agent produces a big impact in its use cases because this agent will require a lot of interactions in order to perform its target objective. This translates in the need of at least one use case for each of these agents, plus the need to interact with agents of the same type. By analysing the agent roles and functionalities we can clearly identify the following use cases:

- New Configuration Requirements – Triggers the configuration process by providing the MAS requirements.
- Configuration Results – Provide the information if configuration was selected or rejected.
- Configuration Assessment Results – Provides the results for an expert assessment of a given configuration solution.
- Simulation Performance Results – Provides the simulation results for a given configuration solution.
- Request Collaboration – The first contact provides the subset of requirements that a given agent requires. If the agent can fulfil some of those it will accept this request, otherwise it will refuse it.
- Request Equipment Module Information – Once the agent identifies a possible configuration solution, it proceeds to request the full information of the involved modules for an assessment.
- Request Formal Collaboration – If the configuration is ranked as high a request for a formal collaboration is performed to all agents involved in the given solution.
- Cancel Collaboration Request – This provides the means to cancel a previously made collaboration proposal.
- Reject Collaboration – This use case allows for the termination of a collaboration.
- Unique Collaboration Request – This use case provides the trigger to assess if an agent wants to choose a collaboration as its final proposal for the system configuration solution.

- Deploy **Performance Simulation Agents** Request – This use case triggers the launch of the lower level agents which will lead to results on the performance of a given collaboration.

5.3.4.2. Agent Behaviour Model

The functionalities of the **Equipment Module Agent** provide the means for the agent to perform their role in the multi agent environment. As stated before this is the core agent for the configuration methodology responsible for the majority of the decision taking. This leads to a quite complex agent behaviour description, therefore this has been broken down into two parts. **Figure 5.7** where the initial steps for establishing a collaboration between different **Equipment Module Agents** are described and **Figure 5.8** that looks at the decisions for collaboration at a later stage in the configuration process.

The first behaviour of this agent is the ability to read and interpret upon creation the equipment module characteristics which he represents. This is provided by the initialization procedure. This is followed by the ability to listen to broadcasts from the **Requirements Agents**. This ability is needed since the broadcast of requirements is the first step in the configuration methodology. It is important to note this ability to listen is only useful because **Equipment Module Agents** can read the information and map it to their own capabilities.

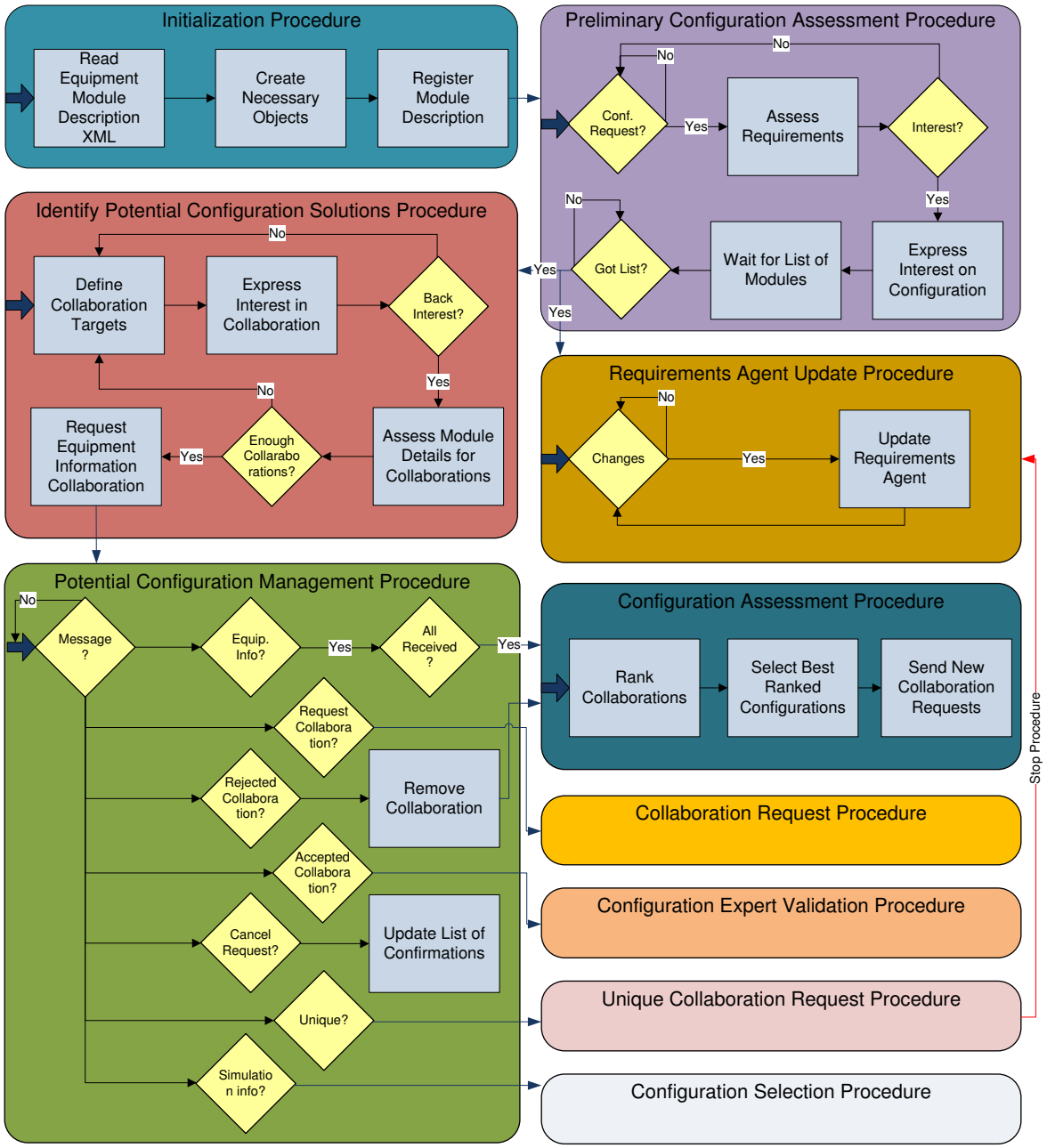


Figure 5.7 - Overview of Initial Equipment Module Agent Behaviour

The reception of new requirements by **Equipment Module Agents** triggers it to leave idle state and perform an evaluation of these requirements. This evaluation consists on the matching of its own assembly capabilities to the ones required, based on the established assembly process classification. The agent shows interest in the requirements if it can contribute to the new configuration, thus making this the first cut point of the configuration methodology. The preliminary configuration assessment procedure seen in **Figure 5.7** provides this behaviour to the **Equipment Module Agent**

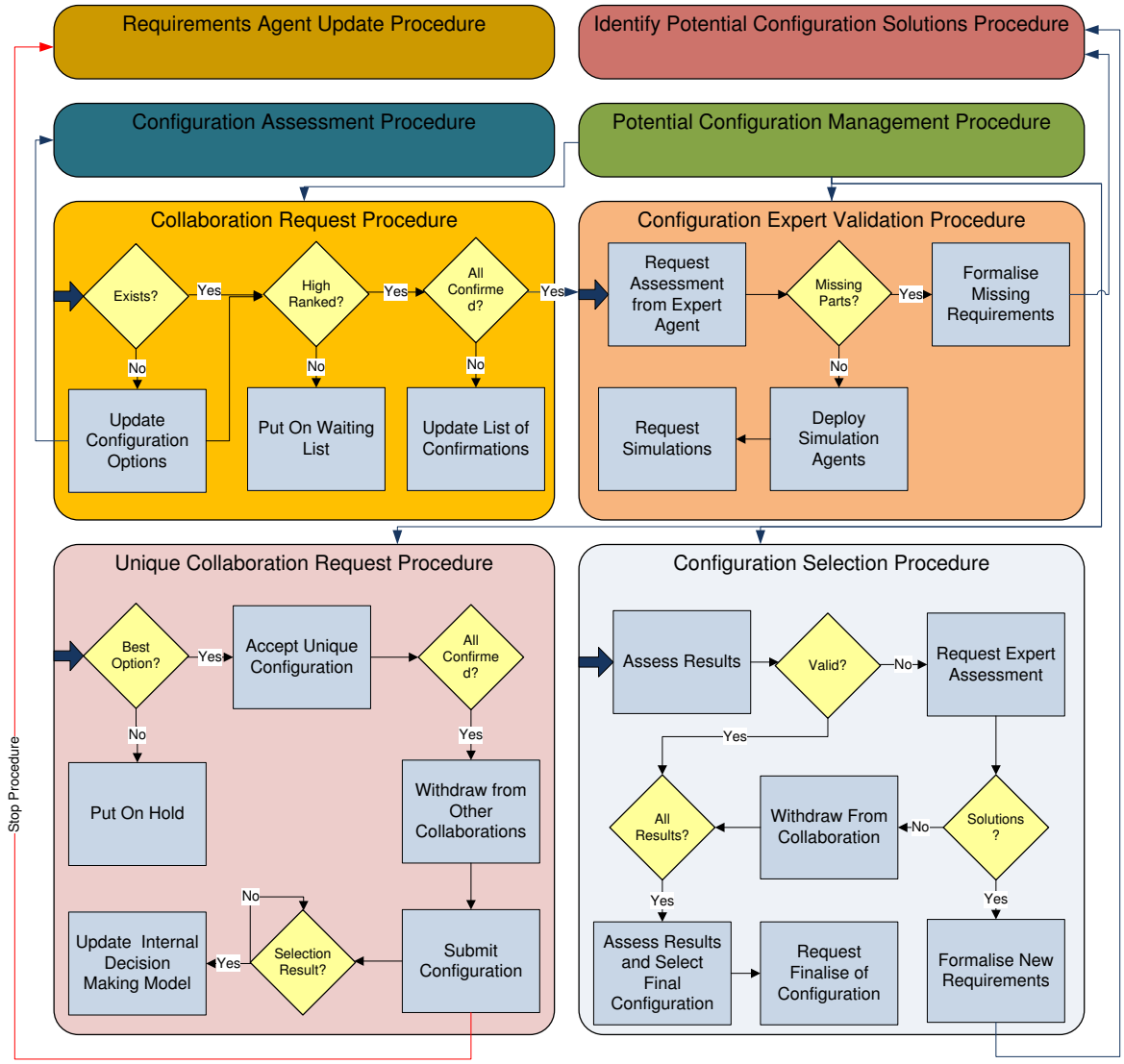


Figure 5.8 - Overview of Final Equipment Module Agent Behaviour

In the event of the **Equipment Module Agent** expressing interest in a given set of MAS requirements it triggers the identify potential solutions procedure. The **Equipment Module Agents** triggers this upon the arrival of a message from the **Requirements Agent** that contains all **Equipment Module Agents** that have shown interest in those requirements, which is an architecture design decision which prevents Equipment Modules Agents from contacting agents that are not interested in the given set of requirements. The procedure is designed to identify potential collaboration targets within the list provided by the **Requirements Agent**. This list of interested agents only provides the pool of agents available for collaborations, therefore the agents need to advertise what they are able to perform out of the requirements, and inform others to identify possible collaboration targets. The **Equipment Module Agents** requires the ability to disseminate what it can do. This

ability can be achieved through the use of an already defined agent capability, which is the interpretation of requirements which uses the model proposed in **Chapter 4.** As such the agents simply need to generate new requirements using the same model, which will consist of the full requirements minus the assembly process capabilities that the agent can provide. In the event of agents being able to provide more than one capability they simply create parallel requirements containing all combinations in which they can fulfil the requirements. In this way, the agent will keep all options open until it can perform an assessment and then make a decision on which option is a better suited solution based on its internal model. This implies also that the agent will be able to assess the replies of other agents to the new set of requirements, and based on the assembly process capabilities determine preliminary possible combinations of equipment modules that allow for the fulfilment of the requirements. This is followed by the interface validation of such configuration possibilities.

The last note of identify potential solutions procedure is the consideration of the size of the list of interested agents. If the list is quite large, an exhaustive approach that requires all agents to exchange information with each other, results in a large set of option which cause a significant increase on the computational effort. So there is a built in method that allows for the constraint of the number of contacts between agents. The method is quite simple, if a minimum number of solutions are possible within the restricted number of agents, then no more communications will be executed. However, if a solution is not found, the agent will proceed to communicate to another certain number of agents and repeat the cycle until the minimum number of solutions is reached or there are no more available agents. Once the minimum number of solutions is achieved, the agent will disseminate its equipment module specific information to those involved in potential solutions, while simultaneously requesting the same information from the involved modules.

One important consideration to add is the restriction of the **Equipment Module Agents** contacts, thus when using a non exhaustive approach, will lead the agent to have a local awareness which is composed of the agents it has contacted. This raises the possibility of receiving a formal collaboration request for a configuration solution that the agent has not considered. This will happen in the event of agents having different collaboration partners. **Figure 5.9** provides an example that highlights the need for a proposal cancelation. In it, agent 1 would have knowledge of agent 2, 3

and 4. Based on this knowledge its best option would be a collaboration with agent 2 and 3. However after he requested this collaboration, agent 4 proposes a collaboration between itself and agent 1 and 5. Agent 1 would assess the configuration and realise it is better, and if in time it would cancel the collaboration with agent 2 and 3. This clearly highlights the need to have a cancelation functionality built in for all proposals, since the best configuration solutions might change over time.

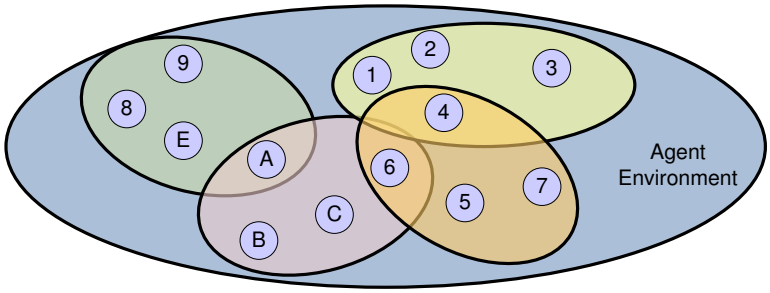


Figure 5.9 - Base Concept of Non Exhaustive Cancelation needs

The proposed extension to the local awareness concept in the **Equipment Module Agents** endows the methodology with the ability to reach better solutions based on the propagation of local decisions. This can be exemplified by analysing again **Figure 5.9** where if the combination of agents is E, A, 6 and 4 would be reachable if all agents consider it as the best.

The **Equipment Module Agents** potential configuration management procedure provides the agent with the ability to trigger different states according to information that it receives. The first information that is expected is the equipment module specific information for the solution it possesses. Once all information is present, the agent needs to perform an assessment which will rank the solutions. The agents present in the highest ranked solutions will be contacted towards establishing formal collaborations. The details of the decision making methods will be presented in **Chapter 6**.

The individual **Equipment Module Agents** make a decision on what to do with formal requests for collaborations through the configuration assessment procedure. The problem here isn't as straightforward as it might appear, since it is not a case of simply accepting or rejecting the formal collaborations. A closer look shows us how a simply accept/reject logic would fail, let us consider an agent that rejects its second

best option. If this is followed by another agent rejecting its first option, then it would have to go for its third best, since it already rejected the second best. Because of this agents put on hold proposals for collaborations that are not ranked as their best. This guarantees the agent always chooses its best possibility producing a better configuration solution. Despite the positive impact on the configuration quality this does imply that the configuration methodology will require more time to reach solutions. Nevertheless this is seen as a good compromise since one of the objectives of this work is the achievement of a methodology that not only reaches solutions but also reaches good solutions. The handling of the proposals is performed by the collaboration request procedure. One final note on this process is the rejection process, which simply removes the configuration from the pool of possibilities, and retriggers the configuration assessment procedure.

The cancel request option in the potential configuration management procedure is designed for the use of the method in a non exhaustive form. If an agent is contacted by another agent that provides a better solution after it has already sent out collaboration requests, then the agent needs to have the ability to cancel the previous request to pursue its new best option. It is important to note that this is only possible if the collaborations are not finalised, otherwise the agent will simply ignore new options.

The configuration expert validation procedure is triggered when a formal collaboration is achieved, thus when all capability requirements are fulfilled. In this state the **Equipment Module Agents** relay the configuration information to the **MAS Expert Agent** and wait for its results. If the configuration solution is not valid a new set of requirements is established, and these trigger the start of the establish collaborations state, and restarts the process for the missing requirements. On the other hand, if the solution is valid the **Equipment Module Agents** will deploy the **Performance Simulation Agents** to get simulated performance results to make their final decision.

The configuration selection procedure, establishes how the **Equipment Module Agents** react to the results provided by the **Performance Simulation Agents**. The performance results might indicate that the configuration solution performs according to the requirements, or below the requirements. In the event of

underperforming solutions the **Equipment Module Agents** will try to salvage the configuration by contacting the **MAS Expert Agent** for possible solutions. It is important to note these solutions target the enhancement of the current solution to compensate for a limitation. The absence of an **Equipment Module Agent** removal from collaboration procedure is due to the nature of the proposed methodology. In this methodology parallel **Equipment Module Agents** combinations are happening in parallel, and replacing an agent from a solution for another would only create the same solution as an existing parallel solution. If no solutions are available the configuration solution is dissolved, if a solution is presented in the form of new requirements then the **Equipment Module Agents** trigger the start of the identify potential configuration solutions procedure, and restarts the process for the missing requirements. Once all high ranked solutions are assessed or dismissed, the **Equipment Module Agents** proposed the unique collaboration to the members of their highest ranked solution.

Upon receiving a request to a unique collaboration, the **Equipment Module Agents** trigger the unique collaboration request procedure. If the request comes for its highest ranked solution the agent waits for all agents involved in the configuration solution. If the request is not for its highest ranked solution it is simply put on hold. This uses the same principle as before to ensure that the agent selects its best solution possible.

An important consideration to be had here is that agents will make decisions at different times, this means agents need to deal with information that is sometimes ahead of their decisions, and sometimes behind. While some agents might be proposing formal collaborations others have not yet gotten all required information. Therefore the functionality of the agents needs to address the real-time constraints of parallel computing, while maintaining the functional requirements detailed above.

The ability of the **Equipment Module Agents** is the update changes to **Requirements Agent**, which runs in parallel to the configuration method, and simply updates the **Requirements Agent** of the current state of the configuration process through the **Requirements Agent** update procedure.

5.3.5 MAS Expert Agent definition

The proposed **MAS Expert Agent** is a MAS expert that focuses on two aspects that are foreseen as the base pillars for the guaranteeing sound configuration solutions, namely constraints of physical aspects and assembly processes (EUPASS [4]). This encompasses the logical completeness of the assembly process requirements and physical constraints that go beyond the mere connectivity of the equipment modules.

The decision to have this expert knowledge to be outside of the **Equipment Module Agents** is the nature of this knowledge. This knowledge is not module specific but rather more global, the type of knowledge that system integrators possess (EUPASS [4]). The fact that this knowledge covers a higher level configuration aspects implies that it is dependent on the context of given configuration solution. As such this knowledge should not be part of the **Equipment Module Agent**. Nevertheless the use of this knowledge in the configuration methodology would provide better solutions. Therefore the **MAS Expert Agent** is introduced to cater for this.

The analysis of this expert knowledge raises two problems; one is the capture of such knowledge; the other is the evolution of this knowledge (Onori and Oliveira [18]). This knowledge currently sits on the brain of system integrators; therefore it is not straightforward to get it all at once into an agent. Nevertheless, it is envisioned that this knowledge might be captured in the future, which would have a huge positive impact on the configuration methodology results.

The other problem with this type of knowledge is the fact that it changes over time. Therefore even if it was possible to take a snapshot of the whole knowledge today, this would not be the same as another taken tomorrow.

It is clear that having this expert knowledge provides better and more complete configuration solutions. At the same time, it is also clear that this should not be hardcoded into the methodology, therefore the decision of taking this knowledge out of the **Equipment Module Agents** and the creation of an advisory agent that is able to be extended in the future to better the configuration results without having to rework the configuration methodology.

5.3.5.1. Agent Role and Use Cases

The main role of the **MAS Expert Agent** is to supply expert knowledge about a MAS configuration. There are two distinct periods of the configuration method where the **MAS Expert Agent** is asked to provide input, the validation of a formal collaboration and in the event of a failure in the MAS performance assessment. The other aspect to consider in the definition of the **MAS Expert Agent** role is the two aspects that it targets, namely the logical constraints of physical and assembly processes.

The validation of a possible configuration is seen in two fold. The assembly process side and the physical aspects of the modules. The **MAS Expert Agent** needs to assess the completeness of the configuration based on these two aspects. It is important to highlight that this agent not only assess the solution against the requirements, but also has the ability to provide missing requirements that were not formalised but are necessary for a valid system configuration.

The second main role of the **MAS Expert Agent** consist of assessing if something can be done to improve a given configuration that does not have valid simulated performance indicators. The idea is that expert knowledge can be used to compensate for a given bottleneck, or an accuracy deviation might be compensated by a measuring system. The **MAS Expert Agent** would be able to provide this feedback and the solution might be improved and salvaged.

The use cases of the **MAS Expert Agent** are quite straightforward since there is only one actor, the **Equipment Module Agent**. This agent is a passive agent, only acting upon a trigger, thus the identification of the triggering is quite important. **Figure 5.10** provides the use case diagram for the **MAS Expert Agent**.

The first use case of the **MAS Expert Agent** is the request validation assessment, which is used by the **Equipment Module Agents** in order to request a completeness and validation assessment of a given configuration solution.

The second and final use case is the request performance assessment, which is again triggered by the **Equipment Module Agents**. This use case allows for the agent to get possible solutions for performance failures in a given configuration, namely if there are strategies that can correct the failures.

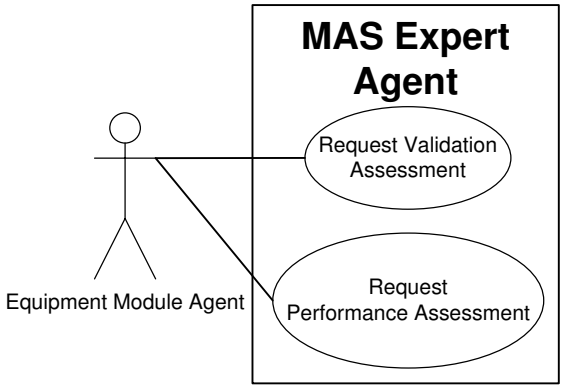


Figure 5.10 - MAS Expert Agent Use Case Diagram

5.3.5.2. Agent Behaviour Model

The abilities of the **MAS Expert Agent** allow for the execution of its role in the agent environment. However the role of the agent describes nothing of its content. As said before this agent is introduced into the system to provide expert knowledge to the system. For the purpose of this work it is expected that this knowledge is hardcoded into the agent, meaning that these agents always exist with their knowledge in the pool of agents. In the event of change to the expert knowledge, the agents will be updated accordantly. The specific methods to collect and distribute the knowledge are considered outside of the scope of this work. Nevertheless, this work identified the importance of having this knowledge in the configuration method, therefore proposes a lightweight knowledge model where some basic rules are defined to exemplify the envisioned operation of this agent. **Figure 5.11** provides an overview of the **MAS Expert Agent** behaviour.

The behaviour for the **MAS Expert Agent** is quite straightforward, since it consists of the execution of only three procedures. The agent waits for an assessment request within the assessment request procedure. On the arrival of an assessment request, the agent determines the request type, if it is a validity request the expert configuration assessment procedure is triggered. If it is a performance failure assessment it triggers the performance failure procedure. Otherwise it simply replies to the request as unknown request.

In the expert configuration assessment procedure, the **MAS Expert Agent** assesses the completeness of the proposed solution. The completeness assessment is viewed as a set of rules that a given solution needs to verify. If a solution is not complete, it

is obviously not valid. In this case, the **MAS Expert Agent** can only determine if the solution is incomplete if it has some internal rules that are able to determine missing elements. A simple example of this is all configurations solutions require a base frame for the equipment modules to plug in to, if this is missing, then the **MAS Expert Agent** will formalise these missing requirements and feed them back to the **Equipment Module Agents**. It is important to note that for this ability this agent needs to be able to produce requirements as defined in **Chapter 4**, so it can enhance the original set of MAS requirements.

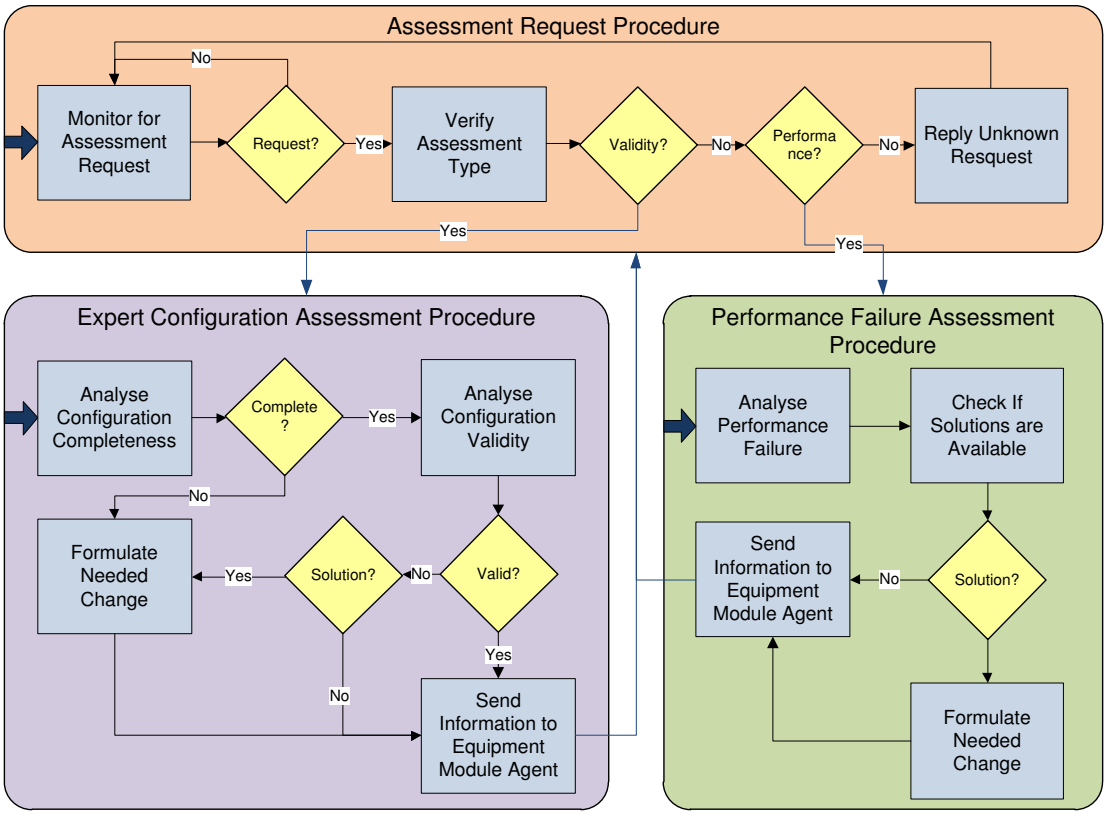


Figure 5.11 - Overview of MAS Expert Agent Behaviour

In the case of the configuration solution being complete the **MAS Expert Agent** analyses the validity of the system. This assessment will have two components, namely if the configuration violates any rules and if it does not follow pre-existing configuration patterns. Both rules and patterns will be part of the internal model of the MAS Expert. This is introduced to add the ability for rejection of a configuration solution even if the reasons cannot be formalised. The idea is this way the methodology will not waste computational time pursuing solutions that will be invalid due to global and context specific aspects. It is envisioned that the knowledge that this agent holds will evolve over time, so new rules and more configuration

patterns. Ideally invalid configuration solutions assessment should always provide solutions, however this is a not a realistic target for the near future.

The performance failure assessment procedure of the **MAS Expert Agent** contains the functionality to deal with performance assessment requests. These requests occur after the simulation takes place, therefore the possibilities of the performance failures are clearly identified. These will consist on the results provided by the **Performance Simulation Agents**, namely results on the performance of cycle time, accuracy, repeatability and energy consumption. The type of performance failure is crucial for the ability to formulate a possible mitigation strategy. For example, if the repeatability of the configuration solution underperforms due to stacking of different equipment modules repeatability, the **MAS Expert Agent** could suggest based on internal rules the introduction of a measuring module that could compensate this stacking of error.

5.3.6 Performance Simulation Agent

Performance Simulation Agents represent the process capability of an equipment module. What this work proposes is that each assembly process step can be represented by a **Performance Simulation Agent** that is aware of the type of assembly process, where it sits in the overall assembly process classification and its plugability requirements and options. It is envisioned that this will provide the agents with the ability to emulate the operation of the assembly process step they represent. Thus in addition to this, it is proposed that these agents are able to connect to other agents of the same to establish a virtual assembly process sequence. Once this is established the agents can exchange information in the established network, therefore simulating and providing results for MAS performance characteristics.

The required information for these agents sits inside the equipment module description files, and therefore it is already contained inside the **Equipment Module Agent**. Because the information is already inside these agents one could argue that the creation of another agent type is not necessary. However, the proposed configuration methodology predicts parallel collaborations among **Equipment Module Agents**, which would lead to much more complex module agents and a lot more computational strain in the machine that is running the agent environment. By

detaching this knowledge to lower level agents, the agent can be deployed in different machines to perform the simulations providing a distributed computation frame, on the most computer intense process of the configuration methodology, the simulation of possible configurations.

The assembly system performance characteristics of MAS are mostly related with the capability side of the system. The characteristics that the **Performance Simulation Agents** can simulate are highly dependent on the information contained in the assembly process descriptions. Assuming all information is present, the **Performance Simulation Agents** are able to perform simulations on the cycle time, the accuracy, the repeatability and the power consumption. These have been selected because they have slight variations under runtime, which are important to consider for the decision involved in the proposed configuration methodology.

The introduction of simulation capabilities allows for more detailed information on configuration solutions, which enables better decisions of the **Equipment Module Agents** in relation to potential solutions.

5.3.6.1. Agent Role and Use Cases

The **Equipment Module Agent** is the creator of the **Performance Simulation Agent**, being in a hierarchically superior position. This means that the **Performance Simulation Agent** is a lower level agent in relation to it and is, in effect, owned by it.

The role of the **Performance Simulation Agent** in the configuration methodology is to provide simulations on the MAS performance attributes that are strictly related with the logical side of the assembly system. Therefore its only role in relation with the high level agents is to provide results for the given configuration proposal.

The role the **Performance Simulation Agent** plays in the other lower level agent is to act as the representation of a given capability and interact with the other **Performance Simulation Agents** that are part of the proposed configuration solution. Based on the configuration details supplied by the **Equipment Module Agent**, these agents can establish a virtual logical configuration that represents the logical sequence of the given configuration. Therefore, the first use case and the trigger for operation of the **Performance Simulation Agent** is the configuration

specification. Once this is done the agents will go through a series of simulations exchanging information towards establishing the MAS performance results for the given configuration. To enable this, the first use case is the establish connection, which allows for the creation of the virtual network that represents the assembly process configuration. The second use case is the simulation data, which enables the agents to exchange and update the simulation object. The third and final use case is the simulation results, which allows all the agents to receive all the simulation results so they can relay them back to their creator, the **Equipment Module Agent**. The final use case is the suicide order, which will terminate the agent. A use case diagram for this agent can be seen in **Figure 5.12**.

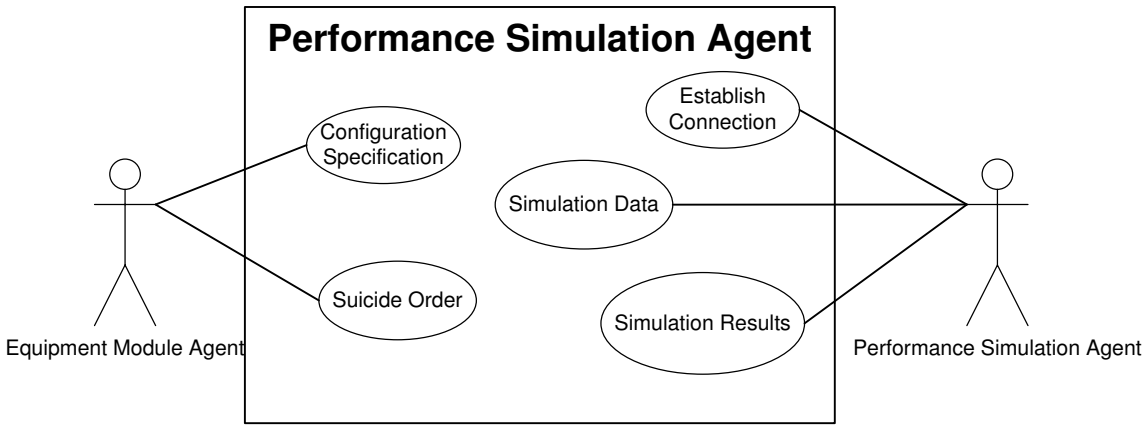


Figure 5.12 - Assembly Process Agent Use Case Diagram

5.3.6.2. Agent Behaviour Model

The **Performance Simulation Agent** comes to existence on demand, meaning it is created with the sole purpose of simulating a given configuration. As such, the monitoring for configuration specifications is the first step in describing its behaviour. On creation this agent will inherent the knowledge related with the assembly process which enables the agent to represent it, since this is the same for any deployment on any configuration assessment.

Once the **Performance Simulation Agents** are deployed and they have the configuration specifications, they proceed to create a virtual network that represents the configuration solution from the assembly processes point of view.

The virtual network provides the means for the agents to emulate a Petri net based model adapted for the synthesis and simulation of MAS behaviour (Ferreira et al.

[128]). A Petri net is a graphical tool for the formal description of the logical interactions among parts or of the flow of activities in complex systems. Petri nets are composed of places, transitions and arcs, which are combined to represent a logical description of a system. The agents represent the place holder and the transitions represent how the performance characteristics are affected based on the assembly process type. The **Performance Simulation Agents** states and functionalities will allow for the execution of this Petri net method for all the established performance characteristics. One important adjustment to this network is the connection between the terminal assembly processes and the starting assembly processes. The passing of information between these two processes is considered the finish of a simulation cycle, therefore is when the results are stored. Further details on this model and its operation inside the **Performance Simulation Agents** can be found in **Chapter 6**

The **Performance Simulation Agent** will possess five procedures, where the first is the simulation request procedure which is triggered by the **Equipment Module Agent**. During this procedure the configurations specifications are obtained which is the trigger for the establishing virtual configuration procedure. In this procedure the **Performance Simulation Agent** is required to interact with the other **Performance Simulation Agents** which represent the assembly processes directly connected to it. Once this is done, the agent verifies if it is a trigger of the simulation process. If the agent is the simulation starter agent the simulation driver procedure is triggered, otherwise the simulation procedure is triggered. **Figure 5.13** provides an overview of the agent functionalities and states.

The simulation driver procedure is performed by the **Performance Simulation Agent** identified has the simulation starter, and it starts with the creation of the simulation object. This object will contain a place holder for the cycle time, accuracy, repeatability and power information. The object is representative of the components of a given product going through the assembly process sequence. Once this is created, the agent will update this object according to the assembly process information, and send the updated objected to the agents it is connected to in the virtual network. When this object returns the **Performance Simulation Agent** checks if the number of configurations was executed, if not it records the results and restarts the process, otherwise, it simply records the results and triggers the finish of

the simulation process by broadcasting the results to all the **Performance Simulation Agents** involved in the simulation, and triggers the simulation finalization procedure

The simulation procedure is a passive procedure where the **Performance Simulation Agent** waits for messages from other **Performance Simulation Agents**. In the case of the message being the simulation object, the agent simply updates it based on the type of assembly process that it is representing, and passes the object on to the **Performance Simulation Agents** directly connected to it. The other message type is the signalling of the end of the simulation which will trigger the simulation finalization procedure.

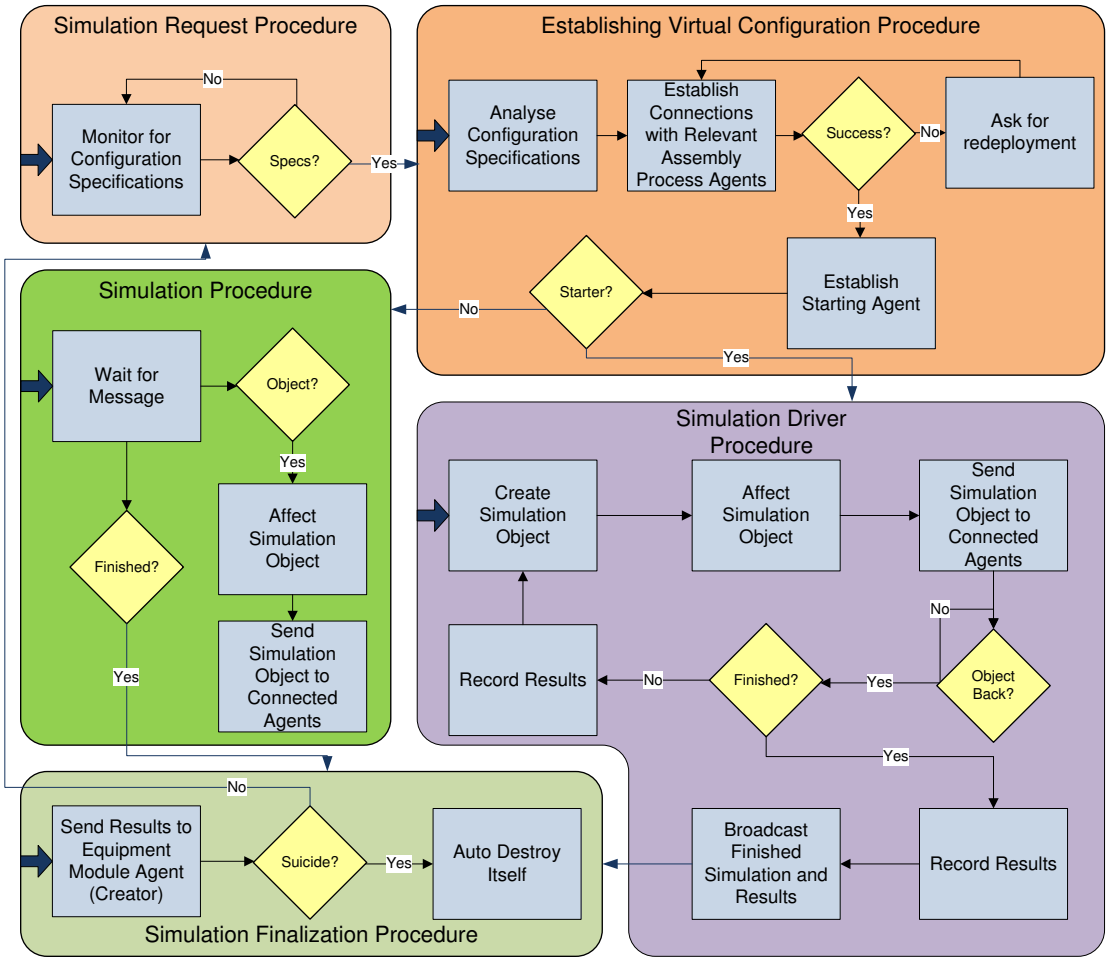


Figure 5.13 - Overview of the Performance Simulation Agent States and Functionalities

The simulation finalization procedure is quite straightforward; the agent relays the performance simulation results to the relevant **Equipment Module Agents**, and waits for a termination order or a stand by order to restart the simulation process.

5.3.7 Agent Interactions

The description of the agent interaction is a key aspect of any agent model, because it maps the possible sequences while identifying the message types that need to be created. The base concept of any agent system is the communication ability that agents possess. However, the ability to communicate implies a clear understanding by all involved agents of a common language and structure for the messages. Therefore it is of extreme importance to map the agent interaction and identify the required message types, required responses and expected sequences.

The agent interactions enable the behaviour of the agents to fulfil their role in the agent architecture. From the previous descriptions it is straightforward to establish who speaks to whom and at what stage of the configuration process this occurs. However these do no establish formally a message type, if a response is required and what sort of response. Agents only know how to react to messages if they are expecting them. Therefore an analysis of the agent interaction should be detailed. Toward that end it is useful to break the interactions down into different stages. **Figure 5.14** provides an overview over the different states that the configuration methodology goes through.

The first stage is between the **Requirements Agent** and all **Equipment Module Agents**. In this stage the MAS requirements are sent to **Equipment Module Agents** and these assess their interest. If interested, the **Equipment Module Agents** provide an expression of interest to the **Requirements Agent**, otherwise they do nothing.

The second stage is triggered once enough time has elapsed for **Equipment Module Agents** to express interest. Once this occurs the **Requirements Agent** interacts with all interested agents to disseminate the list of all interested agents.

The third stage is the interactions between the interested **Equipment Module Agents** to achieve configuration solutions. During this stage the **Equipment Module Agents** will exchange information on their module characteristics, and will validate configurations with **MAS Expert Agents**. The interactions in this stage enable the core decision making process of the configuration methodology.

The fourth stage is the iterations between the **Equipment Module Agents** and the **Performance Simulation Agents**. The **Performance Simulation Agents** are

deployed and the simulation information is passed to them. They are able to interact with each other to simulate a given configuration and provide the results through an interaction to the Equipment Module Agent that deployed them.

The fifth and final stage is final interaction between the **Equipment Module Agents** after these include into their decision making capabilities the simulation results. The interactions here provided the ability to reach final configuration solutions which will be submitted through another interaction to the **Requirements Agent** for final selection.

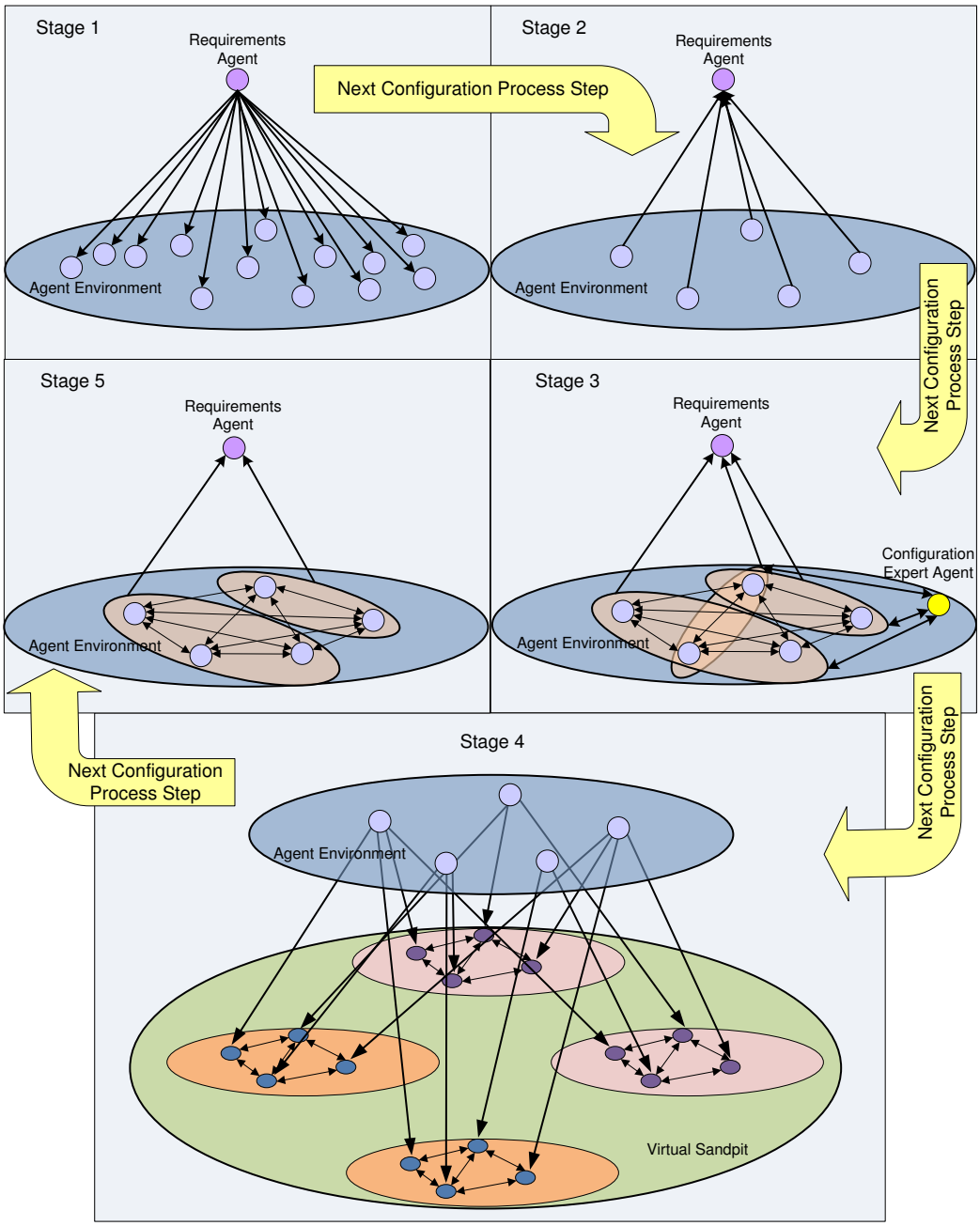


Figure 5.14 - Overview of the Configuration Methodology Steps

The stage overview provides a glimpse of the interactions; however these are much more complex. In the stage overview it is presented an example that is representative of a main flow of interactions. This can also been seen in **Figure 5.15** provides an overview sequence diagram of the main agent interaction cycle, the same one as described above. In this main sequence the agent decisions points are identified, but it assumes that the answer is always positive to simplify the sequence. Also an **Equipment Module Agent** is detached from **Equipment Module Agents** pool to highlight the heavy interactions between the agent types. Detailed sequence diagrams for all agent interactions can be found in **Appendix B**. The set also contains the identification of message types, and the specific procedures in each allowed sequence.

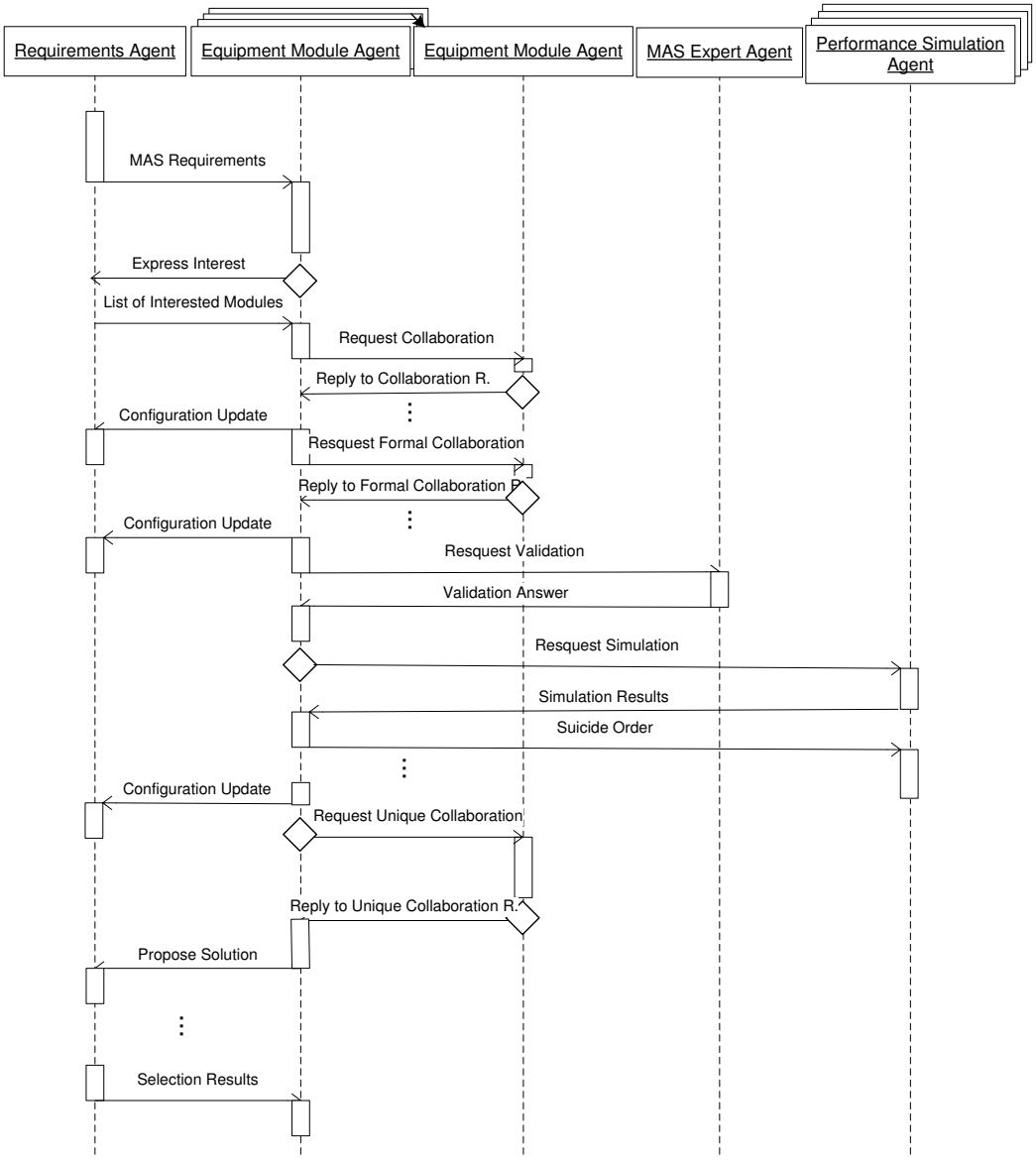


Figure 5.15 - Main Configuration Methodology Sequence Diagram

In **Chapter 6** the used communication protocols are presented. These are included in the detailed sequence diagrams for all agent interactions can be found in **Appendix B**.

5.4 Agent Architecture Deployment

The deployment of the proposed architecture is a crucial step for the enabling of the configuration methodology, while supporting some of decisions taken in the architecture design. In a multi suppliers environment with an infinite number of modules it is not feasible to have all of the **Equipment Module Agents** running on the same computer. The problem is the computational strain to reach solutions would rise exponentially based on the number of available modules. Therefore it is proposed that the deployment of the Equipment Module Agent be done in the suppliers servers, allowing them control over the agents, and more importantly distribute the computational load across different computers. The equipment supplier will have the motivation to have this since it potentially can bring new business for them, while for the system integrator (representing the customer) it is advantageous since solutions will be provided quicker due to the distribution of the computer load. **Figure 5.16** provides a deployment overview highlighting the communications across different computers.

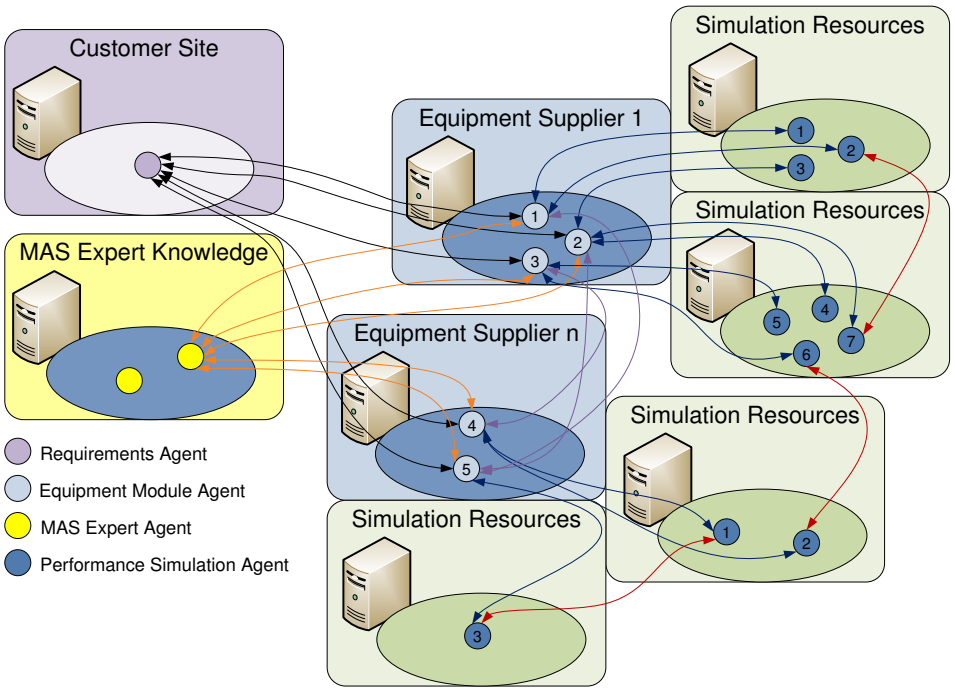


Figure 5.16 - Agent Architecture Deployment Overview

The other important aspect of this distributed deployment is that the Performance Simulation Agent can also be deployed in other machines to distribute to computer processing load, facilitating quicker solutions.

The final consideration of the deployment of the proposed agent architecture is placement of the **MAS Expert Agent** in a separated machine that is updated by configuration experts and where the libraries proposed in **Chapter 4** would also be placed. The information on this machine could be in other machines in order to take advantage of the distributed computing paradigm. Nevertheless, it is crucial that all updates made to the **MAS Expert Agent** and the library change at the same time across different machines to ensure the proper operation of the configuration methodology.

5.5 Chapter Summary

In this chapter a multi agent model for the self-configuration of MAS was proposed. The chapter contains detailed agent descriptions, their roles and behaviours that enable the self-configuration methodology. It also provides a detailed description of the agent model, as well as the necessary interaction to ensure the execution of the self-configuration methodology.

The agent architecture provides an original representation of MAS that is able to reflect its concepts. Furthermore, the proposed agent architecture caters for the evolution of expert knowledge over time, by providing the means to introduce new knowledge without a need for changing the configuration methodology. Finally the proposed agent architecture provides a simulation level that provides early simulation results for potential configuration solutions. Furthermore these results are use in the configuration methodology towards achieving better results.

6 Local Behaviour Models for Distributed Self-Configuration Methodology

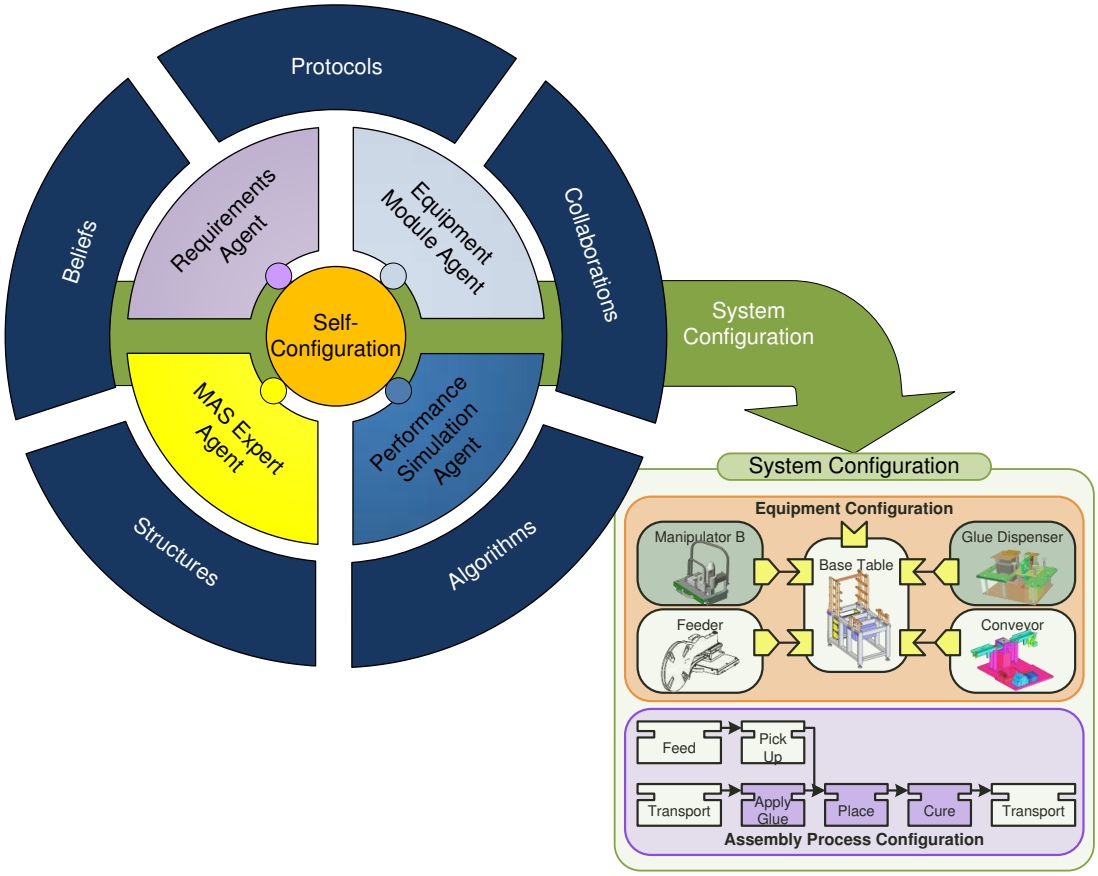


Figure 6.1 - Overview of Enabling Aspects for Emergence of Configuration in Agent Architecture

6.1 Introduction

In this chapter we will cover the specific methods of the multi agent architecture that will enable self-configuration of modular assembly systems. The chapter will break down the agent specific methods as well as provide the method for assessing the validity of the configuration results.

The proposed configuration methods were developed based on the described model of **Chapter 4** and the characteristics of the agent environment in **Chapter 5**. This is important to highlight because the entire input information and agent environment definitions for the proposed configuration methods are already defined in these chapters.

One important note for this chapter is the distinction between configuration and reconfiguration of the module assembly systems. For the purposes of this work reconfiguration is defined as a configuration with some extra constraints. The constraints in the event of a reconfiguration process are the description of the existing system, including the ability to force the use of certain equipment modules. This ability in conjunction with the introduction of equipment module agents with quite advantageous characteristics, such as near zero cost, provides the basis for the reconfiguration using the same methodology as for the configuration process. These constraints are as defined in **Chapter 4**.

The development of methods for an agent environment requires a clear communication structure. This structure entails the definition of available protocols, which enable agents to trigger other agents using predefined collaboration rules that are understood and followed by both. Despite the existence of protocols for multi agent systems, these tend to be domain and solution specific (Kraus [98]). Therefore, protocols for this multi agent system need to be described in this chapter.

In order to develop the methods for the configuration of modular assembly systems using a multi agent environment, the configuration process steps should be clear as defined in **Chapter 5**.

The configuration of modular assembly systems will be driven by an established set of capability requirements. This is the first stage of the configuration process, which

consists of the individual equipment module agents matching their own capabilities to the ones required. Once this is done a cluster of interested equipment modules is created (Oliveira [60]).

These **Equipment Module Agents** will then need to establish preliminary collaborations with other equipment module agents, in order to establish potential configurations. This stage will require an assessment by each individual agent. The method for this assessment will be presented throughout this chapter.

The equipment module agents will be able to participate in a number of different potential solutions. This raises an issue of participation on multiple solution clusters. If a solution is not possible the agent will expand its search parameters for other agents until no more equipment module agents can be found. This highlights the iterative nature of the method. There are two outcomes for this stage, either no solution is found, or a series of potential configuration solutions are found.

The next stage of the configuration process is the assessment of the solutions by the configuration expert agent. The assessment consists of the configuration expert agent checking its internal knowledge for existing configuration patterns and relaying missing elements to the established collaborations. This stage might have required the repetition of the prior stages, if missing elements are identified.

The formulation of the next assessment requires the creation of the simulation agents. These will perform specific methods to validate the potential configurations. Once the results are achieved, these are relayed back to the equipment modules for final assessment.

The equipment module agents perform the final assessments of the potential configuration solutions and decide on which one they foresee to be the best one. This choice involves also the pulling out of other potential configuration solutions, which in turn will lead these collaborations to find other potential equipment modules.

The final stage is the final assessment of the requirements agent for the selection of the top three configurations for system integrator decision.

It is important to underline that the proposed configuration methodology is based on the emergence that distributed systems can obtain (Kennedy and Eberhart [132]). The principle is that simple rules distributed across multiple agents while enabling them to interact will result in this emergent complex solution. In addition, the fact that the domain of modular assembly system raises issues of future scalability of the different systems highlights the need for a distributed system than can be enhanced with more equipment modules and new concepts.

The description of the emergent complexity of the methodology requires firstly the development of the distributed blocks, in this case the agents. The agent environment has been described in the previous chapter, however the specific decision making methods have not been presented yet. Therefore, this chapter will start by covering the specific communication requirements of the agent environment, and this will be followed by the detailed methods for each agent and finally the emergent method of distributed self-configuration of modular systems.

6.2 Communication Definition

The ability to communicate lies at the core of the agent definition. Without this ability the whole agent concept would be void. Therefore it is a crucial development for any agent environment.

Agent technology platforms provide extensive models and communication solutions which provide quite flexible solutions. Therefore these were used in the development of the agent environment and provide the basis for the development of the communication. However, despite the extensive literature on communication models and methods, these still require extension due to the domain's specific issues (Kraus [98]). The current best practices in agent technology use the FIPA protocols as a baseline for the establishment of communications between agents. FIPA provides a quite open model which covers generic agent interactions, therefore reducing quite significantly the effort of developing the agent communication methods. Nevertheless, to establish a multi agent environment, clear and specific protocols need to be defined.

A crucial factor in communication is the language the agents use to understand one another. This is one of these issues that is solely related to the problem domain. The model presented in **chapter 4** provides the information that is domain specific for the configuration of MAS, therefore instead of creating an agent specific language, the use of this model is proposed. The model has been described in XML which is easily incorporated into agent technology, since XML is a standardised description in the computer science domain. This underlines the importance of a transparent and computer interpretable description as provided in **chapter 4**.

The need for the definition of a language is only one of the requirements for viable communication between agents. The other requirement is the definition of communication protocols. Protocols define the rules and regulations for agent interactions. This is a crucial element establishing collaborations among agents, since the rules for establishing these collaborations, rules for cancelling a collaboration, rules for submitting a solution and rules for the interactions of different agent types all need to be defined. The rules also have a big impact on the decision making process, not for the results but to ensure that the agent environment works properly.

The rules will provide the guarantee that conflicts between decisions of different agents do not create stalemate situations. The fact that different agents have different beliefs to what is the best solution might cause stalemates in the established architecture unless clear rules exist. Agent solutions are driven by communication between the agents, this communication enables the individual agent decisions, and therefore the communication protocols need to prevent the agent stalemate situations.

Once the language, the rules and regulations are defined then the focus lies in the decision making methods that enable the emergence of MAS configuration solutions.

In this chapter only the protocols for the requirements agent and the equipment module agents will be described, because these two agents provide the two major input points in the system. The other agents only require protocols to interact with these agents, interactions that are triggered by these main agents. As such their specific protocols are the mere counterpart of the ones provided for the equipment module agent and the requirements agent.

6.2.1 Requirements Agent Communication Protocols

The **Requirements Module Agent** communicates with the system integrator and the **Equipment Module Agent**s. However this work does not cover the communication protocols with the system integrator since these would require a specific frontend solution which is not relevant for the developed aspects of the configuration methodology. Nevertheless it is recognised that extra protocols will be required for the communication between the system integrator backend and the **Requirements Agent**.

The first step of the configuration process is the broadcast of the requirements, which is clearly an information protocol for the consideration of the **Equipment Module Agents**. The language is clear between the two agents, since both are aware of the structures described in **chapter 4** for the description of MAS requirements. The protocol is not time critical, however as defined in **chapter 5** there will be **Equipment Module Agents** that will not show interest, so it is required a definition of a timeframe for **Equipment Module Agents** to show interest in the requirements. This first protocol is defined as **Broadcast of the Requirements**, and is independent from others to ensure that it can be used for other future agents simply to share the MAS requirements.

Although the **Broadcast of Requirements** protocol is independent of the other protocols, it will result in the triggering of the configuration process protocols. The **Equipment Module Agents** will demonstrate interest in fulfilling the MAS broadcasted requirements, which requires a clear protocol for this action, the **Express Interests in Requirements** protocol. The type of message is a request to be involved in finding a solution for the broadcasted requirements. The intention of making this a request is because the **Equipment Module Agents** require an answer back with all agents that are also interested in the MAS requirements. Therefore the communication protocol is triggered by the requests which only require an ID to which MAS requirements the agent is expressing its interest. The Requirements Agent upon arrival adds it to its interest list, and once the timeframe for interest expression is expired it simply broadcasts this list of agent addresses to all, thereby confirming the acceptance of the expression of interest.

The next required protocols are related to the creation of possible collaborations and all their maintenance aspects, namely update actions and delete actions and final submission actions. The nature of these protocols requires them to be defined separately since this triggering action might or might not occur, it is good practice to separate all things that are independent from each other, and even though the maintenance aspects can only be perform on existing solutions, they might occur or not, therefore it is an independent event which needs to be treated separately.

The **Creation of a Configuration Solution** protocol is triggered by any **Equipment Module Agent**, however the finalisation of this protocol is only performed upon arrival of the creation request of all involved agents, this ensures that solutions are only accepted if they are proposed by all **Equipment Module Agents** involved in the solution. This also highlights the need for the description of the configuration when a creation request is performed, which again uses the model in **chapter 4** for MAS configuration solutions.

The **Update of the Configuration Solution** protocol follows the same procedure as the creation of a configuration, since updates need to come from all parties. The only difference is instead of creating a new configuration solution the **Requirements Agent** will replace its previous one with the currently sent.

The **Delete Configuration** protocol requires only the configuration solution ID and reason, and contrary to the other solutions it is an action that can be confirmed even if the **Equipment Module Agents** have not confirmed it. The reasons for failed configurations have been defined in **chapter 5**, and are used for providing extra information to the system integrator.

The final protocol is the **Assessment of Solution**, which is the most complex protocol of the **Requirements Agent**. This protocol follows the same approach as the creation of configuration solution, meaning it will only be triggered if all involved agents trigger it. However, the response will have to wait for all solutions to be found, before actually providing the results to the system integrator. After the system integrator chooses its system, the reply to all agents in all solution is performed to communicate if their solution was successfully accepted or rejected. The type of this protocol is a proposal, while the result will come under the form of acceptance or rejection type. The definition of the content of these messages is quite

straightforward only the solution ID is required for the exchanges. The detailed sequence diagrams for the **Requirements Agents** where these protocols are invoked can be found in the **Appendix B**.

6.2.2 Equipment Module Agent Communication Protocol

The equipment module agent plays a pivotal role in the whole configuration methodology. Therefore, this agent needs to execute a wide range of communication protocols, which are the most complex. Because the communication protocols for the requirements agent have been already defined, these will not be covered in this description since it would be a mere mirror of the previously described protocols. The biggest amount of interaction in the proposed configuration methodology occurs between different equipment module agents. The several stages of these interactions can be found in **chapter 5**, while in this chapter the specific problems of each of these stages will be detailed.

The **Establish Collaboration** protocol is the first one that is required for the interactions between different equipment module agents. The importance of the protocol is quite clear since it provides the formal means to establish a preliminary collaboration. The basis for the decision on whether or not to establish collaboration will be defined in the operational description of the equipment module agent. The protocol however, only requires the description of the options and how to proceed in relation to them. The description provided in **chapter 5** tells us that the equipment module agent will use an updated version of the requirements definition that includes the original requirements minus what the agent will contribute. Therefore there is a clear definition for the content of the message. This protocol falls clearly under the request category, where the agent requests other agents for collaboration based on a given set of requirements. The answer to this is either an acceptance or rejection of the request. The nature of this protocol is to provide a preliminary collaboration, therefore it is designed to achieve simply that, and it is self-contained because it result is a list of collaboration, which is a self-contained result.

The next step of the configuration methodology requires the equipment module agents to exchange information that contain the individual descriptions of all the members involved in a given collaboration. This step requires that all agents

involved in collaborations have provided this information before the individual equipment module agents can proceed to the next step. Therefore this requires an **Exchange Module Information** protocol, which once triggered, it is only complete upon the arrival of all the information. The definition of the type of protocol is quite straightforward, since it is a mere request. However, the contents of the reply need to be clearly defined. In **chapter 4** a model for the description of equipment modules is present and can be used in this exchange. An important note is the restriction that upon this request equipment module agent always needs to provide an answer, otherwise the configuration method might freeze. Also within this protocol and based on the described models, the request for the equipment module information will only occur if a preliminary collaboration exists. However, the protocol should cater for the eventuality of this request being performed by an agent that does not have a preliminary collaboration established. In this situation, the agent should reply an invalid request, to ensure the overall described behaviour.

The following step is the establishment of formal collaborations which requires a more complex configuration protocol. The reason for this is the fact that different Equipment Module Agents have different beliefs on what is the best solution, which leads them to making different decisions. In this step the agents are required to establish formal collaborations which are based in their internal assessment model. The problem with these decisions is that according to the model, agents only refuse formal collaborations if their collaboration quota has been fulfilled, as defined in **chapter 5**. Therefore, opposite decisions might lead to stalemate situations where the agents are in an infinite wait state. **Figure 6.2** provides an example of this where agents A, B and C want to select conflicting collaborations, therefore creating a stalemate situation as described above.

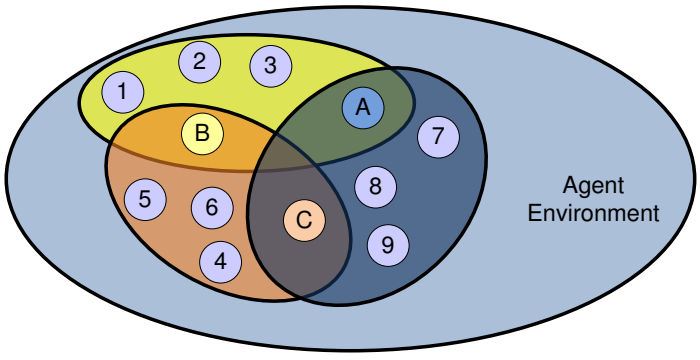


Figure 6.2 - Example of Stalemate Situation

To avoid stalemate situations the **Establish Formal Collaboration** protocol has a timer. The timer period reflects the valuation that each individual equipment module agent has of a given configuration solution. The idea is that the higher the valuation the more time the agent is willing to wait for it. This is important for the protocol because the Equipment Module Agent drops the formal collaboration request, and it has to inform all involved members through a cancellation message. Besides this, there are two other outcomes for this protocol to finish, either all have accepted the formal collaboration or in the case of one rejection this collaboration is dropped. The content of the exchanged messages requires only the ID of the configuration.

Once the configurations are established, the equipment module agents proceed to the validation phase. In this phase an **Expert Validation Request** protocol is required for the interactions with the MAS expert agent. The information that needs to be sent to this agent consists on the configuration description, which follows the model presented in **chapter 4**. Similarly to the previous defined protocols, this is a request where an answer is mandatory. The answer might confirm the validity and completeness of the solution or provide a new set of requirements, which again will follow the models presented in **chapter 4**.

The creation and deployment of performance simulation agents is quite straightforward, and the creation process allows for the transferring of a lot of the information required. However, because these agents were designed to be reused, the connections that it establishes are sent in the **Request for Simulation** protocol. It is important to note that only the owner of the agent can request this, therefore if other agents try to request a simulation the agent will return an invalid request answer. So the trigger for this protocol is a request by the equipment module agent to the performance characteristic agent that contains the connectivity information. The answer for this request is either the failure to perform simulation, which will occur when a performance simulation agent cannot be reached, or the results for simulation. The content for these replies use either the results of the simulation or provides the information of which agent or agents could not be reached to perform the simulation.

The kill order is processed in a separate protocol to ensure the clear separation between the different actions. This protocol is named **Kill Order**, which consists of an order type message that returns a confirmation.

The final protocol that detailed interactions between different equipment module agents is the **Establish Unique Collaboration** protocol. This protocol follows the same approach as the one previously described to establish formal collaborations to avoid stalemate situations. Therefore this protocol is triggered by a request for a unique collaboration which merely needs to provide the ID of the configuration solution. The reply might be a rejection or an acceptance, with the safeguard of the possibility of the cancelation of the request. The detailed sequence diagrams for the **Equipment Module Agents** where these protocols are invoked can be found in the **Appendix B**.

6.3 Agent Methods to Enable Self-Configuration of Modular Assembly Systems

The proposed methodology focuses on the concept of distributed decision making. The hypothesis is that very simple rules distributed across different agents can produce valid and optimal MAS configurations. Therefore it is crucial to define and understand the simple rules that will enable the distributed decision making process.

The individual strategies of each of the agents in the proposed multi agent environment will provide these rules. The majority of the proposed agents are facilitators that provide extra information for the decision making processes of the equipment module agent. The configuration solutions will be assessed by the requirements agent, who is responsible for selecting the best configuration based on the inputs from the system integrator.

The proposed configuration method works in two stages, the first is a logical matching between the requirements and the agent capabilities. This means that agents only show interest in configuration requirements for which they can contribute. The MAS expert agent acts as a configuration expert who can add more requirements based on existing configuration patterns internal to them. This means that the requirements might be enhanced during the configuration process, which will

enhance the number of logical matches between the requirements and the agent capabilities.

The second stage of the configuration method is based on the assembly system key attributes, namely cost, time, quality (repeatability and accuracy) and flexibility. The attributes need to be combined to assess the results for considered configuration possibilities. This task is straightforward for attributes that have constant values, however for variable values a simulation method was built as part of the configuration method to achieve better results. The Performance Simulation Agent is responsible for this task and therefore will require methods to enable the simulation of these attributes.

Once all the attributes are combined they can be compared across different configuration solutions, however due to their diversity they cannot be directly compared with each other. It would not be possible to directly compare a number that is usually high, like cost, with a number that is usually quite low, like repeatability. Nevertheless, in order to compare different configuration solutions it is quite important to take into account the different attributes. Although it is clear that all aspects will contribute for the decision making process, the question that arises is in what way? In agent technology a quite common negotiation method is the presence of some sort of currency, which allows the agents to clearly assess the value of different offers. This approach has been used extensively in the presence of multi variable decision making, which is the case of the configuration method. Therefore it is proposed that the configuration method will use a currency system for which the currency will be the ultimate configuration value.

The calculation of the ultimate configuration value requires that all values are joined together. However, as it was previously stated, these values have quite different scales. Therefore there is a need to normalize these values before progressing to the calculation of the ultimate configuration value.

There are two aspects that provide uncertainty to the agent environment solutions, one is the presence of a weight matrix that is provided and adjusted by the module supplier. This was introduced so that the module supplier has a form of participating in the decision making process. However it is expected that these values will change from what it is established when the module is described. Therefore it is proposed to

include a self adjusting mechanism to the agents that does not interfere with the weight matrix established by the module supplier. This will happen in the normalization functions which will allow parameter change to adjust normalizations based on successful configurations.

These aspects will be described within this sub chapter, and it is hypothesized that the combination of these in the proposed agent environment will result in the self configuration methodology.

6.3.1 Performance Characteristics for Modular Assembly Systems

The strategic attributes for assembly systems are provided in the literature as cost, time, quality (repeatability and accuracy) and flexibility (Chryssolouris [14]). MAS are a subset of assembly systems and therefore the same attributes are important.

The proposed model in **chapter 4** defines the agent environment inputs, which contains the definition for each of these assembly system attributes. In this chapter these will be used for normalization.

6.3.2 Mathematical Normalization of Performance Characteristics

The normalization of the attributes is a key aspect in the configuration method. The diversity of the attribute types would render any combination of values impossible before normalization. This is one of the issues that highlight the importance of normalization in order to make decision in the configuration process. However the proposed solution target also agent's self-adaptation based on the past configuration results. Therefore there are two aspects for the normalization, the mathematical functions that normalize the values, and how these functions can be adjusted over time.

The first step in normalizing a value is to understand its source and type. Some values should be maximized and others minimized, this distinction needs to be clear before establishing any normalization method. The source and type of the attributes considered provide a clear view on which values are intended to be as low as possible, or the reverse. In the proposed method both types of values are present,

time, cost, repeatability and accuracy are values that we want to minimize, while flexibility (defined as spare capabilities) should be maximized.

The values intended for minimization have straightforward normalization limits. The lower limit of the normalization is zero since it is the utopian value, meaning it is the best value achievable but it is not very likely. The upper limit is the provided by the modular assembly system requirements, since the provided value has been defined as the maximum possible for this type of attribute. Therefore, the normalization function will have the lower limit (Ll) and the upper limit (Lu) as its first parameters.

The defined limits allow for an easy normalization if the mathematical function is defined. Using the two, it is quite straightforward and common to use a mathematical function type, e.g. a linear function or an exponential function. However, the use of one of these functions would allow for a unique self adjusting function that can be transformed as shown in **Figure 6.3**. This provides the ability to have more realist normalization, which can be adjusted over time.

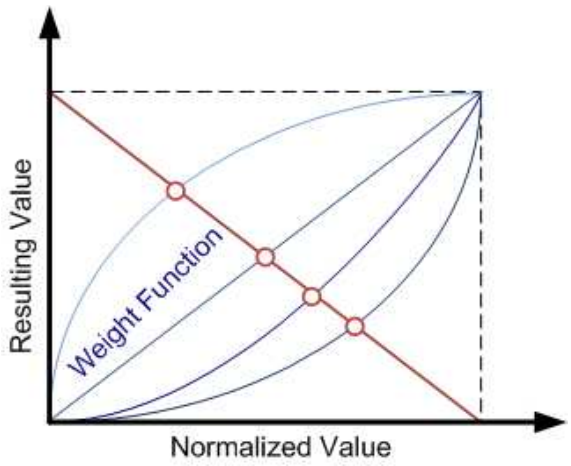


Figure 6.3 - Conceptual Assembly Characteristics Variation

The proposed method entails a function that is transformable in terms of concavity. This enables the adjustment of agent's normalization function which provides the ability for the agent to adjust their beliefs in each assembly characteristic. To achieve this, an exponential function can, in the limit, be transformed into a linear function. However, to enable the concavity to be regulated based on parameters, it is suggested the use of a polynomial function of the second degree. The choice of such mathematical function is supported by the existing parameters. The first two parameters establish the limits of the normalization function, whereas the other two

parameters are an intermediate point in the normalization function which should have a specific normalized value.

The introduction of these two parameters is based on the analysis of the attributes. The idea is that values that need to be minimised will never reach zero. For example, cycle time can be reduced to a minimum but it will never reach zero. If a linear function was used, it would provide linear normalization values. This would result in a progressive conversion of the cycle time, which would rate nearly impossible improvements, like near zero cycle times, the same way as reductions near the established requirements, which are more likely.

On the other hand, the use of an exponential function would resolve this issue, providing the possibility of having a function that could be adjusted to a limit that would make it linear. However it would not allow the change of rate for its inverse based on parameter change and at its limit it could become a linear function. Moreover, the cost attribute in an initial stage is of the same nature as the cycle time and it is not likely, when configuring the first system, that modules have a cost of zero. However, it is important to note that in the case of a reconfiguration of a system it is probable that the module cost will be close to zero in certain situations. Therefore it is proposed that the normalization function should be adjustable to rate values depending on the evolution of the configuration choices.

The simplest way to define the behaviour described above is by setting an intermediate point for which the mathematical function needs to go to. This point also makes sense for a simple early definition, since one can say that at the midpoint, the function should be valued at 50%, or 80% depending on the type. This point also enables the readjustment of the function, providing two variables, one on each axis. The final restriction of the normalization function is the need for it to be strictly descending, which can be guaranteed by its derivative being zero which would mean no inflexions exist. In sum, the requirements for this mathematical function are as follows:

$$f(Ll) = 1 \xRightarrow[Simplification]{} Ll = 0$$

$$f(Lu) = 0$$

$f[j * Lu] = i\ , where\ 0 < j < 1\ and\ 0 < i < 1$

$f'(x) = 0, for\ Ll < x < Lu$

The following equation provides the result of this normalization function while the demonstration can be found in **Appendix C**.

$$f(x) = 1 - \frac{(-1 + i + j^2)x}{(-1 + j)j(Lu)} - \frac{(1 - i - j)x^2}{(-1 + j)j(Lu)^2}$$

6.3.3 Formulation of Mathematical Beliefs Readjustment due to Failed Collaborations

The rationale behind the need for a readjustment of the normalization function is, to embed in the agents the capability to adjust their beliefs based on their success or failure. The idea is that the normalization functions can provide insight into trends that are impossible to predetermine. Towards that end the normalization function previously described has a variable point that enables its behaviour to be readjusted. However, there are limitations for this readjustment. A detailed analysis of the mathematical function shows that it only guarantees the required characteristics if this point is placed under a certain area. The problem lies in the imposition of the derivative being zero, which requires the regulation point to fall under a restrictive set of conditions. **Figure 6.4** provides the graphical spectrum for which it is valid to adjust the point as well as the mathematical formulation that defines this area. The demonstration of this area can be found in **Appendix C**. Once the limitation is considered, it is possible to establish a method for readjusting this point based on failures or successes. The proposed solution is that the point is readjusted vertically until it reaches a lower limit, in which case it is adjusted horizontally to re-shift the working space for readjustment as shown in **Figure 6.4**.

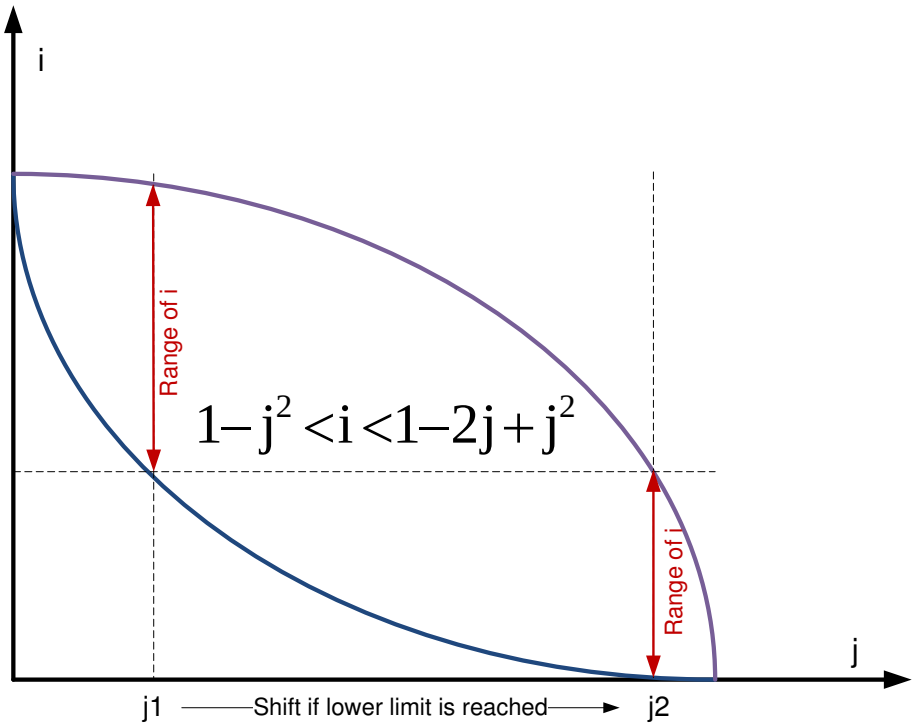


Figure 6.4 - Graphical Illustration of Operational and Respective Spectrum

The ability to adjust the agent beliefs on the different characteristics of the MAS enables the definition of an internal method that allows for the adjustment of beliefs over time supported by success or failure of proposed solutions. This enables the agents to follow trends in the configuration solutions simply by participating in potential solutions.

6.3.4 Requirements Agent Operational Strategy

The Requirements Agent has two major operational roles, the advertisement of requirements and the final ranking of the found solutions. These are supported by minor operational roles, namely the constant ability to update the system integrator on current state of the configuration methodology state and the feedback to the **Equipment Modules Agents** of the system integrator decision.

In operational terms the minor roles are quite straightforward and do not require detailed descriptions on the decision making process of the **Requirements Agent**. These are information tasks triggered by the system integrator. On the other hand, the major operation roles of the **Requirements Agent** require the establishment of clear rules that enable it to make the necessary decisions for the operation of the self-configuration methodology.

The broadcast of requirements might seem straightforward, however the broadcast targets need to be available somewhere, considering that some **Equipment Module Agents** will be running across different agent platforms. Therefore **the Requirements Agents** need to retrieve a list of available agents for the broadcast. The use of a yellow page service has been extensively used in the literature to solve similar problems, which would simply require all **Equipment Module Agent** to register a couple of their attributes (Sugumaran [124]). Despite the availability of standard yellow page services in agent platforms, these have a quite significant restriction in the amount of results they provide. This is a serious problem for the future scalability of the self-configuration methodology and therefore needs to be addressed in this work. Therefore, this work introduces the creation of a yellow page service that has no restriction on the number of results it provides. The service takes the form of an agent that will have a known location to all agents that take part in the self-configuration methodology. All agents will register with it, and it will provide the list of available agents to those who require it. Because this is a technical adjustment to the agent platform this was not included in the agent model, but it is important to mention it to understand the source of information for the **Requirements Agent**.

The other important operational aspect of the **Requirements Agent** is the ranking of the found solutions. To rank the solution this agent uses the information contained in the requirements definitions, namely in the assembly system targets. The assembly system targets define both the overall targets and their importance. The ranking of found solutions will use the importance of the values against the results of the solutions and determine a value. This operation is the same as the one performed by the equipment module agents to establish their final ranking; the only difference lies in the different weighting of the solutions. The mathematical formulation of this operation is described in the equipment module agent collaboration assessment.

Once all values are calculated, the highest valued solutions are presented for the system integrator and he will choose the solution that is more suitable according to its knowledge.

6.3.5 Equipment Module Agent Operational Strategy

The **Equipment Module Agent** is the key player in the proposed self-configuration methodology. It is the agent that is ultimately responsible to find configurations' solutions. In **chapter 5** this agent has been broken down into operational states, which require a series of operational assessments that will be described in this chapter. This chapter will be broken down into subsections that will focus on the major assessments, while the minor decisions making rules will be explained briefly.

The first decision point for the **Equipment Module Agent** occurs upon the arrival of new requirements. In that situation that agent will execute an assessment of the requirements and decide on whether or not it is interested. This assessment will be covered in this subchapter.

In the event of a positive interest in the requirements, the agent needs to start communicating with other **Equipment Module Agents** to identify possible collaboration targets. This implies that the agent needs to query other agents about their interest to collaborate with them. Once the targets are identified the **Equipment Module Agent** needs to assess if it has enough collaboration to establish a solution that can fulfil the established requirements.

The following operational step of the equipment module agent is to assess each potential solution, in the collaboration assessment. This will be explained in the relevant subchapter.

Once configurations are established and evaluated, the MAS expert agent is contacted for extra inputs to the configurations. The results of this assessment do not require any extra reasoning from the module agent, since it can simply create additional requirements that will trigger the prior processes.

The next operational requirement of the equipment module agent is the deployment of performance simulation agents. This is quite straightforward task, since the agent already possesses all the necessary information to create these agents. The information is extracted from the module description, where all the capabilities of the module are present. Each of these capabilities that are involved in the given solution, will represent the need for performance simulation agent. On creation, the equipment module agent needs to provide all the attributes for the given capability, not just the

capability type. The equipment module agent also needs to relay the configuration solution so that the performance simulation agents can establish a virtual network that represents the configuration.

The reception of simulation results triggers the next operational step of the equipment module agent. Upon arrival of all expected results, the agent will proceed to select the configuration most advantageous according to the collaboration selection assessment, which is performed through the established mathematical model for the agent's beliefs.

The agreement and subsequent submission of a configuration solution follows the simple logic of using the highest ranked solutions based on the simulation results. This is followed by waiting for the final selection results so that the agent can update their internal models for decision making.

6.3.5.1. Expression of Interest

The identification of interest in the requirements provided by the requirements agent follows a very simple set of rules. The basic concept is whether the Equipment Module Agent can execute any of the given set of capabilities. If it can, its interest is established. However this is the simplification of the problem since the agent might have multiple interests in the requirements. It is simple to understand that a robot might be interested in multiple handling tasks, but it is also clear that the same robot cannot be involved in handling tasks across different workstations.

The question that arises is how the Equipment Module Agent identifies that assembly processes are in different workstations. The answer is in the requirements descriptions. For the purpose of this work, Equipment Module Agents can only be interested in multiple capabilities if these are related to the same workstation in the requirements definition.

The equipment module agent will manage internally the multiple interests that it has based on the given set of requirements. It will actuate each of these interests as parallel interest, and in the end select the one that gives it the best value for being selected. This means that the agent will have interests in multiple workstations, but also it will create alternatives for executing one capability, two capabilities, or whatever number of capabilities. All these will be considered alternatives that the

agent needs to maintain internally. **Figure 6.5** provides an example of a set of requirements, which is composed by two workstations in which a given conceptual module has interests. In this example the equipment module agent would create four parallel configuration processes as described in the **Figure 6.5**.

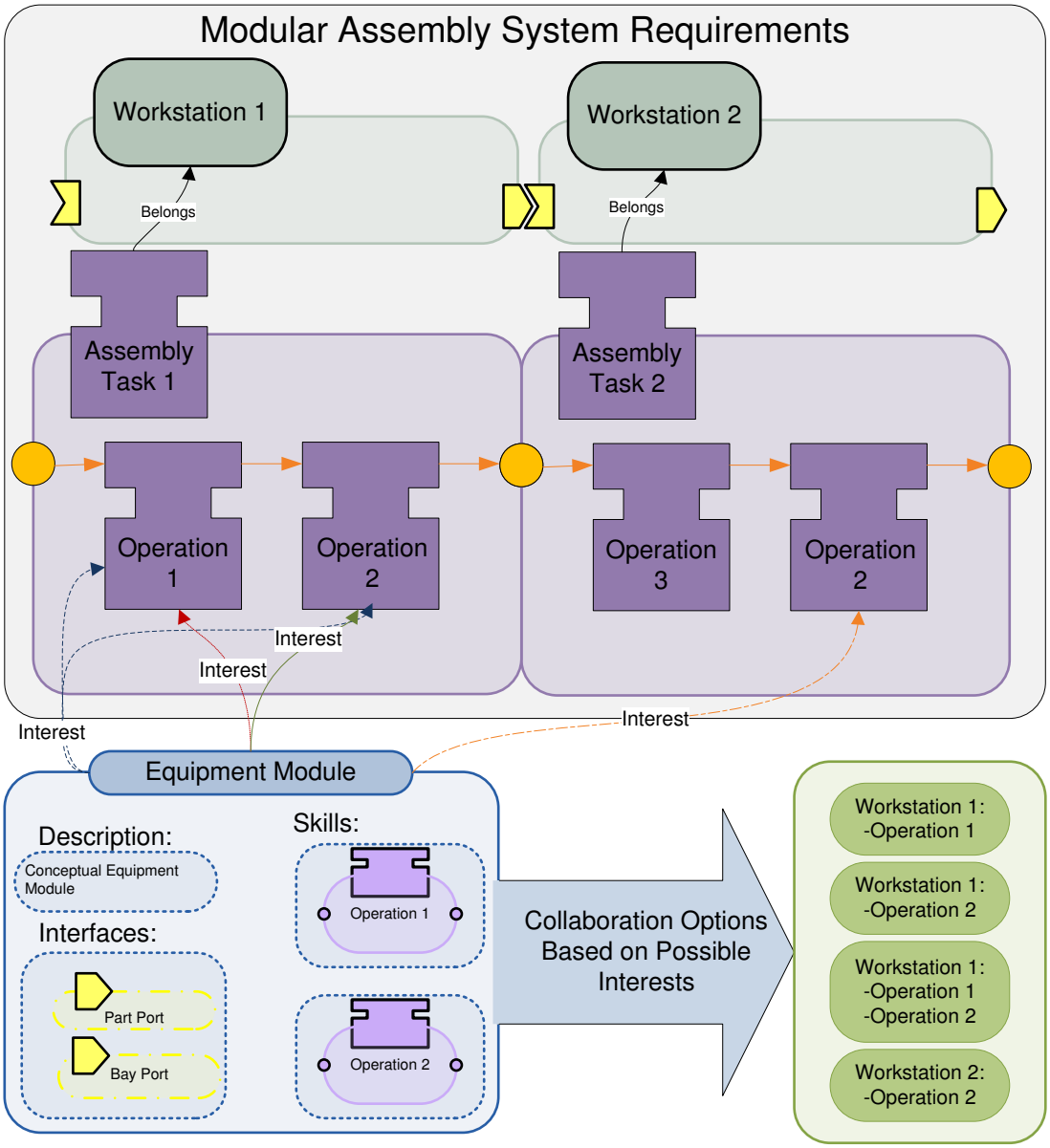


Figure 6.5 - Conceptual Example for Multiple Interests in Given Set of Requirements

The identification of the agent interest in a given set of requirements is the first step in the expression of interest process. This is followed by the need for identification of potential collaboration partners. This implies that the agent needs to query other agents about their interest to collaborate with them. The question is how this query should be executed, and more importantly what is it about. The decision on whether or not to collaborate is based on having something to gain, thus it is logical to

construct a query on the capabilities that the agent cannot execute. Instead of defining a specific model to exchange information for this collaboration, it is proposed that the agent uses the model for requirements that is already established and which all of them already understand. The agent will simply update the requirements based on its capabilities, namely by removing them and stripping down all other aspects from the requirements. This exchange will follow a known protocol, so that agents can make a distinction between these requirements and the ones provided by the **Requirements Agent**. It is important to highlight that the decision to show interest is only made through the established capabilities, all other aspects will be considered during the later stages of the methodology.

The defined method to identify collaboration targets poses a question highlighted on the previous chapter, which is the issue of scalability. If there are a high number of **Equipment Module Agents**, and if they are allowed to establish as many collaboration targets as they deem fit, this will result in high computational resource consumption. Therefore it was proposed in the previous chapter that a limitation should be introduced, so that one can test what would be the optimal number of configurations that should be allowed. In operational terms the algorithm works pretty much in the same way, but it caters for a limitation on the number of possible collaboration targets that can be identified. The only difference is the need for the introduction of some randomization of the potential collaborators list. Otherwise the agents would all contact primarily the same agents which would make certain agents more important than others.

Once the targets are identified the Equipment Module Agent needs to assess if it has enough collaboration to tackle the established requirements. If it is not the case, the agent will contact more agents for collaboration, and repeat the process. The algorithm that executes the above described operations is presented in **Figure 6.6**.

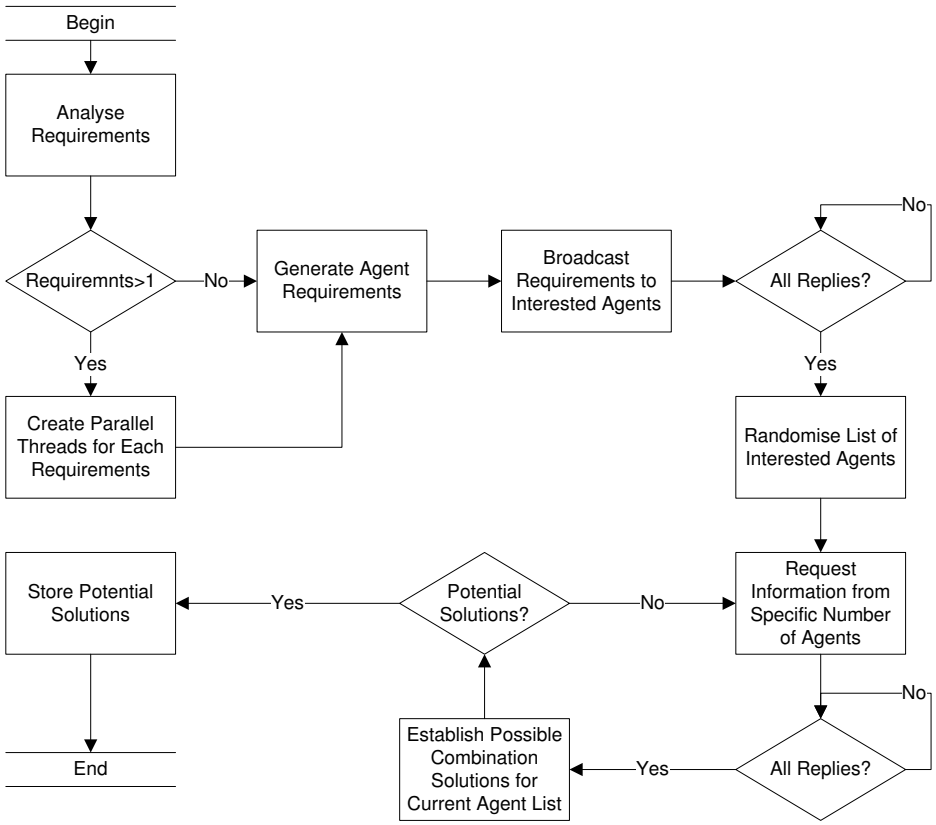


Figure 6.6 - Expression of Interest Decision Making Process

The presented algorithm provides the agent with all possible configuration solutions based on its network of collaboration targets. Once these are established, the agent can proceed to the assessment of each possibility, in the collaboration assessment.

6.3.5.2. Collaboration Assessment

The collaboration assessment is executed for all viable configurations. A viable configuration is defined as a configuration that fulfils all the established requirements. Therefore the collaboration assessment's first operation is to identify which collaborations are viable. However this operation is not straightforward since the equipment module agent only possessed information on other equipment module agents and not on specific solutions. Therefore the Equipment Module Agent needs first to establish the possible solutions based on the information it possesses in order to assess the number of viable configurations. If no viable configurations are found, the equipment module agent needs to find more agents to collaborate as defined in the previous chapter. The procedure for establishing configuration solutions takes into account what was defined in the expression of interest, therefore it uses those rules in addition to the internal management method described in **Figure 6.7**. In a

nutshell, this method enables each agent to maintain a potential solution table based on the information it collects from other equipment module agents. This process is finalised when all the information is acquired, which triggers the suppression of the incomplete solutions from the internal table.

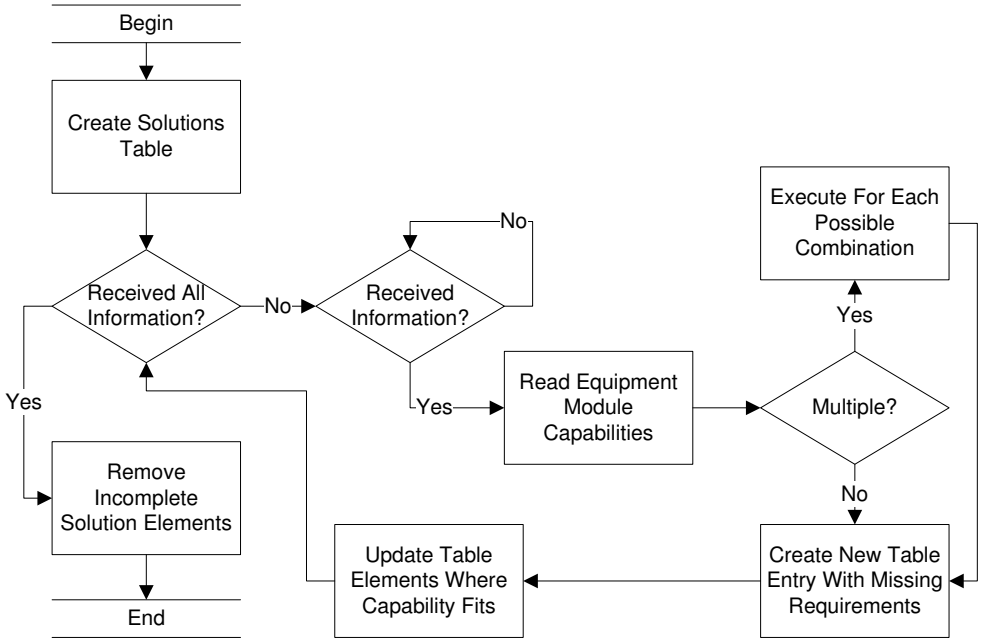


Figure 6.7 – Equipment Module Agent Collaboration Management Method

Once the set of viable configurations is determined these need to be ranked. The ranking of configuration solutions is based on the assembly performance characteristics. However this method is not a simple adding, since some of the capabilities affect the assembly performance characteristics differently. For example, capabilities that occur in parallel will affect cycle time differently then capabilities that occur in sequence. Therefore a clear set of rules needs to be identified in order for this assessment to take place.

To establish the rules, an analysis is required to understand the assembly performance characteristics. Flexibility is the simplest characteristic, since it is defined as additional capabilities. Therefore, it is obtained by simply adding the spare capacities of any given solution. Cost is also very straightforward since it is based on the equipment module cost. Additional considerations on assembly processes cost will be considered by the simulation agent.

Cycle time requires the consideration of capabilities that occur in parallel and in sequence. The rule is if assembly processes occur in sequence that cycle time is

simply added, however if assembly processes happen in parallel then the highest value is used and the lowest is disregarded.

Finally, accuracy and repeatability will use the classification established in chapter 4 for assessing assembly process type. The type will determine if the values should be added, replaced, or fixated. **Figure 6.8** provides the algorithm for the method that enforces these rules.

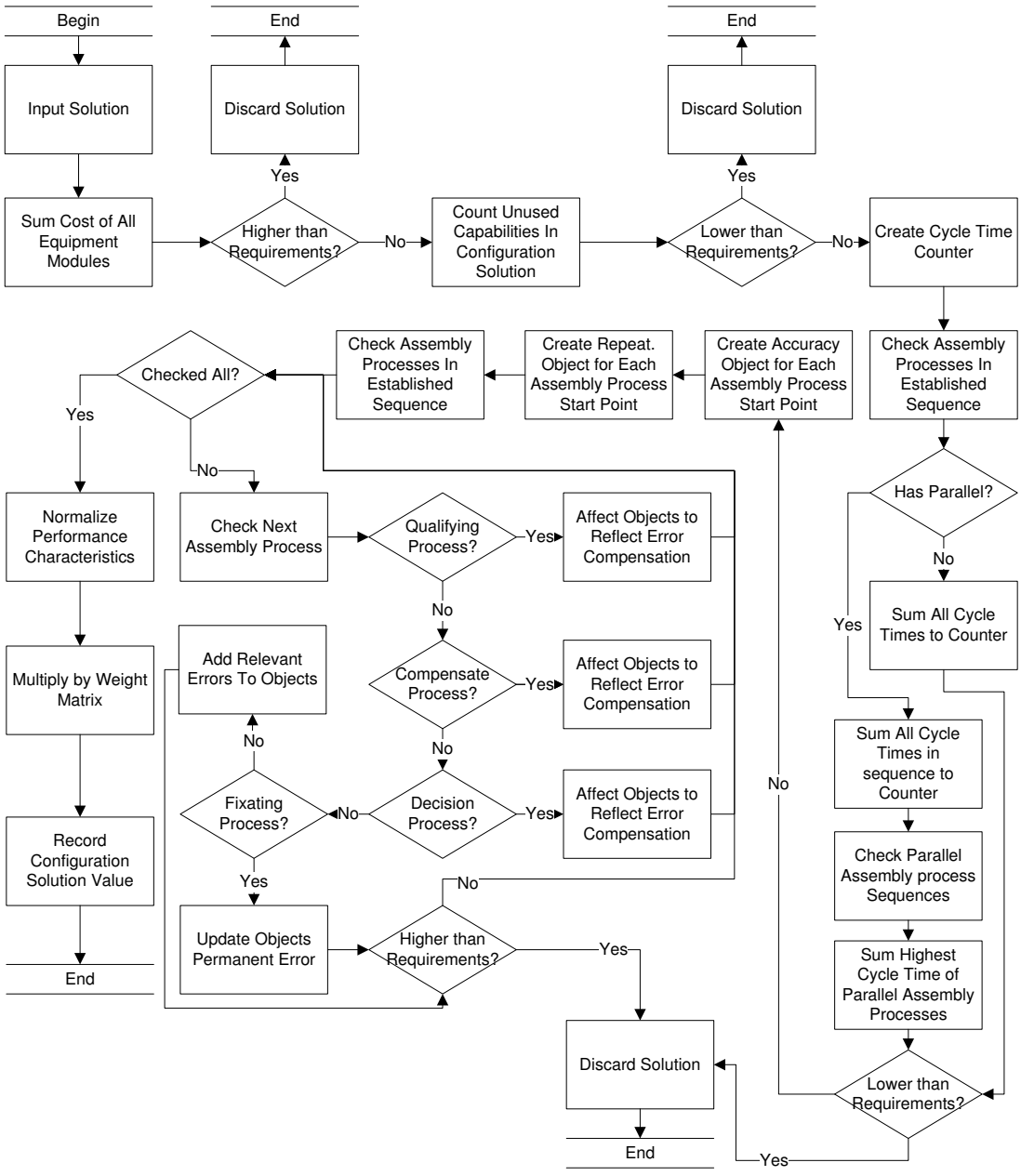


Figure 6.8 - MAS Configuration Assessment Method

The proposed method is performed for all configuration solutions, going through the assembly characteristics, namely cost, flexibility, cycle time, repeatability and accuracy, if any of these is outside of the requirements the solution is discarded. The method is used for the calculation of the assembly characteristics of configuration solutions, which will enable the decision making capabilities of equipment module agents. Once all values are determined for each configuration these will be normalized and valued against the internal weight matrix, which will result in ranking index value.

6.3.5.3. Collaboration Selection

The collaboration selection comprises the assessment of the results obtained from the performance simulation agents. These agents will produce results based on Monte Carlo simulation, which results in a series of results that cannot be directly used. This means that some treatment of the result is required, meaning that some rules need to be defined for this.

The first rule says that simulations cannot, at any point, conflict with the requirements. The idea behind this rule is to guarantee that even in the worst conditions the requirements are always met. The elicitation of this rule is quite straightforward since it is just a comparison between the worst values and the requirements.

Once it is guaranteed that the configurations' solutions are not in conflict with the requirements, an assessment needs to be made on the several simulation results. The first step is to normalize the results to be able to deal with the different characteristics in a uniform way. Once the values are normalized an average value is determined. The average value provides the central value for assessing the configuration solution.

Nevertheless, the simulation results produce more insight into the configuration solution than the mere average value. In fact these raw data allow for the analysis of the standard deviation of the results. This determines the stability of the achieved average values. The impact of this result for the decision making process in question can be described as the entropy of the assembly system, which by definition is the disorder in the system (Chang [133]). **Figure 6.9** provides an overview of the

standard deviation values, which accounts for 68% of the samples, where in blue one obtains more entropic result which might be within the establish requirements.

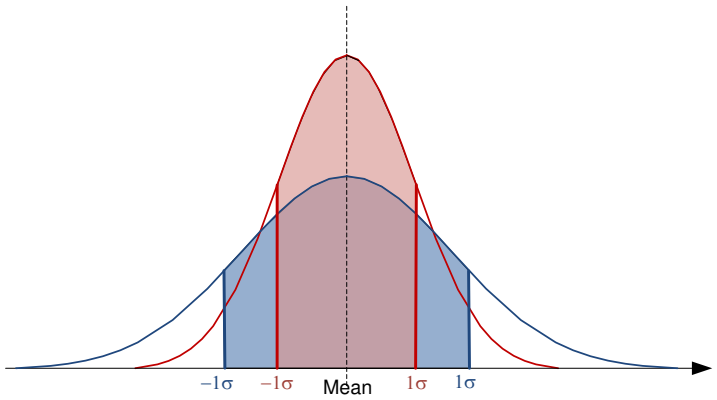


Figure 6.9 - Standard Deviation Example

It is clear that this entropy has a relevant impact because the solution would vary much more. It is also fairly straightforward to think that disorder should be penalised; the question is by what measure. This is a question that cannot be answered by any one individually because it depends on several aspects which are attributed to the sensibility of the system integrators and module vendors. Therefore the configuration methodology has defined this as an input value, where the different module vendors can establish the weight of this disorder in the decision making process of their **Equipment Module Agent**. Similarly, the system integrator will define this weight for the **Requirements Agent** assessment of the proposed solutions.

In sum, the final assessment can be described by the process of normalizing the results, getting the average value, getting the standard deviation of those values and weight those based on the internal weights that each **Equipment Module Agent** possesses. The following equations provide the mathematical formulation for determining the final configuration solution value, which is the decision factor for ranking the solutions.

$$Mean.Value = \begin{bmatrix} Normalised(Repeatability.Mean) \\ Normalised(Accuracy.Mean) \\ Normalised(Cost.Mean) \\ Normalised(Time.Mean) \\ Normalised(Flexibility.Mean) \end{bmatrix} \begin{bmatrix} Repeatability.Weight \\ Accuracy.Weight \\ Cost.Weight \\ Time.Weight \\ Flexibility.Weight \end{bmatrix}^T$$

$$StdDeviation.Value = \begin{bmatrix} Normalised(Repeatability.StdDeviation) \\ Normalised(Accuracy.StdDeviation) \\ Normalised(Cost.StdDeviation) \\ Normalised(Time.StdDeviation) \\ Normalised(Flexibility.StdDeviation) \end{bmatrix} \begin{bmatrix} Repeatability.Weight \\ Accuracy.Weight \\ Cost.Weight \\ Time.Weight \\ Flexibility.Weight \end{bmatrix}^T$$

$$Final.Value = Mean.Value * Mean.Weight + StdDeviation.Value * StdDeviation.Weight$$

The final step of the collaboration selection is simply to choose the highest ranked one.

6.3.6 Performance Simulation Agent Operational Strategy

The Performance Simulation Agent will execute the simulation of a given configuration. To achieve this goal, two main operational states need to exist, namely the establishment of the simulation model that represents the configuration solution and the execution of the simulation, as defined in **chapter 5**. To that end it is proposed the use of the syntheses model presented in (Ferreira et al. [128]), extending it to cater for all performance characteristics. It recognises that the assembly process accuracy and repeatability, of an assembly system depends upon two aspects; the physical arrangement of different pieces of equipment and the logical sequence of operations which they need to jointly execute to achieve their common assembly objective. Furthermore, the same model can be extended to cater for identified performance characteristics. **Figure 6.10** provides an overview of this model.

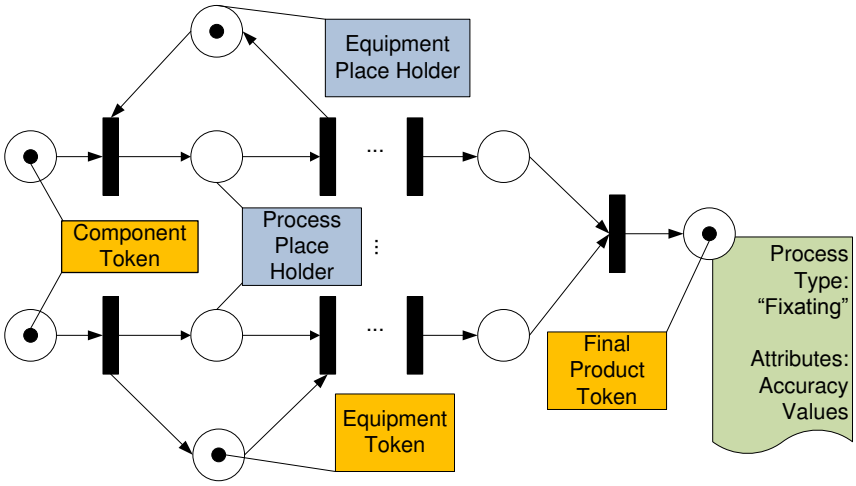


Figure 6.10- Token flow description for an example of MAS

The use of a Petri net based model has been adopted which allows the use of different token types for components and equipment that are propagated throughout the assembly process chain. Essentially, a token is being created for each component which is being assembled. Component tokens are merged into a product token when the assembly process is of "Fixating" type. However this does not take into account the modules that are responsible for the assembly processes. These are represented through module specific tokens which carry the repeatability properties of the equipment. These are merged into the component token when another module takes responsibility for the component or a "Fixating" type process occurs.

It is proposed that **Performance Simulation Agents** emulate this behaviour through the exchange of messages that contain structured information on equipment tokens and assembly process tokens. This information consists of updated objects for each of the performance characteristics, which represent the assembly process tokens, in conjunction with the last agent that effected the equipment token and the equipment ID. This way, once the equipment is different, the agent can simply give back the equipment token to the relevant agent. Therefore the agent is able to perform the two place holder roles defined in the model.

In addition, the agent is also responsible to perform the necessary operations for the transition that precedes it, this in effect ensures the emulation of the **Petri Net** model. Each **Equipment Module Agent** will deploy the required **Performance Simulation Agents** for a given solution with the specific information on the assembly process that is executing, plus the equipment responsible for it. In addition to this, the

Equipment Module Agents also need to provide the required connection for the execution of the simulation, so that it provides the individual connections that each **Performance Simulation Agent** needs to establish. This provides a straightforward manner to establish the virtual configuration which enables the behaviour model of a given solution.

Once the behaviour model has been synthesised, it enables the simulation of the underlying system behaviour based on token passing approach. An unaltered Petri net, however, does not provide the desired behaviour characteristics and requires a more specific definition of how the tokens behave in the model through the established place holders and transitions.

The transitions are responsible for the management of the tokens, making sure tokens exiting the flow are incorporated into the component token. This is its basic behaviour if a token is exiting the flow this needs to be passed on to the component token, otherwise the tokens are simply passed to the next process place holder. **Figure 6.11** describes the operational behaviour of the transitions presented in the model. This diagram details how the different performance characteristics are processed in the transitions in order to calculate the results.

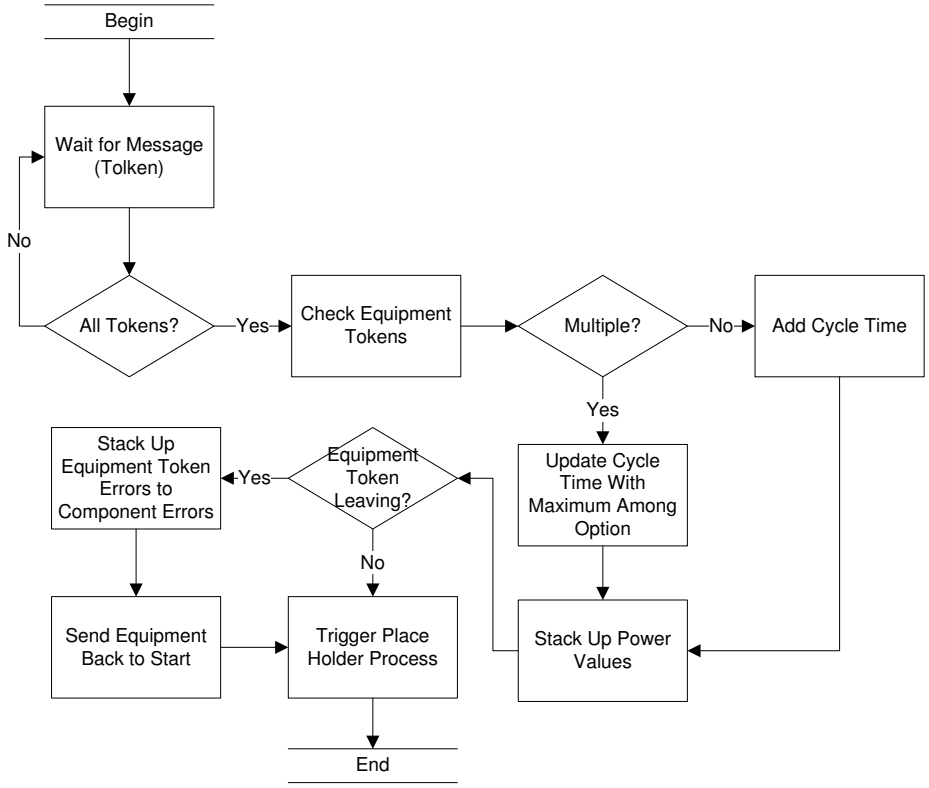


Figure 6.11 - Transition Behaviour Algorithm

The process place holders are assembly process specific and thus they incorporate the process classification, as defined in **chapter 4**. Their behaviour is driven by the assembly processes classification which serves as input to establish how to affect the tokens, namely in the relation to the assembly system errors. **Figure 6.12** describes the process place holder behaviour where it is clearly defined how this model should react to the different process types. The behaviour of the place holders will be broken into the four performance simulation characteristics which will require different actions. The repeatability and accuracy performance characteristics can be broken down into three types: one for Qualifying, Decision and Compensate processes which affect the equipment token compensate error matrix which will be incorporated into the component token once the equipment token leaves the system. The other type is the fixating processes which merge all tokens present into a new component type token (or final product). The final type is for any other process types which simply stack up the equipment token with the relevant error. It is important to note that only when the equipment token leaves the system will the stack up of component token errors occur. The power consumption has no compensation possibility and therefore it is simply stacking the value that all the assembly processes are consuming in the simulation, which is treated in the transitions algorithm. The cycle time uses a different approach for the merger of the cycle time values, if two tokens are to be merged, therefore coming from two different sources the value that is set is the highest, which again is treated in the transitions algorithm. Therefore the place holders operate only on the precision aspects as seen in **Figure 6.12**.

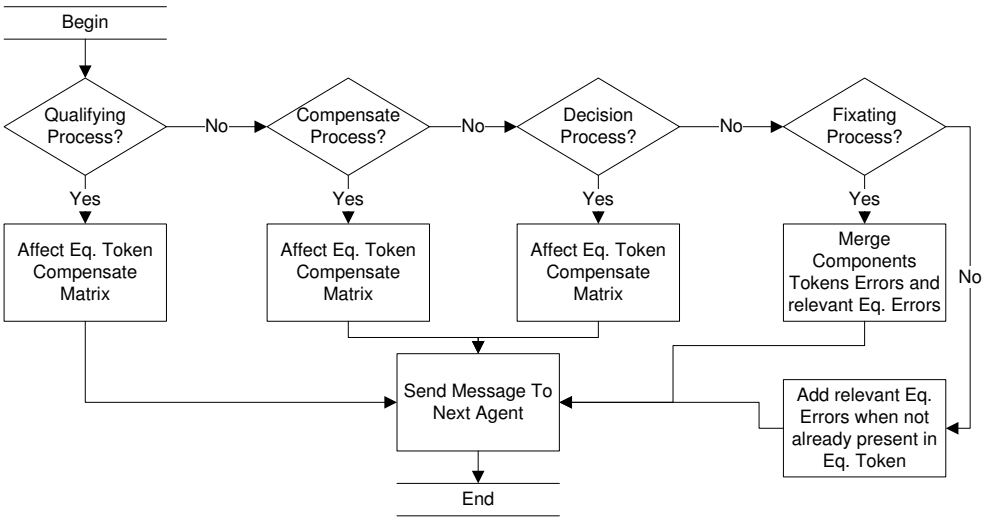


Figure 6.12 - Place Holder Behaviour Algorithm

The synthesis algorithm for the precision characteristics is based on a state transition approach which is used to construct a 3D parametric model using 4x4 matrix transformations of all contribution factors and sources of errors leading up to and during the completion of a full assembly process. The algorithm distinguishes between processes that contribute to the error, those that do not and those that compensate errors from previous operations. Each Module/Skill can contribute in 6 Degrees of Freedom to the assembly error of the workstation (3 translations and 3 rotations). Each error is expressed by its variation (upper and lower bound) and the accuracy of the error value (3 or 6 sigma) which is provided in the equipment module description. The synthesis algorithm for the remaining characteristics is less complex since it is a simple value, so no matrix is required, yet the overall behaviour is the same as the synthesis algorithm for the precision characteristics.

6.3.7 MAS Expert Agent Operational Strategy

The **MAS Expert Agent** is defined as the expert of MAS configuration and performs two assessments in the configuration process: the expert configuration assessment and the performance failure assessment which were defined in **chapter 5**. Despite this division, the internal operation of the **MAS Expert Agent** is quite similar, since it is based on the existence of patterns and rules for both assembly process configuration and physical system configuration. Therefore any assessment of the **MAS Expert Agent** covers two sub assessments, the assembly process assessment and the physical assessment. As it was stated before, this agent only contains a lightweight set of rules that demonstrates the agent potential in the configuration methodology once more knowledge can be acquired and incorporated into the agent.

The operational behaviour of the **MAS Expert Agent** for the expert configuration assessment firstly looks at the completeness of the given solution. The completeness of a solution assessment is performed in phases. The first looks at predefined system completeness rules and is followed by the matching with internal patterns for MAS solutions. The decision of looking firstly to the rules resides in the fact that these can provide early insight into missing elements in the solutions using minimal effort. This follows the smallest effort and biggest impact approach, therefore the rules act as the first tier for the completeness assessment. The rules have to be absolute and

cannot have any conflicts, while the patterns provide the means to have parallel solutions.

The rules for the completeness assessment look at both physical aspects and assembly processes aspects. A set of very basic rules for completeness assessment are proposed as follows to provide an overview of the impact that these might have in the configuration methodology:

- Incomplete physical interfaces – The existence of non plugged physical ports that are not indicated as optional requires the establishment of requirements based on the global interface definitions where the matching port pair or pairs are defined.
- Incomplete assembly process parameter interfaces – The existence of mandatory parameters for an assembly process that are not connected due to the absent matching parameter port.

If these rules are not breached, that is, if the solution follows the rules, the MAS Expert Agent performs the matching of existing patterns to the given solution. Because this is viewed as an evolving agent, in the absence of patterns, as in the absence of rules, the agent simply assumes that the solution is valid. The patterns allow for the definition of alternative configuration patterns that indicate what the necessary elements in a configuration are. The ability to have alternatives is crucial because the rational is that the solution needs to follow one of the given set of patterns, and if it does not, the missing element or elements of the closest match should be established as missing requirements. The patterns can be both physically related and assembly process related. **Figure 6.13** provides a conceptual overview of pattern structures that the **MAS Expert Agent** contains. If any pattern exists then the solutions would be required to fulfil at least one of the defined variants.

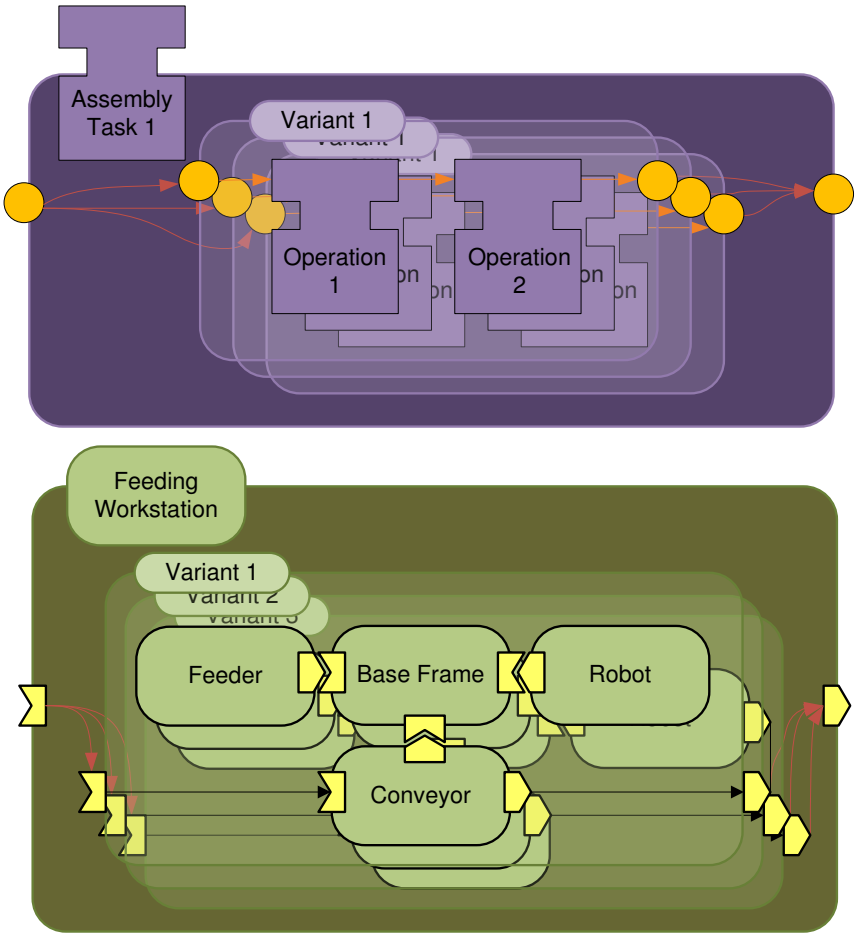


Figure 6.13 - Conceptual Overview of Pattern Structures

It is also proposed that the predefined patterns structure cater for definitions that allow for defining mandatory components of the given variant, while others are merely optional. This characteristic provides the system with the ability for not very complex definition since the only constraint that the **MAS Expert Agent** imposes is the fact that all alternative patterns need to exist otherwise possible valid solutions might be disregarded. The model provided in **chapter 4** can cater for this description with a small enhancement for dealing with variants, thus it is proposed its use.

The other assessment performed by the **MAS Expert Agent** is the perform failure assessment as defined in **chapter 5**. The introduction of this assessment is supported by the possibility that certain failures in solutions can be compensated, namely the MAS repeatability and cycle time. However the form on how to compensate the failure of these is expert knowledge that requires a wide understanding of MAS.

It is proposed that the **MAS Expert Agent** contains a set of rules that indicates how a given failed configuration can be salvaged, or if in fact it is impossible. To

establish these rules one needs to look at the assembly characteristic at fault and find the source of the problem. While for cycle time the source is normally a bottleneck, the repeatability is more of a stacking up problem. Therefore the rules for dealing with these two aspects are quite different and require a separation. So the first thing the **MAS Expert Agent** needs to assess is the type of failure, and verifies if rules exist to compensate for this error. The power consumption which is associated with the running cost of the system and accuracy do not have rules to compensate for it. Nevertheless a future iteration of this agent might provide extra definition of rules for these aspects. For this work it is proposed two sets of rules, one for compensating for cycle time, and another for repeatability.

The proposed rule for dealing with the cycle time failure is quite straightforward, if the bottleneck station cost is inferior to the maximum cost minus current cost, then requirements for a parallel station are formulated, otherwise there is no means for compensating.

The proposed rule for dealing with repeatability needs to look at the type of assembly processes being performed. In **chapter 4** a classification for process types was introduced and it is used for the implementation of this rule. The only way one can compensate for error is before the fixating processes, since after these, the error is permanent. Therefore these processes need to be found to determine which ones produced the biggest error impact to the solution. Once these are identified, the **MAS Expert Agent** establishes new requirements for a qualifying process that should occur before the fixating process with most impact. However this qualifying process needs to occur when compensation can still occur, as such it should be placed before the previous handling process.

In the event of multiple failures, that are both aspects failed, the approach of the **MAS Expert Agent** is to verify cycle time first, since it is the rule that is more likely to not produce compensation options. The rules are internal to the agent, however it is expectable that in the future a rule engine should be incorporated into this agent.

6.4 Reconfiguration of Existing Modular Assembly Systems

The emergent configuration methodology that results from the combination of **chapter 4**, **chapter 5** and **chapter 6** not only provides the means to configure the system but also to reconfigure it. The definition of reconfiguration has been given as the enhanced configuration problem. The only difference between configuring and reconfiguring MAS can be summed up as added constraints (Ferreira et al. [134]).

What this work theorizes, is that if an equipment module is available already it will have zero cost, therefore this will produce a huge impact in the decision-making process leading near zero cost modules to be the best solution the majority of the times. In fact, only when a specific capability is not present in the current system will external modules have a real chance for a participation in a system solution. Even so, if external solutions are better for some reason, the method will use them.

Finally, the possibility of mandatory equipment modules being defined in the requirements definition, provides the system integrator with the tools to ensure that a given set of modules is used. In sum, it is not required to change any aspect in the configuration methodology for it to be able to cater for reconfiguration solutions. The only difference is in the definition of the requirements.

6.5 Chapter Summary

This chapter provides the methods and formal descriptions of the decision making characteristics that enable the emergence of configurations from the proposed agent architecture. It formally describes the required agent protocols which enable agent interaction in the context of MAS configuration. The decision making process methods are proposed, providing an innovative bottom up approach for the establishment of configuration solutions. In this chapter it was also presented an innovative performance simulation model, which can be executed by agents or other technologies, which can cater for the variable characteristics of MAS.

7 Illustration and Validation

7.1 Introduction

In this chapter the application of the proposed methodology will be illustrated and validated. The aims and objectives of this work target a domain that is quite extensive. The validation of this work will focus on a set of representative scenarios that reflect the key problems and characteristics in the domain of MAS configuration. The complete validation of the proposed methodology for the whole domain is outside of the scope of this work.

This work has been broken down into three core contributions that will be validated independently in this chapter given a set of scenarios that will be illustrated also within this chapter. The illustrative scenarios will target the verification and validation of the models and methods while enabling the demonstration of their operation.

The first target of this chapter is the validation of the MAS configuration model that provides the inputs for the self-configuration methodology. The model was embedded into a manual MAS definition and configuration tool, which will be used for its validation. This tool was developed to be used in the EUPASS project which highlights the applicability and relevance of this model within the MAS domain. Furthermore, the use of the tool by MAS experts in the context of this project is viewed to provide the necessary validation of the proposed model.

The second core contribution of this work is the agent architecture that enables the creation of a distributed environment that is able to provide a bottom up configuration methodology for MAS. In this chapter the implementation of such architecture is assessed and its behaviour is validated accordantly to the proposed architectural definition. Furthermore, results on the computational effort required for achieving configurations will be provided for both memory and processing time. These results will also provide insight into the scalability issues for the MAS configuration problem.

Finally the operational demonstration of the self-configuration methodology will be shown for a given scenario. This will contain the analysis if the results achieved by the methodology and their validity. The self-configuration methodology also proposed a new method for the simulation of performance characteristics. This new method will also be demonstrated and validated for a given scenario.

7.2 Validation of Model for Agent-Based Self-Configuration of Modular Assembly Systems

The model for agent-based Self-configuration of MAS provided in Chapter 4 was implemented in the background of the manual configuration tool. This tool allowed for expert users, namely system integrators and module providers, to define the two main inputs for the MAS configuration problem, the MAS requirements and the equipment modules. The tool provides the means to manually configure a given MAS system for a given set of requirements. Finally, and most importantly for this validation, the tool is able to generate the instances for the MAS requirements, the equipment module descriptions and the configuration solution according to the proposed model, which can be imputed into the self-configuration methodology.
In this subchapter a complete configuration scenario used for the EUPASS project will be presented. The scenario will be broken down into the three main aspects of the proposed model, namely the MAS requirements, the equipment descriptions and the solution of a manual configuration process. These will provide insight into the important aspects of the proposed model, while demonstrating its validity to represent the available data.

7.2.1 Validation Scenario

The validation scenario described in this subchapter has the main objective of validating the proposed MAS configuration model. Additionally the scenario intends to illustrate and benchmark the configuration process which is fairly complex, with a series of constraints that mostly sit on the head of the expert user. The development of a tool to capture this, demonstrated the benefit of having an automatic configuration process.

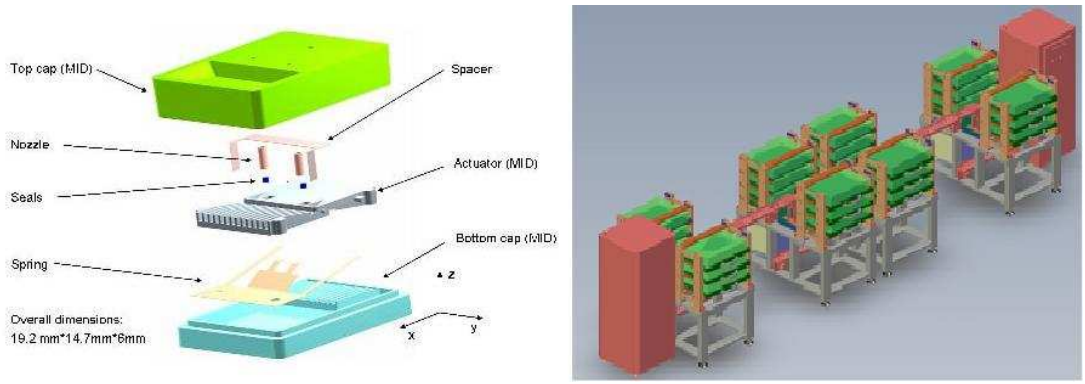


Figure 7.1 – Overview of the EUPASS Final Demonstrator

The validation scenario for the model for agent-based self-configuration of MAS is extracted from the EUPASS project final demonstrator. This demonstrator is composed of three workstations that assemble the two main components of a valve for Festo **Figure 7.1**. The proposed model does not cover the product description, thus for the purposes of the validation, the definition of the product is outside of the scope of this work. As such, the validation scenario starts with the definition of the assembly process requirements and is followed by the assembly system requirements definitions. The requirements definition process will also cover the definition of the business aspects related to the required MAS System. The creation of MAS requirements is preceded by the task of creating the definitions for equipment module. These are stored in an equipment module library and will be used for the manual configuration process which will use them for the fulfilment of the given set of requirements. **Figure 7.2** provides an overview of the whole process for the definition of this validation scenario.

The assembly process requirements defined take into account the product requirements. These are defined by establishing what is seen to be required for an assembly of the given product. The assembly processes' library, as described in

chapter 4, provides an extensive list of possible assembly processes to use as requirements. Another important aspect is the level of granularity of the assembly processes, which can define very restrictive requirements, e.g. specifying the lower level assembly processes that will be required, or a higher level, leaving it up to the configuration process to define the specifics of the lower levels. In the EUPASS project, the system integrator that defined the requirements established very strict requirements, since the equipment module pool was not very wide. As such, the assembly process requirements are quite detailed, which obviously facilitates the manual configuration process.

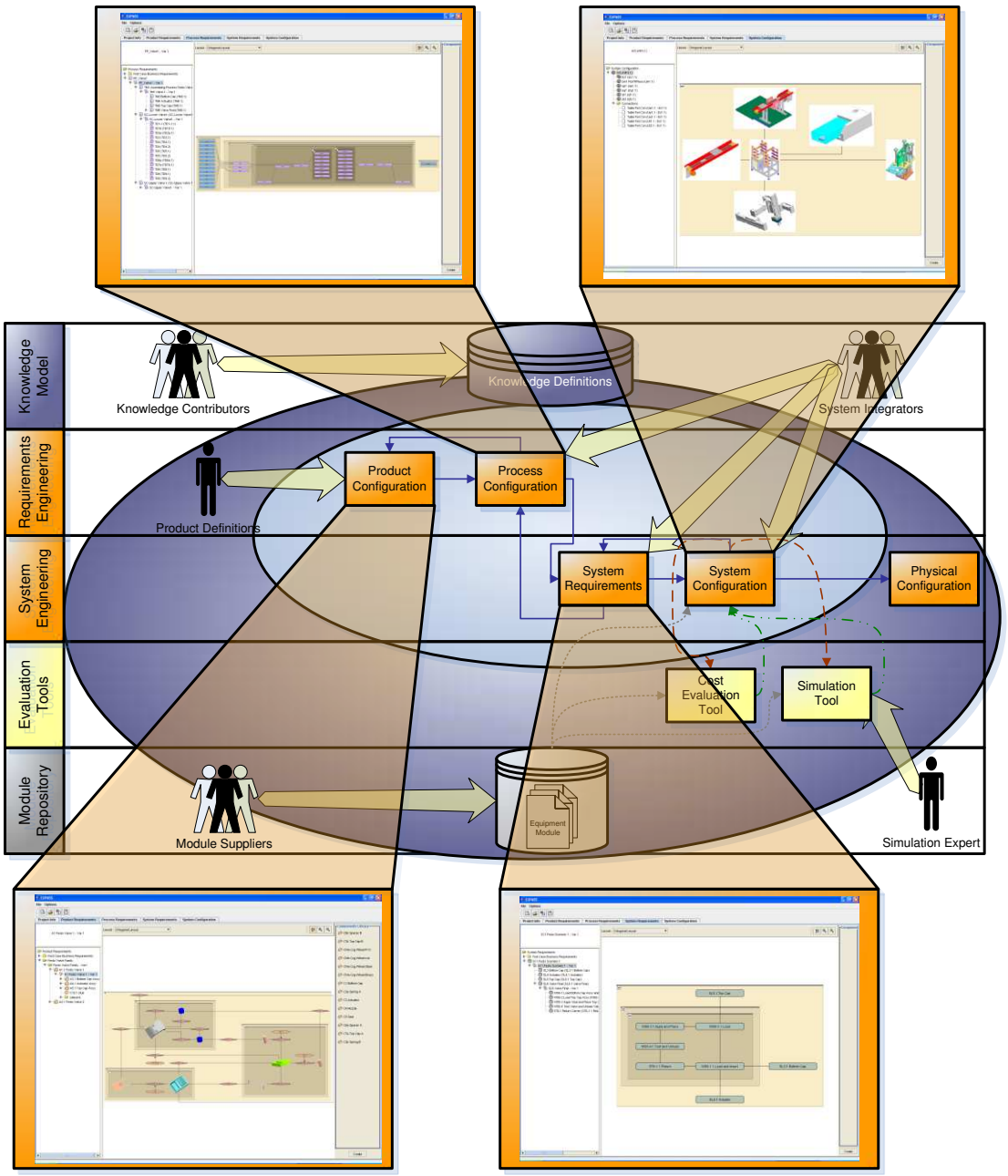


Figure 7.2 – Overview of the EUPASS project configuration process

The validation of the proposed model requires a break down into the three aspects of the configuration process, namely the definition of equipment modules, the definition of the MAS configuration requirements and finally the configuration solution description. Therefore these aspects will be covered individually in the following sub-chapters.

7.2.2 Instantiation of Equipment Modules for Illustrative Scenario

The instantiation of equipment modules for the given EUPASS scenario consists of several equipment modules. However, for the validation of the proposed model one only requires the definition of one of these modules, since the other modules would be a repetition of this process. Therefore the validation of the equipment module description provided in **Chapter 4** will provide a conceptual description of one of the equipment modules present in the scenario, followed by its representation given the proposed model.

The equipment module chosen for instantiation was the manipulator, which is one of the most complex equipment modules available. This equipment module in terms of MAS configuration consists of physical aspects and logical aspects. In physical terms this module fits in a given bay structure. Therefore, its description requires the definition of an interface that is composed of two physical ports, which represent the equipment and the bay structure where it fits. This provides the connectivity of the module to the system. Following the proposed model, the interface library would have to contain the description of this interface, and one of its ports has to be part of this equipment module. The other physical port required for this equipment module is one that allows for its connection to the gripper. Again, for this definition to be valid, the respective interface needs to be defined. However, the equipment module considered the manipulator and the gripper as a whole due to restrictions on levels of granularity, which resulted in the final port physical part which is the component port, which again is part of a defined interface.

The restrictions on the equipment module granularity have an impact on the logical description of this equipment module. In this case, this resulted in a high level ability to handle products or components. This is a complex assembly process that combines moving and gripping, which enables the equipment module to pick, handle and place

components. The definition of these assembly processes required an assessment of which assembly processes contained in the assembly process library reflect the capability of this module. The equipment module provider was invited to establish its equipment module capabilities based on an existing assembly process library. The result of this equipment module was described to have a handling operation which contains the standard control ports, which enable the triggering of this capability, and one parameter port that enables the definition of a destination point. In addition to this assembly process, the equipment provider also identified the ability of the module to execute a pick operation and a place operation. **Figure 7.3** shows a conceptual view of these assembly processes and their respective ports.

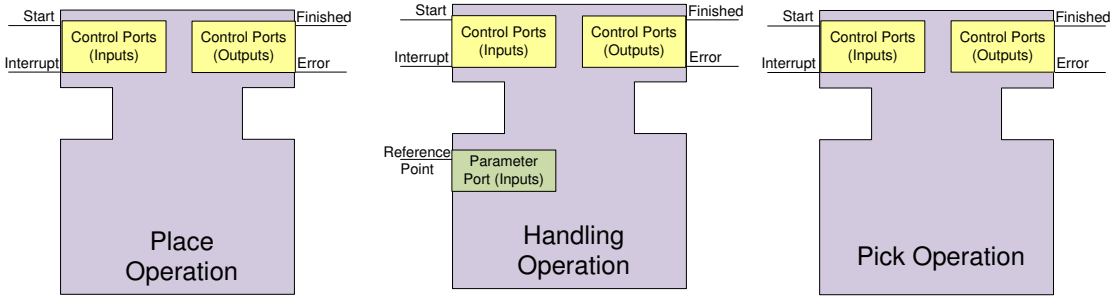


Figure 7.3 - Conceptual Definition of Assembly Processes

The definition of equipment module also contained the physical port descriptions as well as other control specific aspects. It is important to note that these are not relevant for the MAS Self-Configuration methodology since they focused on specific implementation problems. **Figure 7.4** provides a conceptual overview of the full equipment module description and its relations with the existing libraries.

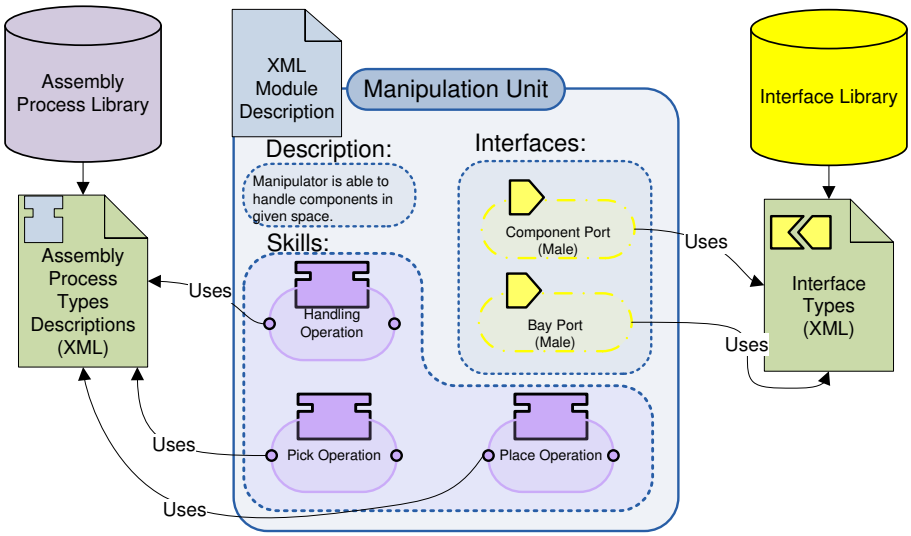


Figure 7.4 - Conceptual Manipulator Unit Description

The definition of this module using the model proposed in Chapter 4 uses the XSD file for the equipment module for the generation of a template. This template already contains the restrictions for using only ports that are present in the interfaces library and only the assembly processes contained in the assembly process library. This ensures that the important aspects for the configuration methodology used the same terminology as the MAS requirements definition which also adheres to definitions contained in these libraries. Furthermore, a template also enables the obligation to define certain aspects, namely the ones that enable the decision-making capabilities for the configuration methodology. **Figure 7.5** provides an XML grid overview of this equipment module description.

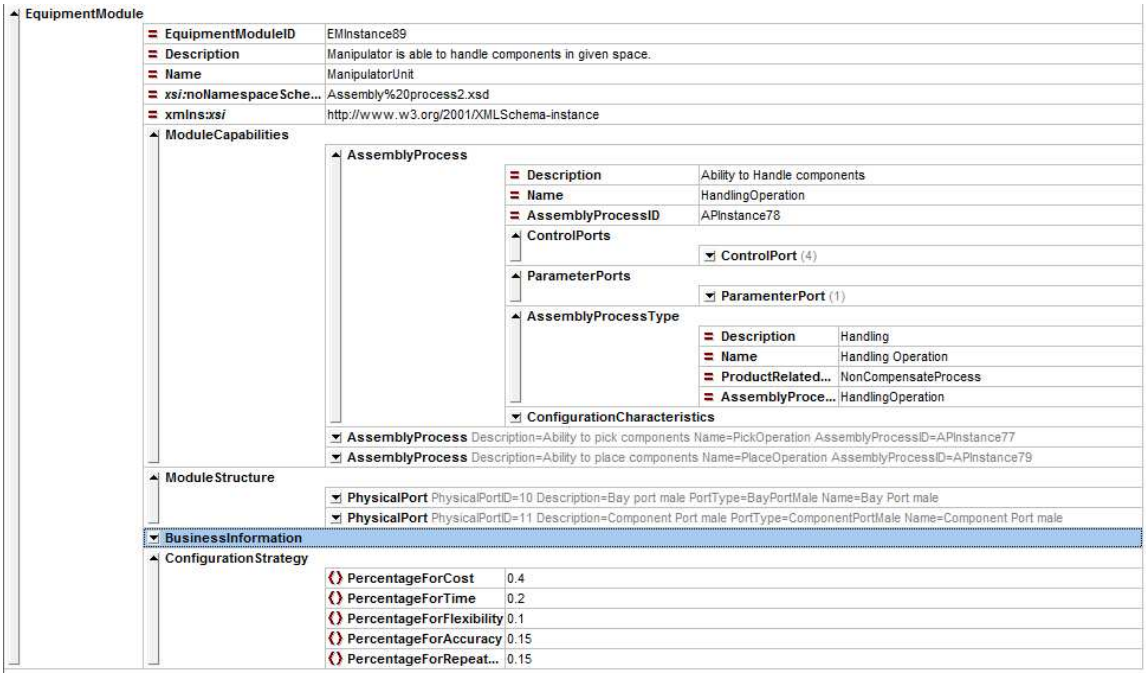


Figure 7.5 - Grid Overview of Manipulator Unit XML Description

7.2.3 Instantiation of MAS Requirements for Illustrative Scenario

The definition of MAS requirements defined, within the context of EUPASS demonstrator, are quite extensive and detailed. The requirements were broken down into product requirements, process requirements and system requirements. The business requirements were kept separate from this and were the first aspects to be defined. The proposed model does not cater for product requirements, therefore this description will only focus on the process requirements, system requirements and business requirements.

This scenario targets the assembly of the final components of a valve. This is important to understand the assembly process requirements, since these are based on the product requirements. The assembly process requirements at high-level require the definition of assembly processes for the loading of the two components into the system and their assembly. The assembly involves a gluing process that binds the two components together to form a product. The final stage is of course the extraction of the final product from the system. **Figure 7.6** provides an overview of the conceptual high-level requirements already using the terminology contained in the assembly process library.

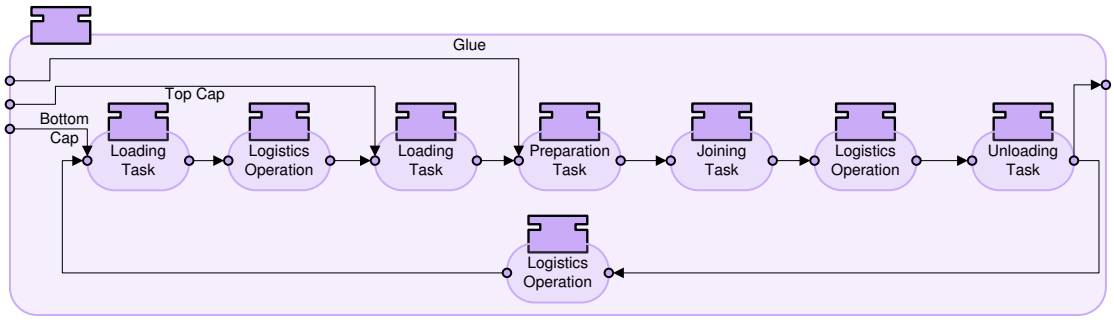


Figure 7.6 - EUPASS Demonstrator Conceptual High Level Assembly Process Requirements

The high-level assembly requirements can be broken down into lower level assembly process requirements using the concepts described in the proposed model. These enable several levels of granularity which can be used for more detailed requirements. **Figure 7.7** provides the details for preparing the top cap of the valve.

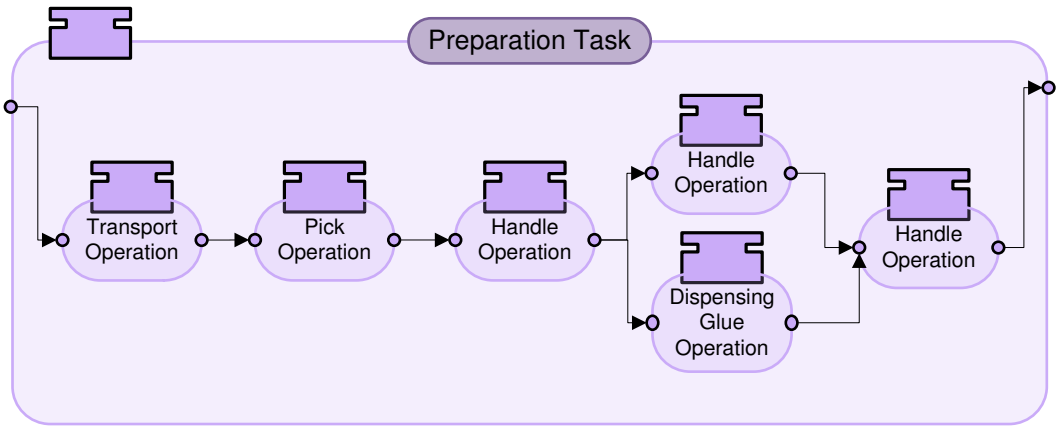


Figure 7.7 – Detailed View of Preparation Task (Assembly Process)

The definitions of the assembly processes requirements can contain several characteristics which are important for the MAS self-configuration methodology as

defined in **Chapter 4**. The details of these will be introduced based on the developed requirements definition tool.

Figure 7.8 shows an overview of the assembly processes defined for the EUPASS demonstrator. This contains several assembly processes with different levels of granularity, namely tasks and operations. Also present are the supply chain processes which are defined in a different colour to emphasize their difference.

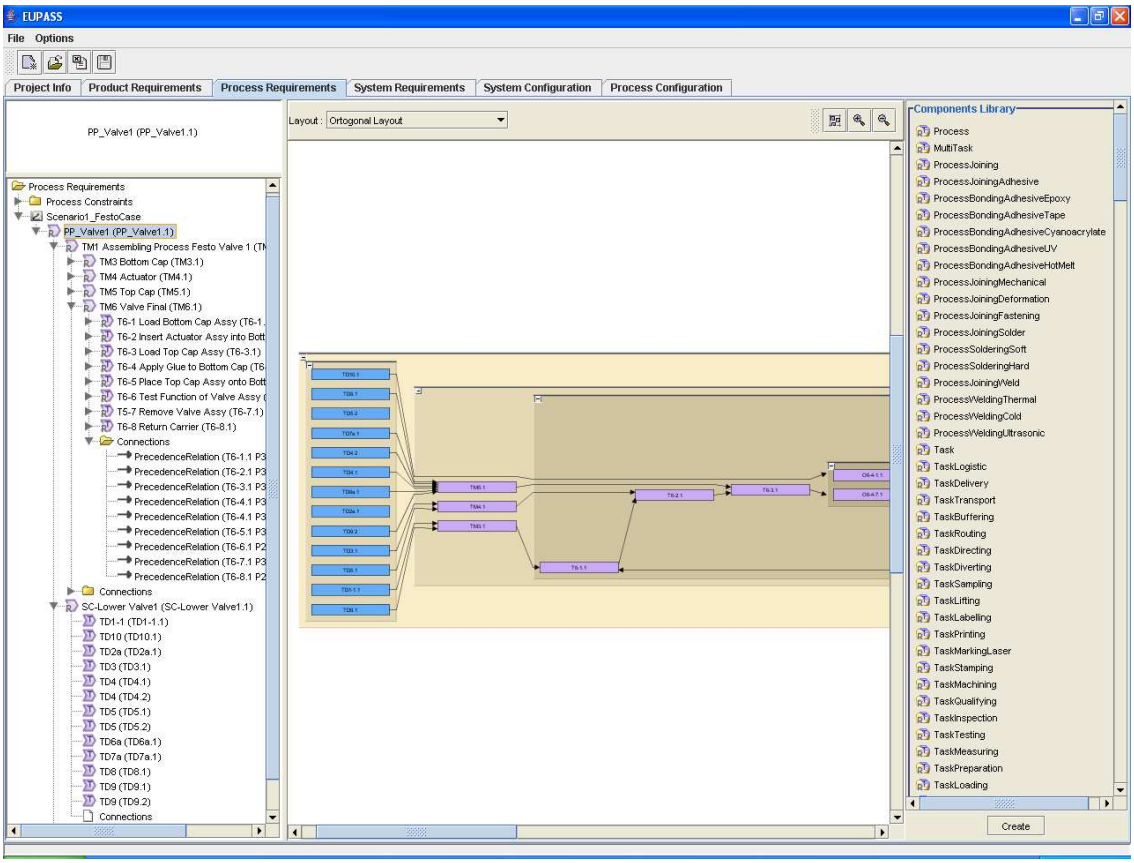


Figure 7.8 - Overview of Process Requirements Specification Front End

The process requirements definition details can be found in **Appendix D**, were the overview of the requirements definition tool is detailed.

The definition of assembly process requirements is followed by the definition of the system requirements. In the case of the considered demonstrator, system requirements are quite detailed since they targeted also the introduction of new concepts like the bay structure for the assembly line. This also facilitated the configuration process since it was a manually driven process.

The system requirements for the last demonstrator established four workstations that enabled the distribution of the high-level assembly process requirements across the

different workstations. In addition to the workstations, some transport units have also been introduced between certain workstations to demonstrate the potential of the modular approach. It is important to note that these requirements were very specific to demonstrate different aspects of modular systems. **Figure 7.9** provides a conceptual view of the system requirements and their assigned assembly process responsibilities.

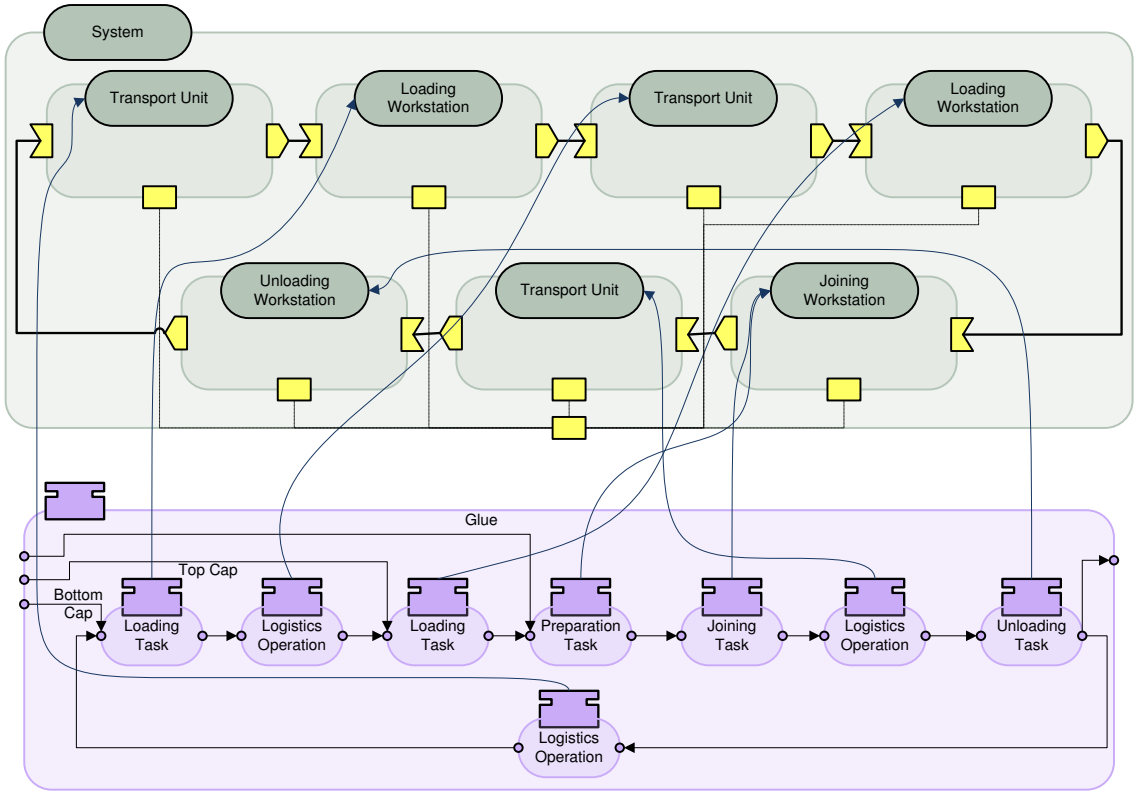


Figure 7.9 – EUPASS Demonstrator System Requirements Overview and Assigned Assembly Process Responsibilities

The definition means that the specific workstations will be responsible for the task they have been associated with. For the instantiation of these requirements the system requirements tool was used. The tool is similar to the assembly process requirements definition and allows for the conceptual assembly system design process which defines an assembly system concept which in turn fulfils the set of requirements. **Figure 7.10** provides a screen shot of this tool where the conceptual system for the valve is shown.

The export of the MAS requirements into the proposed model was done using the XSD file with all the constraints defined in chapter 4. The use of this file allows for the validation of its content, including the use of the require terminology for the

assembly processes and interfaces. This export functionality was used for exchanging requirements with other tools and experts user within the context of the EUPASS project. **Figure 7.11** provides an XML description overview of the requirements.

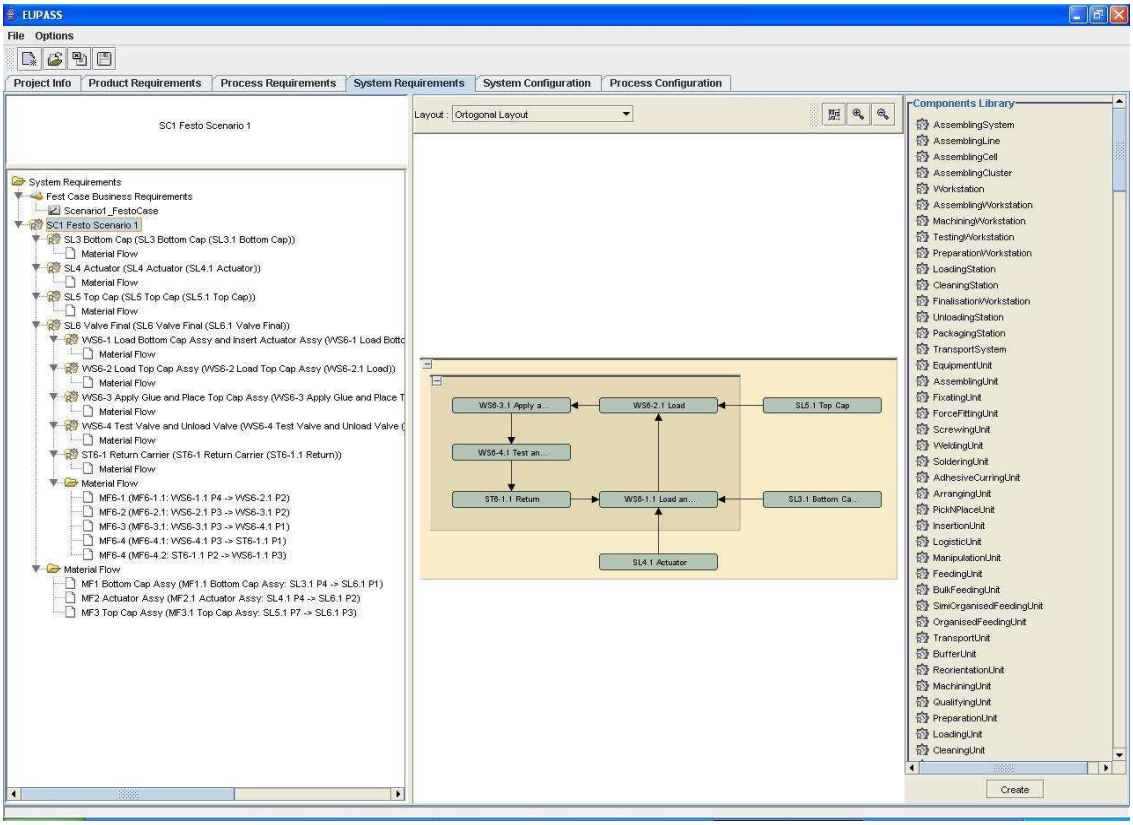
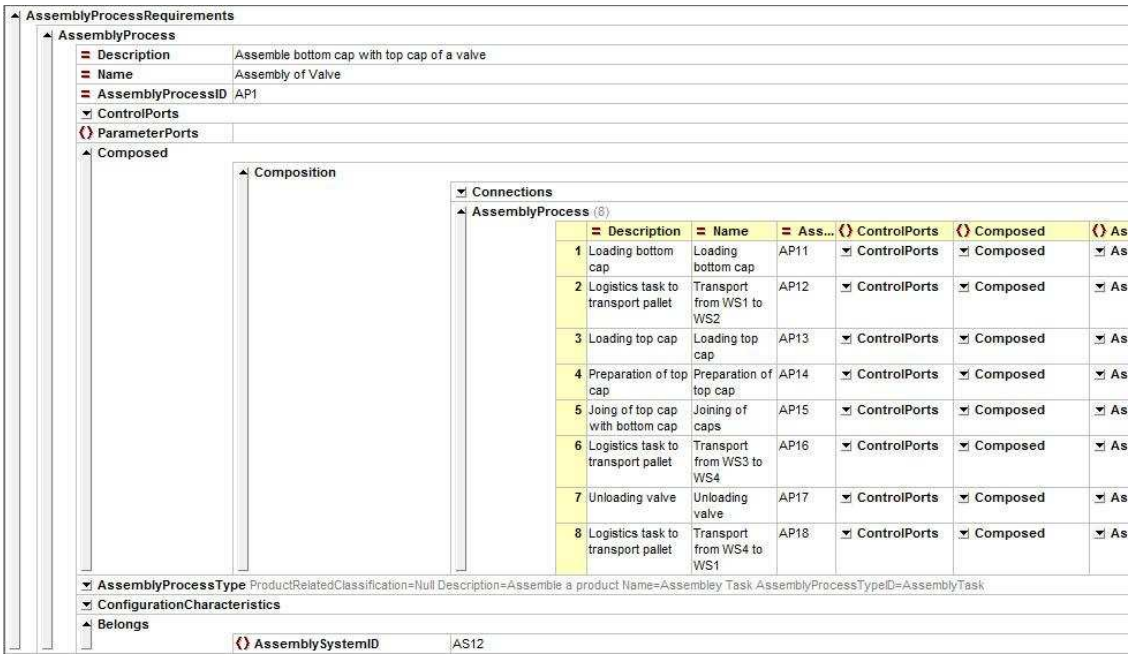


Figure 7.10 - Overview of System Requirements Specification Front End

XML		
Comment	edited with XMLSpy v2010 (http://www.altova.com) by jack (un)	
Comment	Sample XML file generated by XMLSpy v2010 (http://www.altova.com)	
AssemblySystemRequirements		
= Description	EUPASS demonstrator requirements	
= Name	EUPASS Demonstrator	
= RequirementsID	R12	
= xsi:noNamespac...	.\Assembly%20process3.xsd	
= xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance	
AssemblySystemTargets		
FixedTargets		
Time		
{} CycleTime	60s	
{} Commissioning...	2months	
{} Accuracy	0.01±0.005	
Cost		
Buy		
= Value	500000	
= Currency	Euros	
{} Repeatability	0.01±0.005	
ConfigurationStrategy		
{} PercentageForCost	0.4	
{} PercentageForTime	0.2	
{} PercentageForFlexibility	0.1	
{} PercentageForAccuracy	0.15	
{} PercentageForRepeatab...	0.15	
PhysicalSystemRequirements		
AssemblySystem	Description=EUPASS demonstrator physical requirements Name=EUPASS Demonstrator AssemblySystemID=AS...	
AssemblyProcessRequirements		
AssemblyProcess	Description=Assemble bottom cap with top cap of a valve Name=Assembly of Valve AssemblyProcessID=AP1	

Figure 7.11 – EUPASS Demonstrator Grid Overview of MAS Requirements

The overview does not provide much information on the main descriptions of the physical system requirements and the assembly process requirements due to the amount of information it contains. Nevertheless it is important for the model validation to expand the descriptions focusing on certain aspects of the defined requirements. **Figure 7.12** provides an overview of the XML description for the assembly process requirements, where it is clearly shown that it is composed of a set of assembly processes as seen in **Figure 7.6**.



	Description	Name	Ass...	ControlPorts	Composed	As
1	Loading bottom cap	Loading bottom cap	AP11	ControlPorts	Composed	As
2	Logistics task to transport pallet	Transport from WS1 to WS2	AP12	ControlPorts	Composed	As
3	Loading top cap	Loading top cap	AP13	ControlPorts	Composed	As
4	Preparation of top cap	Preparation of top cap	AP14	ControlPorts	Composed	As
5	Joing of top cap with bottom cap	Joining of caps	AP15	ControlPorts	Composed	As
6	Logistics task to transport pallet	Transport from WS3 to WS4	AP16	ControlPorts	Composed	As
7	Unloading valve	Unloading valve	AP17	ControlPorts	Composed	As
8	Logistics task to transport pallet	Transport from WS4 to WS1	AP18	ControlPorts	Composed	As

Figure 7.12 - EUPASS Demonstrator Grid View of XML Description of High Level Assembly Process Requirements

The high level assembly processes were broken down into lower level ones. This highlights the flexibility of the proposed model for describing several levels of granularity. The high level assembly process described in **Figure 7.7** resulted in the XML description seen in **Figure 7.13.**

Similarly the physical requirements that are described in **Figure 7.9** are provided in the XML format using the model proposed in **chapter 4**, and can be seen in **Figure 7.14**.

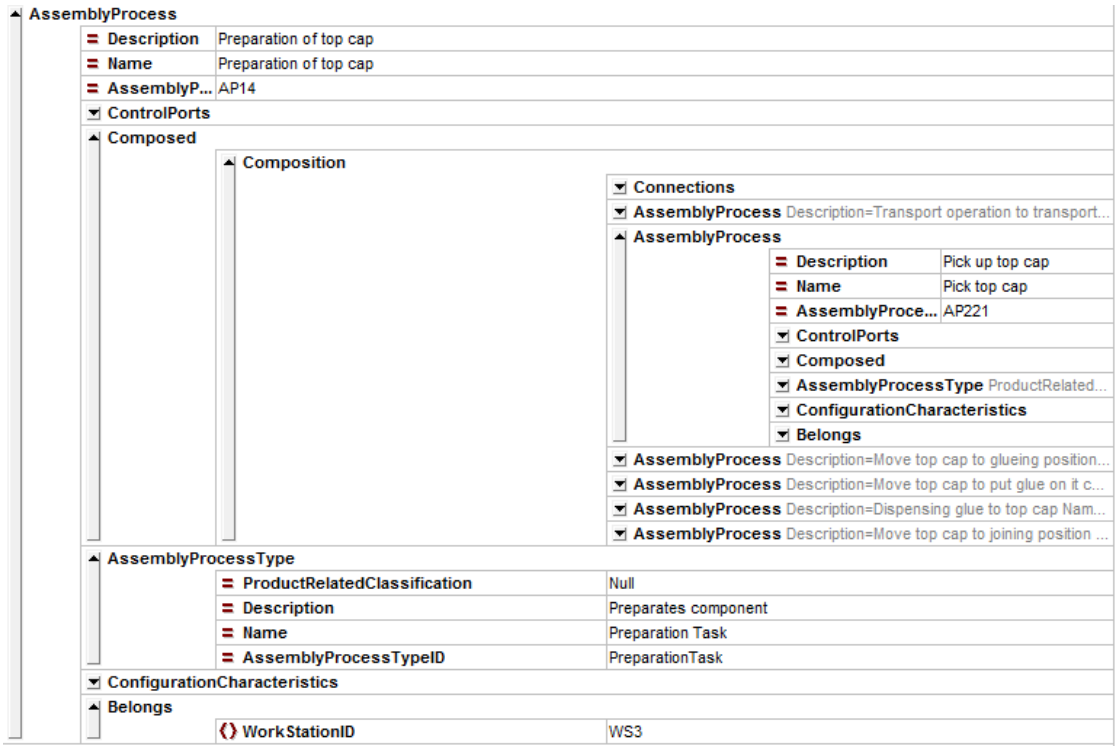


Figure 7.13 – Detailed Grid View of Preparation Task (Assembly Process) XML Description

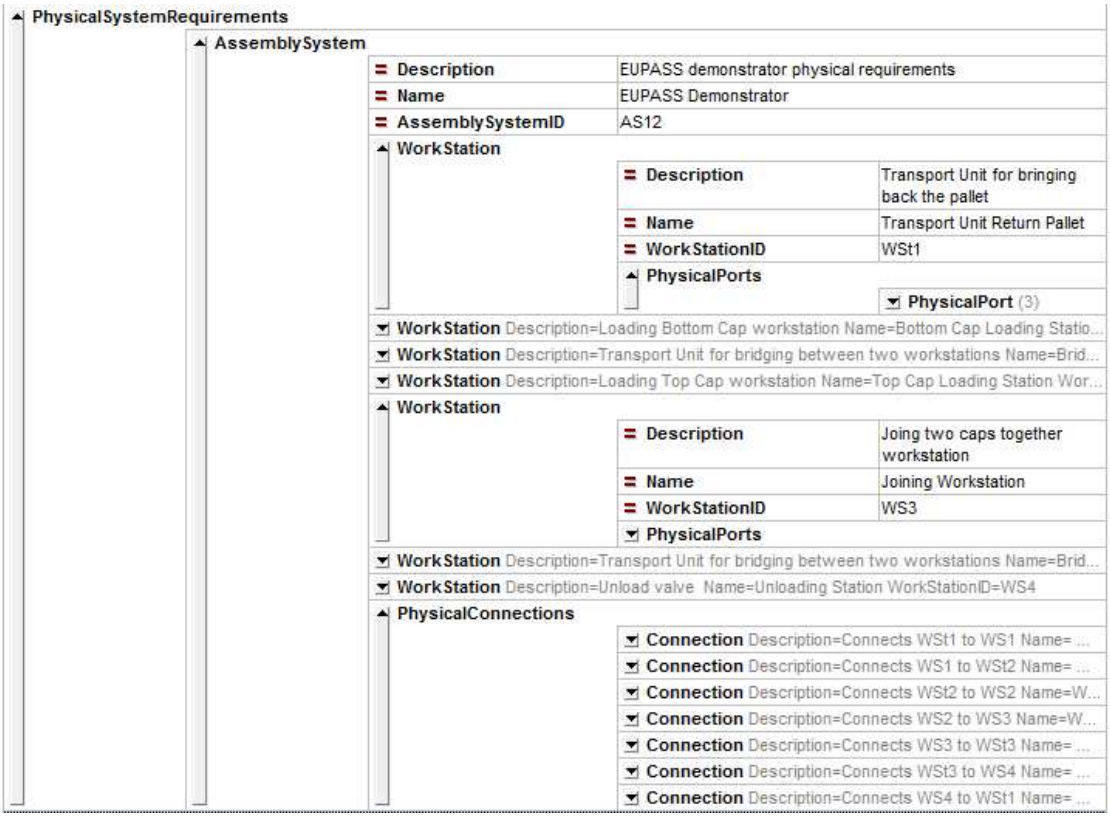


Figure 7.14 - EUPASS Demonstrator Grid View of XML Description of System Requirements

7.2.4 Instantiation of Configuration Solution Output

In order to assess the configuration solution output one needs to provide a solution. Furthermore for the purposes of understanding the configuration methodology it is useful to follow the steps of the manual configuration process. For simplicity this illustration will focus on one of the three available workstations.

The configuration process essentially consists of two parts; the selection and configuration of available equipment modules into possible physical system solutions and the configuration of the control logic. The physical configuration of the system focuses on the selection of appropriate equipment modules based on their capabilities and interconnection constraints and connecting them together. The process logic focuses on defining the sequential order between the skills of the equipment modules selected for a system configuration.

Figure 7.15 shows an overview of the main interface used by the assembly system configuration tool. The illustrated example shows a possible workstation configuration for the placing and gluing of a valve top cap onto the main assembly. The interface shows the hierarchical structure of the configuration and the physical interrelationships between the modules. All equipment modules are integrated by reference only into the underlying assembly system configuration model. The main objective of the tool is to find the most suitable modules and connect them to each other.

The configuration of assembly system solutions is a bottom up approach. The tool does however create an empty system structure based on the associated assembly system concept to maintain the consistency of the models. This structure is strictly speaking generated in a top down fashion but remains empty until it is being populated with detail from the lowest level (bottom up). Details on the configuration process using the assembly system configuration tool can be found in **Appendix E**.

The next step of the MAS configuration process is the configuration of the logical aspects of the system, namely the definition of the possible assembly process sequences for a specific system configuration. The configuration tool allows the configuration of assembly processes contained in the system which are provided by the equipment module descriptions files.

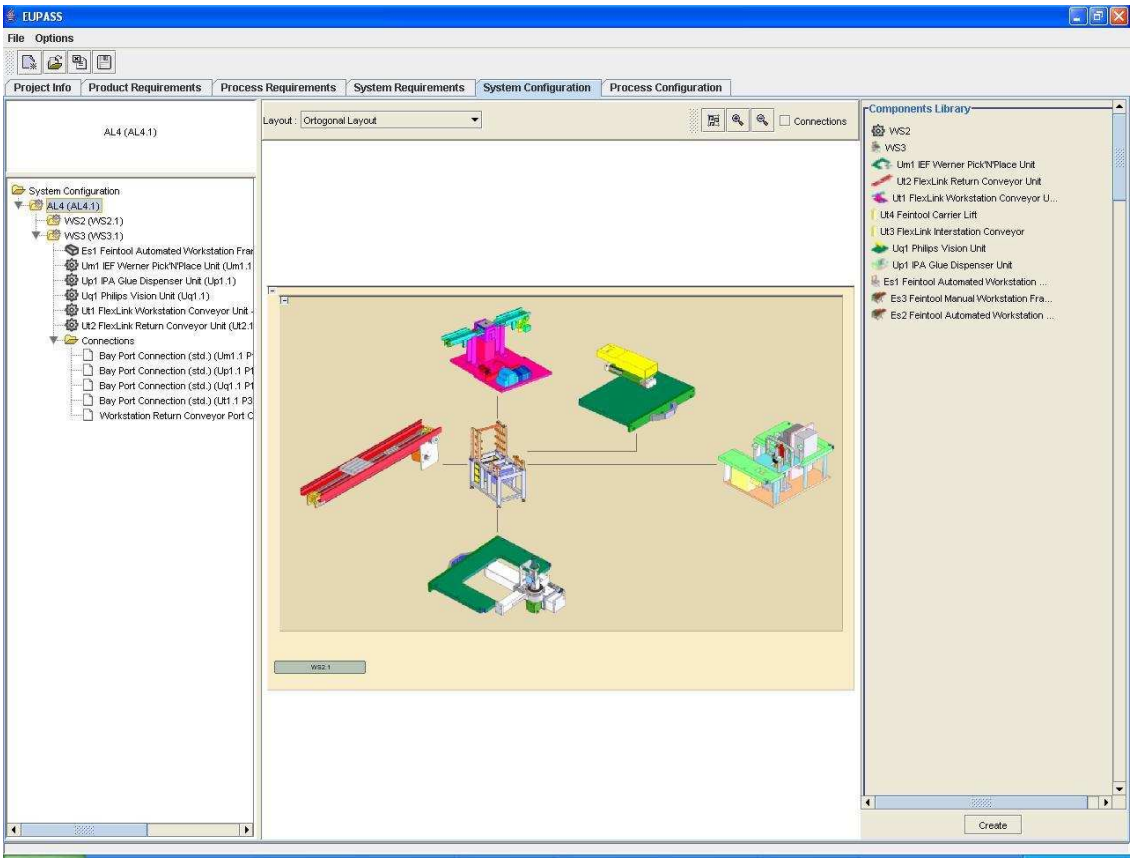


Figure 7.15 - Overview of Physical Configuration Front End

Figure 7.16 shows an overview of the main interface used for the process configuration. The illustrated example shows a process configuration for the proposed workstation configuration for the placing and gluing of the valve top cap onto the main assembly. The interface shows the sequential structure of the configuration as well as the control interfaces between the processes. The main objective of the tool is to find the best possible process configuration for a given system and convey it to the line configurator.

The Assembly Process Configuration tool is intrinsically related with the System Configuration since the base structure for process configuration is generated from the system configuration structure in a top down approach and it remains empty until it is being populated with detail from the lowest level (bottom up). Further details on the Assembly Process Configuration tool can be found the **Appendix E**.

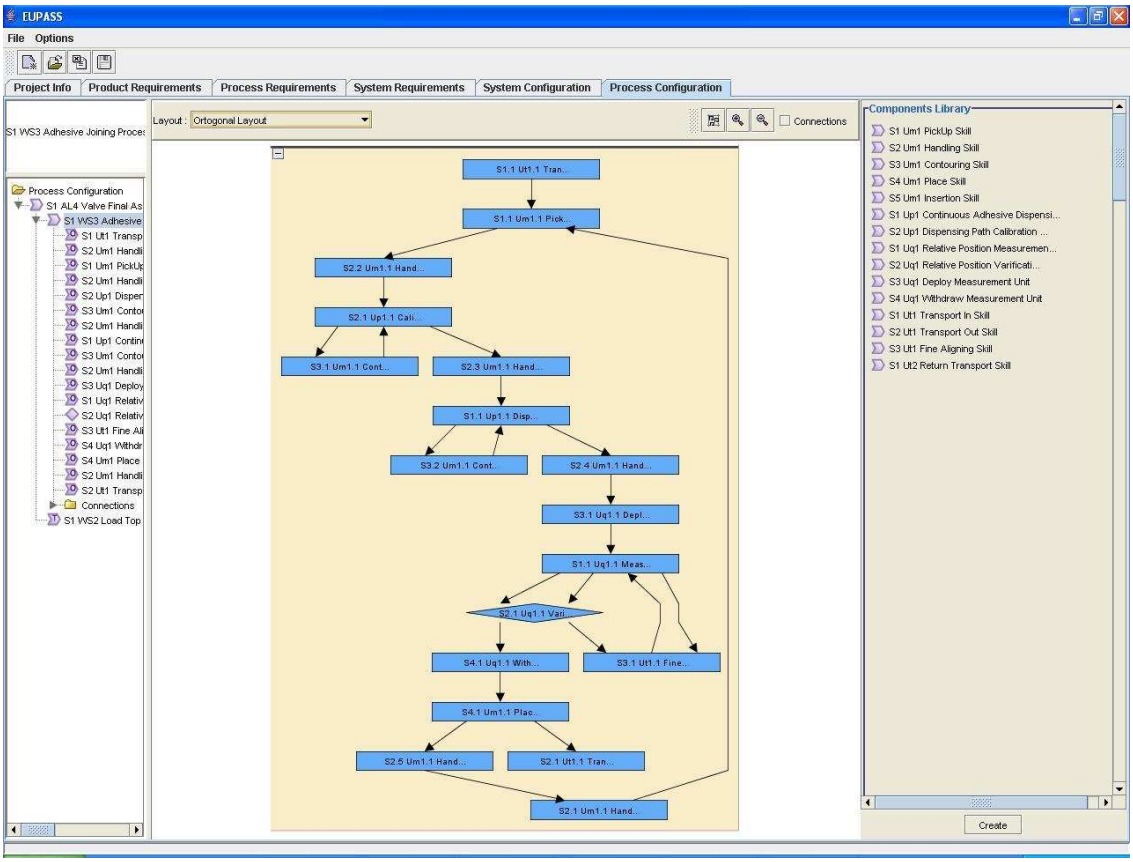


Figure 7.16 - Overview of Assembly Process Configuration Front End

The final stage of the configuration process is to export MAS configuration solution which uses the model defined in **chapter 4**. The output of the tool is validated against the XSD model to determine its validity according to the model. The model builds on the requirements model filling in the missing elements as they are configured. As such, the important aspect to focus on is the lower level elements. In the considered demonstrator the assembly process requirements are quite specific, thus the focus of the configuration output analysis is on the added equipment modules and the assembly processes they have been assigned to configure. **Figure 7.17** provides an overview of the joining workstation configuration highlighting the assembly processes responsibilities of some equipment modules.

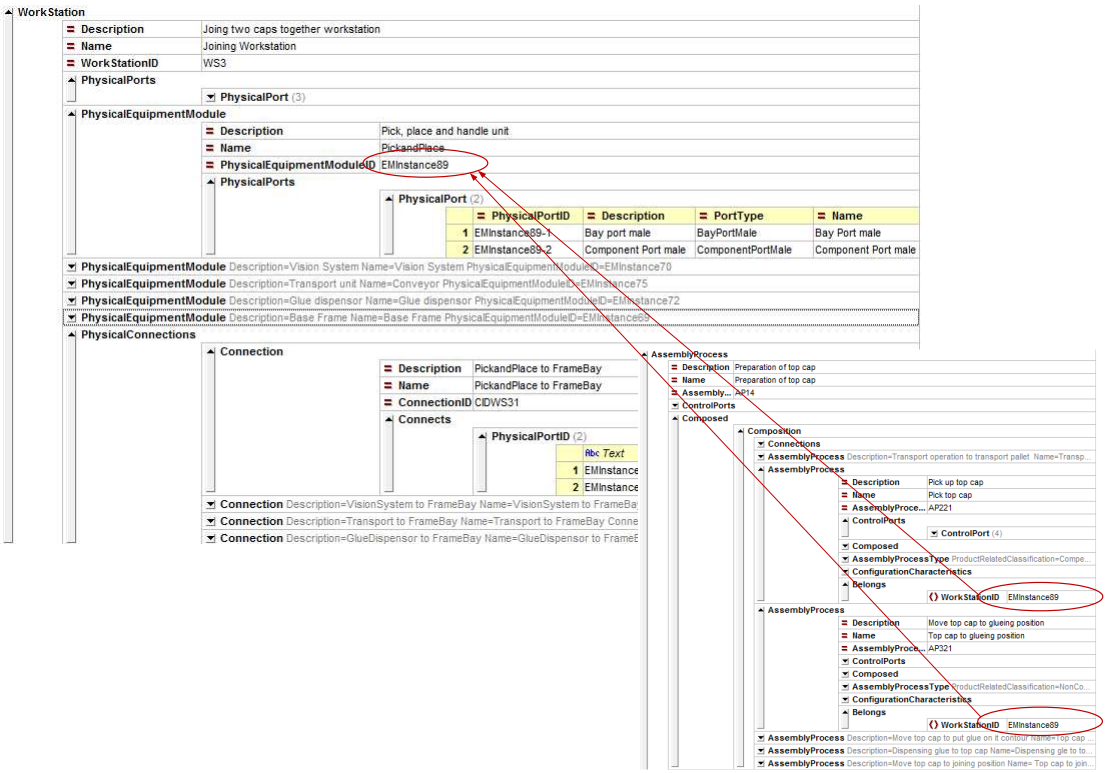


Figure 7.17 - Grid View of Joining Workstation XML Description of the EUPASS Demonstrator

7.2.5 Analysis of Validation Results of Model for Agent-Based Self-Configuration of Modular Assembly Systems

The proposed model was used under the EUPASS project, were it was tested under the scope of the project. The model proved to be useful particularly in the exchange of structure data between different tools and partners. The available data was accurately represented by the model and was used for the development of the attributes included in the model. The validation description provides an overview of the stages of the configuration process and the importance of the proposed model for capturing structured information that is required across the whole configuration process. It is important to note that this model was developed with the collaboration of EUPASS project partners.

The proposed model was considered suitable by academic and industrial experts involved in the EUPASS project which targeted the advancement of MAS, therefore the proposed model is viewed as a good contribution for the development of the MAS domain.

7.3 Operational Validation of Agent Architecture for Distributed Self-Configuration Methodology for Modular Assembly Systems

The operational validation of the proposed agent architecture for distributed self-configuration methodology provides insight into the projected interactions between agents and verifies that the execution of the overall architecture behaviour, which is expected to provide configuration solutions using a bottom up approach. With this in mind an agent environment that implements the proposed architecture was developed using the JADE platform.

The verification that all agents operate according to the projected overall behaviour is the first validation step of the proposed architecture. If the agents overall behaviour is not as expected, then the proposed architecture is obviously flawed. To achieve this verification a simple scenario should be followed so that the interactions between agents are restricted to a limited number for better clarity of the results.

A second aspect that needs to be verified is how the proposed solution behaves when the solution pool grows. It was theorized that if the solution pool grows, the computational effort will also grow exponentially. Therefore, a scenario for large solution pool should be developed and the behaviour of the agent environment should be assessed in terms of computational effort. Another important aspect to consider is the number of messages exchanged between agents in the time it takes to find the configuration, since it provides an indicator for the communication effort. These indicators will be used to assess how the agent environment behaves with a growing number of solution possibilities.

The proposed architecture established the possibility to restrict the number of contacts between equipment module agents. This restriction was introduced for the verification of the impact this limitation would have on the quality of the solutions. Despite the fact that the calculation of the quality of the solution will be verified in the next subchapter, the results are important here for the validation of the overall architecture, since this is expected to reduce significantly the computational effort, the time to find a solution and the number of message exchanges between agents.

Finally, perhaps the most important validation element is the distribution of the agents across different computers to assess the impact of this distribution. Towards that end a scenario needs to be defined so that the results are clear in terms of the distributed behaviour of this architecture.

7.3.1 Validation Scenarios

The analysis of the validation aims defined previously clearly identifies the need for the definition of three validation scenarios. The first scenario should be quite simple to clearly demonstrate the overall behaviour of the proposed architecture. Therefore, this scenario will consist of a workstation configuration with a very limited number of equipment module agents. By using this simple scenario it is easier to follow the overall agent behaviour which in turn provides better clarity of the results.

The requirements for this scenario will be a workstation that is able to feed in a component and place it in a pallet. **Figure 7.18** provides a conceptual overview of the assembly process requirements, including the business requirements and the weights for assessing the configuration solutions.

Characteristic	System Targets	Assessment Weights
System Cost	10000 €	0.3
Cycle Time	60 s	0.2
Flexibility	0	0.1
Accuracy	0.05 ± 0.01mm	0.2
Repeatability	0.05 ± 0.01mm	0.2

Figure 7.18 - Conceptual MAS Requirements Overview

The available equipment modules will be four feeders, three grippers, two manipulators and finally a vision system. The agents will all have the same weights for assessing solutions as the ones established in the requirements to minimize entropy in the system. **Table 7.1** provides overall descriptions for these equipment modules.

Table 7.1 - Overview of Equipment Module Description

Equipment Module	Capabilities	Cost	Cycle Time	Accuracy	Repeatability
Feeder 1	Feeding Operation	2500	3	0.001	0.001
Feeder 2	Feeding Operation	3000	2	0.0001	0.0001
Feeder 3	Feeding Operation	2000	3	0.001	0.001
Feeder 4	Feeding Operation	4500	2	0.0001	0.0001
Gripper 1	Pick Operation	1000	0.5	0.001	0.001
Gripper 2	Pick Operation	700	0.5	0.001	0.001
Gripper 3	Pick Operation	500	0.5	0.01	0.01
Manipulator 1	Handling Operation	4500	5	0.01	0.01
Manipulator 2	Handling Operation	3500	10	0.01	0.01
Vision System	Measuring Operation	1500	0.2	0.0001	0.0001

The second scenario consists of an enhanced version of the previous one. Because the description of equipment modules would take a long time, a random generator of equipment modules of these types was developed. This method enables us to define equipment modules of these types with a certain variation on the attributes that is random, or provides a hardcoded variation in the required number of modules. That is one could not evaluate the solution quality if the equipment module characteristics were not constant. In this scenario growing numbers of equipment modules should be put into the environment to assess the performance of the environment based on computational effort, time to achieve a solution and number of exchanged messages between agents. Furthermore, this scenario is also suitable to test the performance of the environment under different interactions restrictions between equipment modules.

The final scenario can use the previous scenario as a base. The idea is using this scenario in an agent environment distributed across different computers one could assess the performance of the architecture. **Figure 7.19** provides an overview of the distribution of the architecture across three computers, where the requirements agent runs on a separate computer and the other agents are distributed between two other computers.

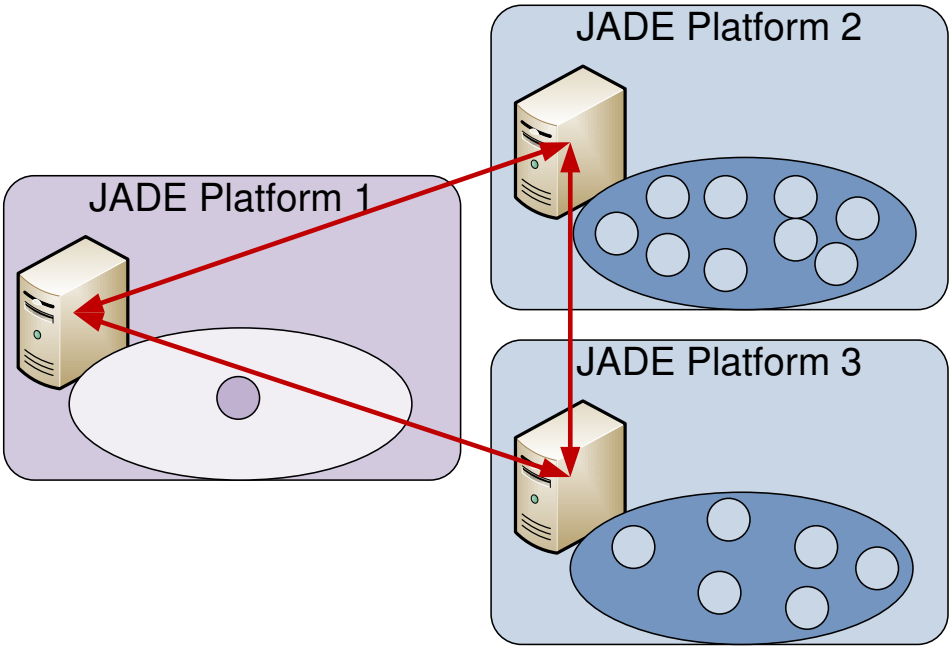


Figure 7.19 - Overview of Architecture Distribution

7.3.2 Operational Verification of Architectural Design

The execution of the first validation scenario provides a quite complex interaction diagram between all agents. Nevertheless, the results clearly show that the agents are able to execute their intended roles as defined in **chapter 5**. This is the first step towards achieving the proposed configuration methodology.

The verification of all agent interactions would be quite unreadable and not suitable for a written document. Therefore, the illustration of the interactions focuses only on interactions between two equipment module agents. Further details on interactions can be generated using the developed software environment. **Figure 7.20** provides a snapshot of interactions between three equipment modules which are obtained using a sniffer agent that is part of the JADE platform (Bellifemine et al. [135]). It highlights the types of messages exchanged between the agents, namely the requests for information, which is answered with the sending of information, or the rejection of a proposal that is followed by another proposal, etc. This provides the evidence that the developed system behaves according to the designed architecture.

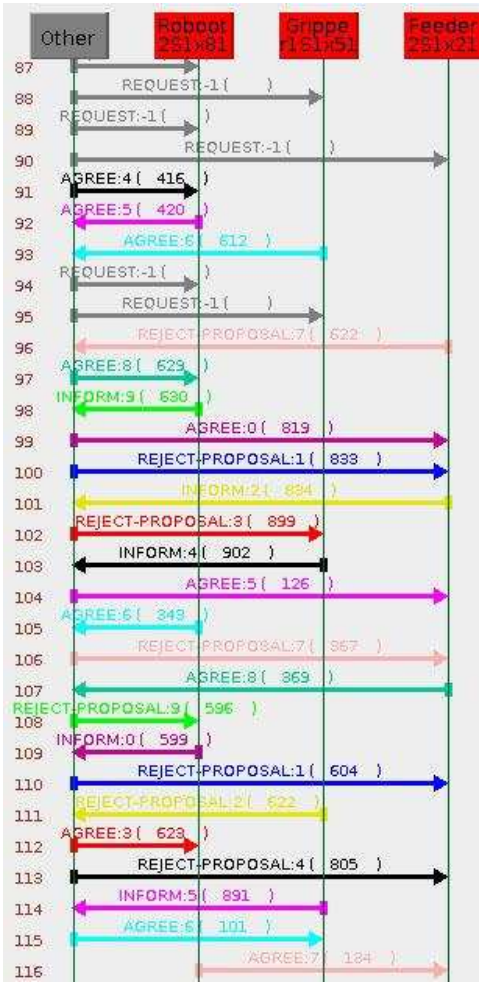


Figure 7.20 – Equipment Module Agent Interactions Screenshot

The agent environment reached configuration solutions as expected. The details of the quality of the solutions will be covered in the operational validation of distributed behaviour in **Sub-chapter 7.4**. Nevertheless, the procedure using the local behaviour was repeated five times, always with the same results to ensure the validation of the architecture operation, while also ensuring the repeatability of the methodology. **Figure 7.21** provides an overview of the possible solutions and highlights the ones selected by the configuration methodology. It is a quite complex diagram, and for a better understanding of it one should focus on one of the equipment modules as a fixed point. **Table 7.2** provides such a view, namely focusing on the manipulator 2 potential solutions.

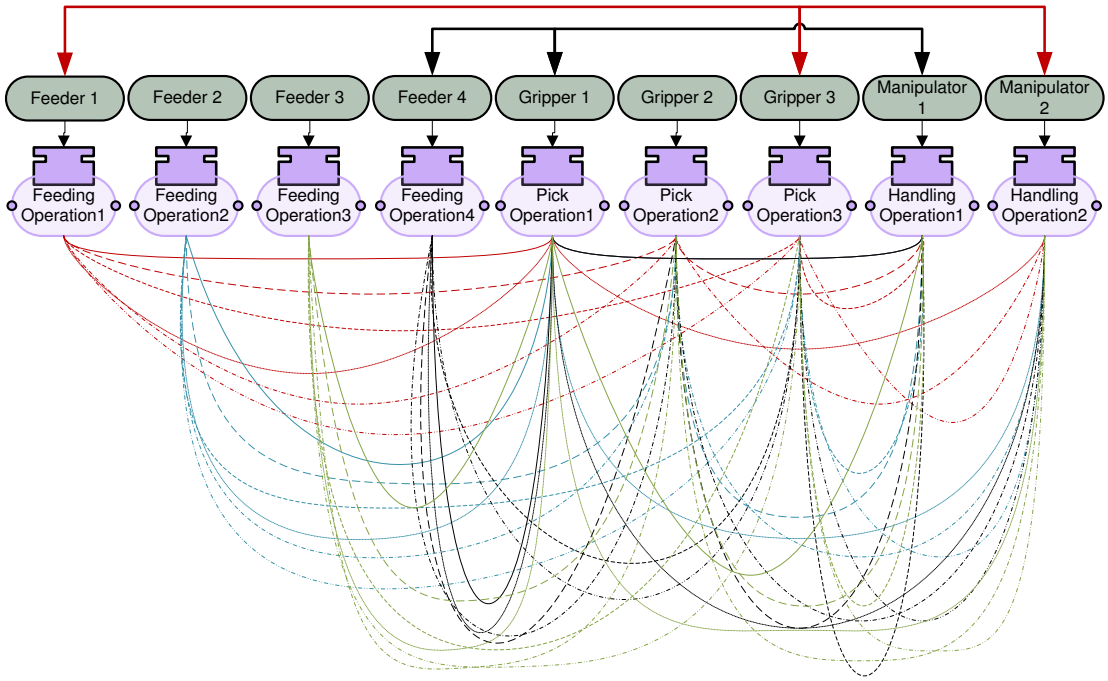


Figure 7.21 – Conceptual Overview of Potential MAS Configuration Solutions

7.3.3 Verification of Architecture Overall Behaviour Performance

The verification of the performance of the proposed overall architecture behaviour will focus firstly on an exhaustive assessment of all configuration solutions. This happens when there is no limitation to the number of interactions that equipment modules can have between each other. The set of available modules from the simple scenario will be duplicated using the equipment module generation method. This procedure of duplication will be repeated and repeated and the results for the performance will be registered.

The results for the performance of the environment are broken down into two sets. On one side, the computational effort which requires an expert tool to assess the computer's memory consumption during the configuration process. **Figure 7.22** provides the screenshots of the memory consumption for the growing number of equipment modules. It clearly shows the impact of having larger numbers of equipment modules in the computational memory resources. The derived results highlight the scalability problem of the proposed solution, using only one computer and without any restrictions on the number of interactions agents are allowed to perform. A final important note for these results is that for more than 160 equipment

modules, the agent platform running on a single computer became unstable and some of the agents crashed which made the results unusable.

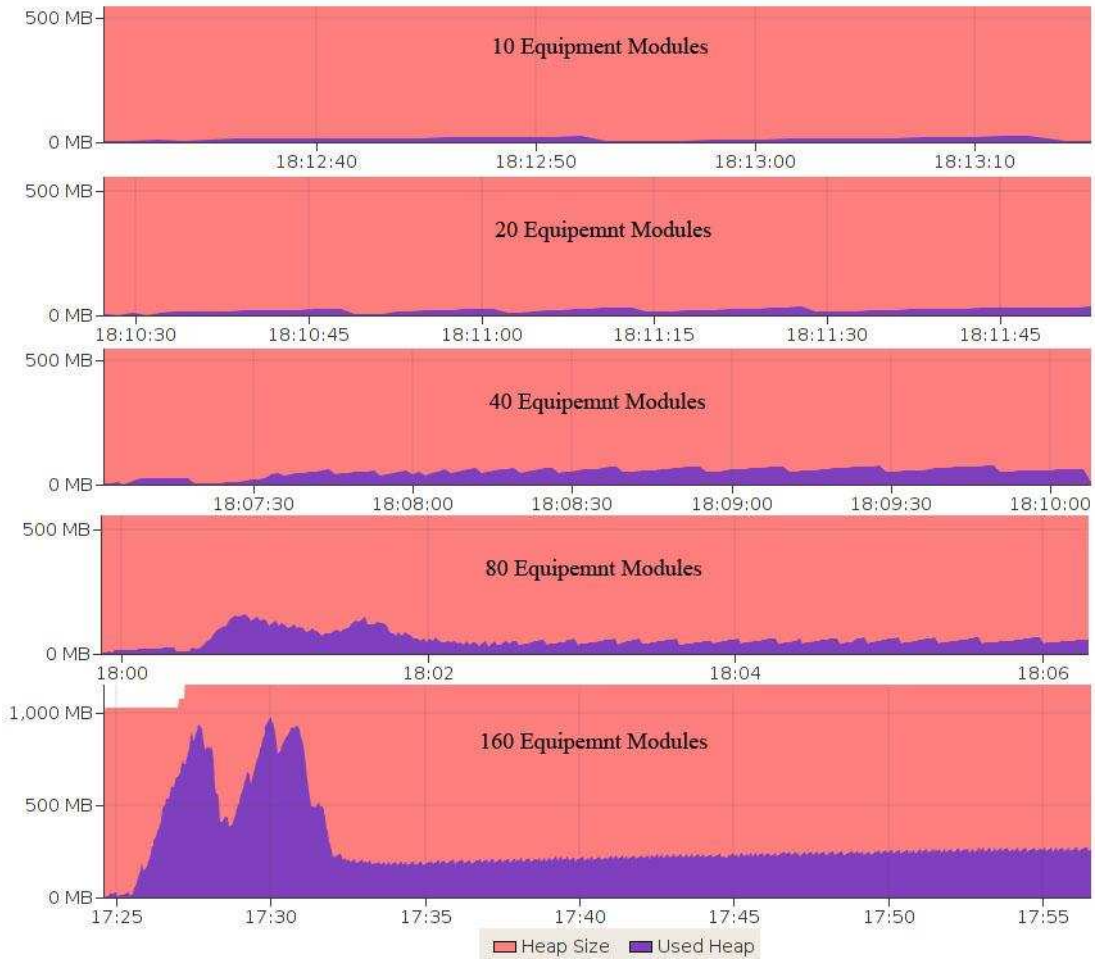
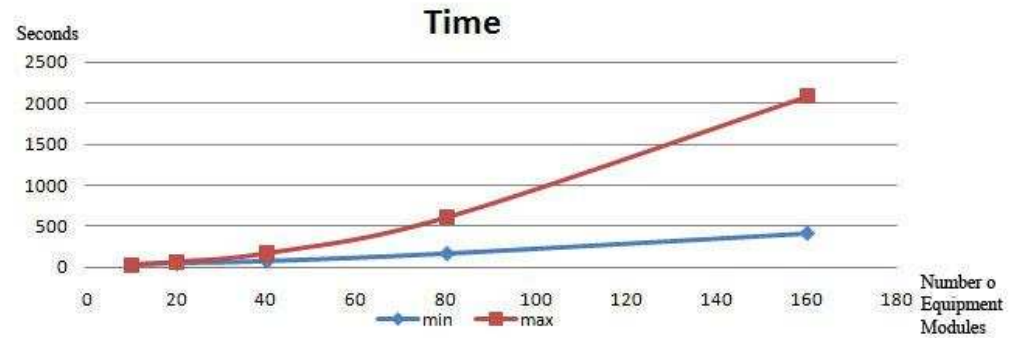
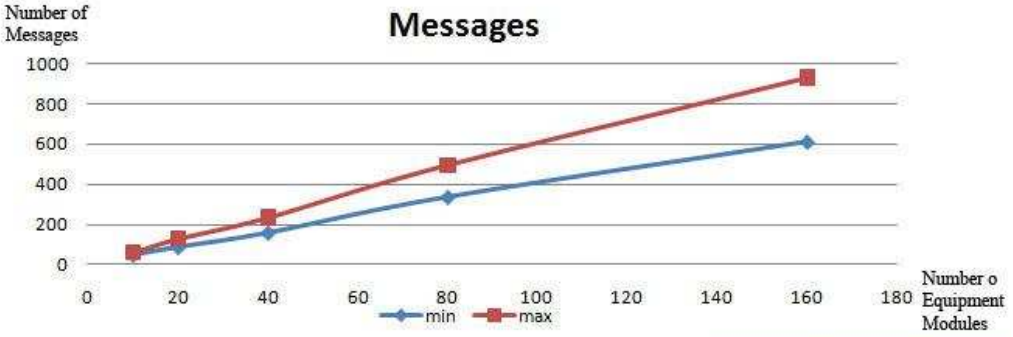


Figure 7.22 – Memory Consumption for no Limitation on Agent Interactions

The other performance results are provided directly by the agent environment and consist of the time to reach a solution, how many messages were exchanged between agents and the value of the configuration. The analysis of these results focuses on its limits, namely the minimum and maximum time to achieve the configuration solution and the number of messages exchanged between agents for those solutions. The value of the solution is not only assessed in its limits but also provides an average value. Furthermore relations between these aspects should be created for a better assessment of the results. **Figure 7.23** details the relevant results achieved under the same conditions used for the computational assessment.

	Time for Solution		Exchange Messages for Solution		
Nº of Modules	Minimum	Maximum	Minimum	Maximum	Nº of Solutions
10	19.786	24.211	46	60	2
20	39.286	61.789	85	129	4
40	64.625	164.529	156	235	8
80	158.537	602.794	335	496	16
160	407.134	2084.531	611	930	32





				Solution Value		
Nº of Modules	Time per Solution	Messages per Solution	Time Variance	Minimum	Maximum	Average
10	12.1055	30	4.425	2.608	3.2418	2.9249
20	15.44725	32.25	22.503	2.6065	3.2418	2.923775
40	20.566125	29.375	99.904	2.60125	3.2418	2.921525
80	37.674625	31	444.257	2.59225	3.2418	2.917025
160	65.14159375	29.0625	1677.397	2.57425	3.2418	2.908025

Figure 7.23 - MAS Configuration Performance Results for no Limitation on Agent Interactions

The results provide very interesting insight into the inner working of the methodology. The fact that the number of messages exchanged increases in a linear fashion, for both first and last solutions found, more or less doubling the previous value, indicates that the methodology does not put too much stress on the network due to agent interactions. The time for the first configuration is somewhat linear, despite growing slightly more than what one would expect from linear functions. However, the last configurations increment significantly more, demonstrating a behaviour closer to exponential. This was expected considering that agents only reject a configuration once they established one, which means the last agents have to wait for the domino effect that is triggered by the rejection process, to finish before making their last decisions. This results in a wait for exploring other options which

has an impact on the increment of the time variance. Another interesting number extracted from these results is the fact that the increase from 20 to 40 equipment modules only produces an increase of 25% in the time per configuration solution, while the next increment has an almost doubling effect. This alone provides a good indication for the number of agents running on individual computers. Finally, it is important to note that the solutions are valid and repeatable, however the performance results vary slightly as expected in any computational intensive task.

The second verification uses the same approach but restricts the number of interactions between different equipment module agents to 10. As expected, the computational effort was significantly reduced as showed by results in **Figure 7.24**.

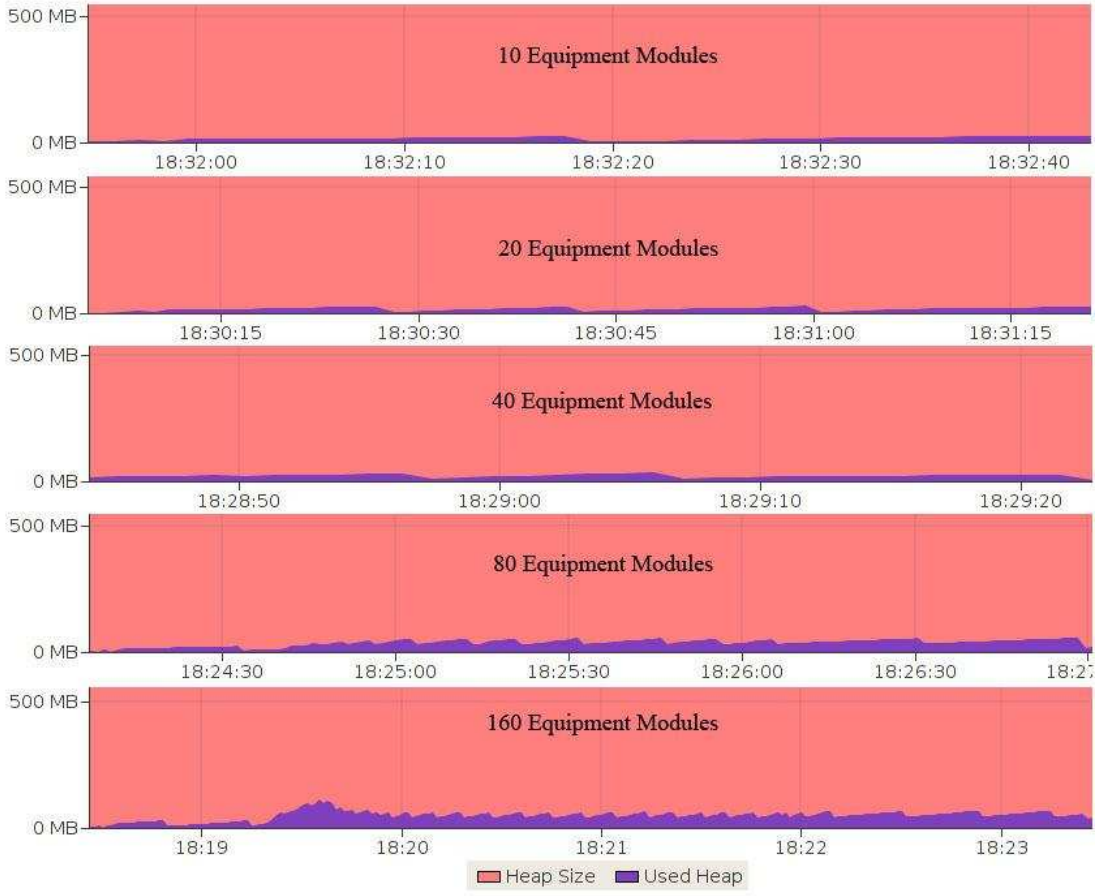
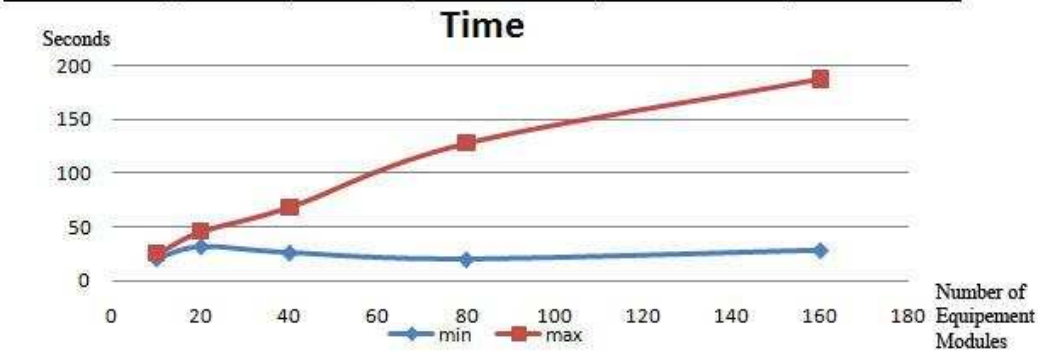


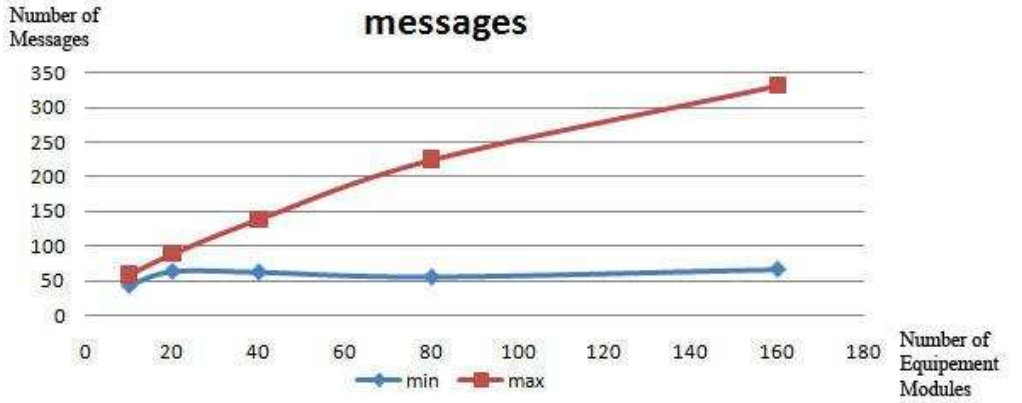
Figure 7.24 - Memory Consumption for Agent Interactions Restriction of 10

The results for computer's memory consumption with restriction on the number of interactions allowed for each agent to provide a significant improvement comparatively to the results shown in **Figure 7.22**. The comparison of the results for 160 equipment modules, which is the worst case in the presented scenario, shows 10 times less memory consumption, which is a huge impact.

The other performance results for this scenario are seen in **Figure 7.25**, and demonstrate not only the impact in the performance of the methodology but also the quality of the solutions in terms of their ranked value.

	Time for Solution		Exchange Messages for Solution		
Nº of Modules	Minimum	Maximum	Minimum	Maximum	Nº of Solutions
10	20.881	24.848	44	59	2
20	31.964	45.61	64	89	4
40	26.158	68.348	63	139	8
80	20.029	127.689	56	225	16
160	28.452	187.352	67	332	31





				Solution Value		
Nº of Modules	Time per Solution	Messages per Solution	Time Variance	Minimum	Maximum	Average
10	12.424	29.5	3.967	2.794667	3.055133	2.9249
20	11.4025	22.25	13.646	2.60575	3.2418	2.923775
40	8.5435	17.375	42.19	2.507	3.24105	2.909556
80	7.9805625	14.0625	107.66	2.5945	3.2373	2.888197
160	6.043612903	10.70967742	158.9	2.2965	3.2373	2.889513

Figure 7.25 - MAS Configuration Performance Results for Agent Interactions Restriction of 10

The comparison of these results and the results with no interaction restrictions will be presented later in this subchapter. Nevertheless, it is important to highlight a few aspects of the results shown in **Figure 7.25**. The most meaningful result is the loss of a potential solution for 160 equipment modules. Despite the results of these solutions being expected to vary, since there is a random element on the agent iterations, the

lack of one or two potential solutions was constantly obtained, when the restriction to 10 interactions was enforced to the methodology. Another interesting result is the fact that the number of messages and the time per solution actually decreased as the number of equipment modules increased. This is an interesting result since it indicates that under these conditions, the more variety exists, the faster and more effective configuration solutions are found.

The scenario of the computational performance limited to 20 interactions per agent is shown in **Figure 7.26**. The comparison of these results with the results provided in **Figure 7.24** show a small increase in memory consumption but not very significant.

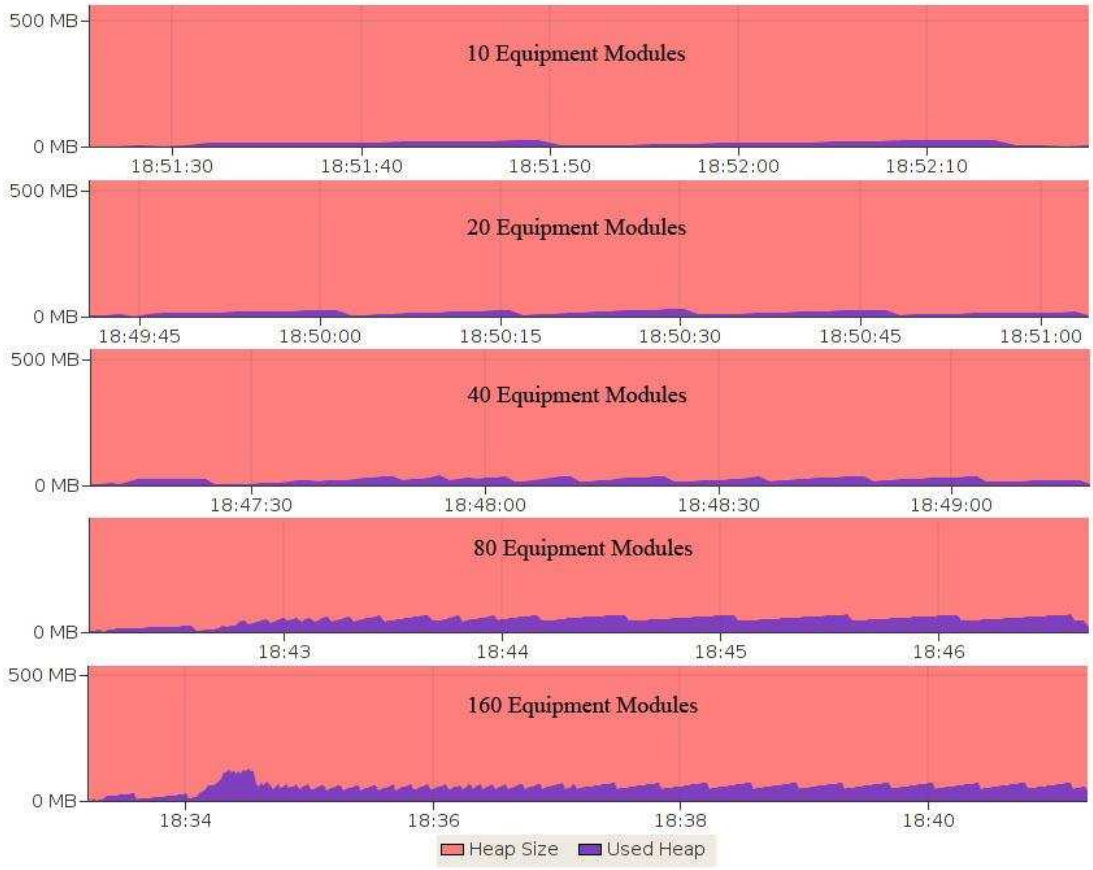
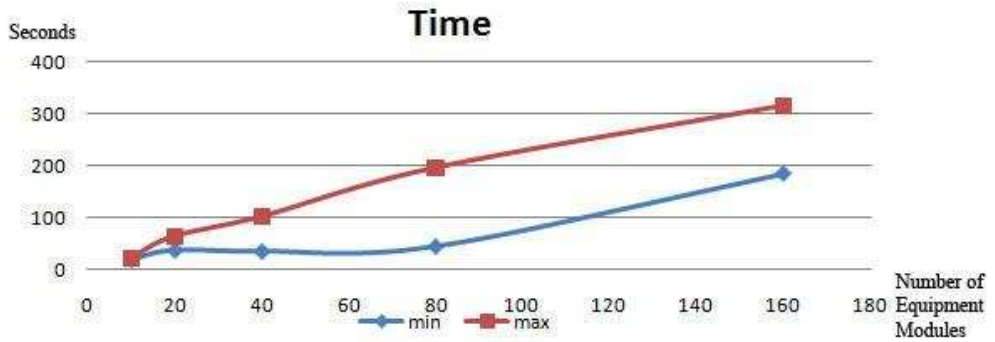
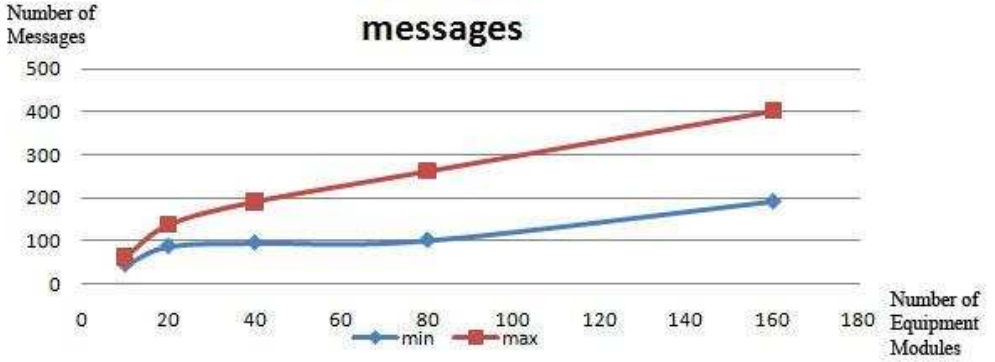


Figure 7.26 - Memory Consumption for Agent Interactions Restriction of 20

On the other hand, the impact of restricting the agent interactions to 20 on the performance characteristics, as seen in **Figure 7.27**, do provide an indication of some improvements in relation to the restriction of 10 interactions.

	Time for Solution		Exchange Messages for Solution		
Nº of Modules	Minimum	Maximum	Minimum	Maximum	Nº of Solutions
10	20.165	22.89	44	62	2
20	38.11	64.49	87	137	4
40	35.866	102.029	95	191	8
80	45.304	195.869	101	262	16
160	184.419	315.098	191	402	32





				Solution Value		
Nº of Modules	Time per Solution	Messages per Solution	Time Variance	Minimum	Maximum	Average
10	11.445	31	2.725	2.608	3.2418	2.9249
20	16.1225	34.25	26.38	2.60575	3.2418	2.923775
40	12.753625	23.875	66.163	2.60125	3.2418	2.921525
80	12.2418125	16.375	150.565	2.398133	3.2418	2.902975
160	9.8468125	12.5625	130.679	2.39825	3.2403	2.89297

Figure 7.27 - - MAS Configuration Performance Results for Agent Interactions Restriction of 20

The results achieved for the restriction to 20 interactions provided the same trends in terms of time per solution and message per solution as the results for the restriction to 10 interactions. However, there was a significant improvement, which is the number of potential solutions was always reached during all the performed tests.

The comparison among different results offers an overview of the performance and the overall behaviour of the architecture. This is particularly important when the number of potential solutions is high, therefore the analysis should focus on the datasets for 40, 80 and 160 equipment modules. **Figure 7.28** provides us with the

comparison of different results using the environment with and without restrictions in the number of agent interactions.

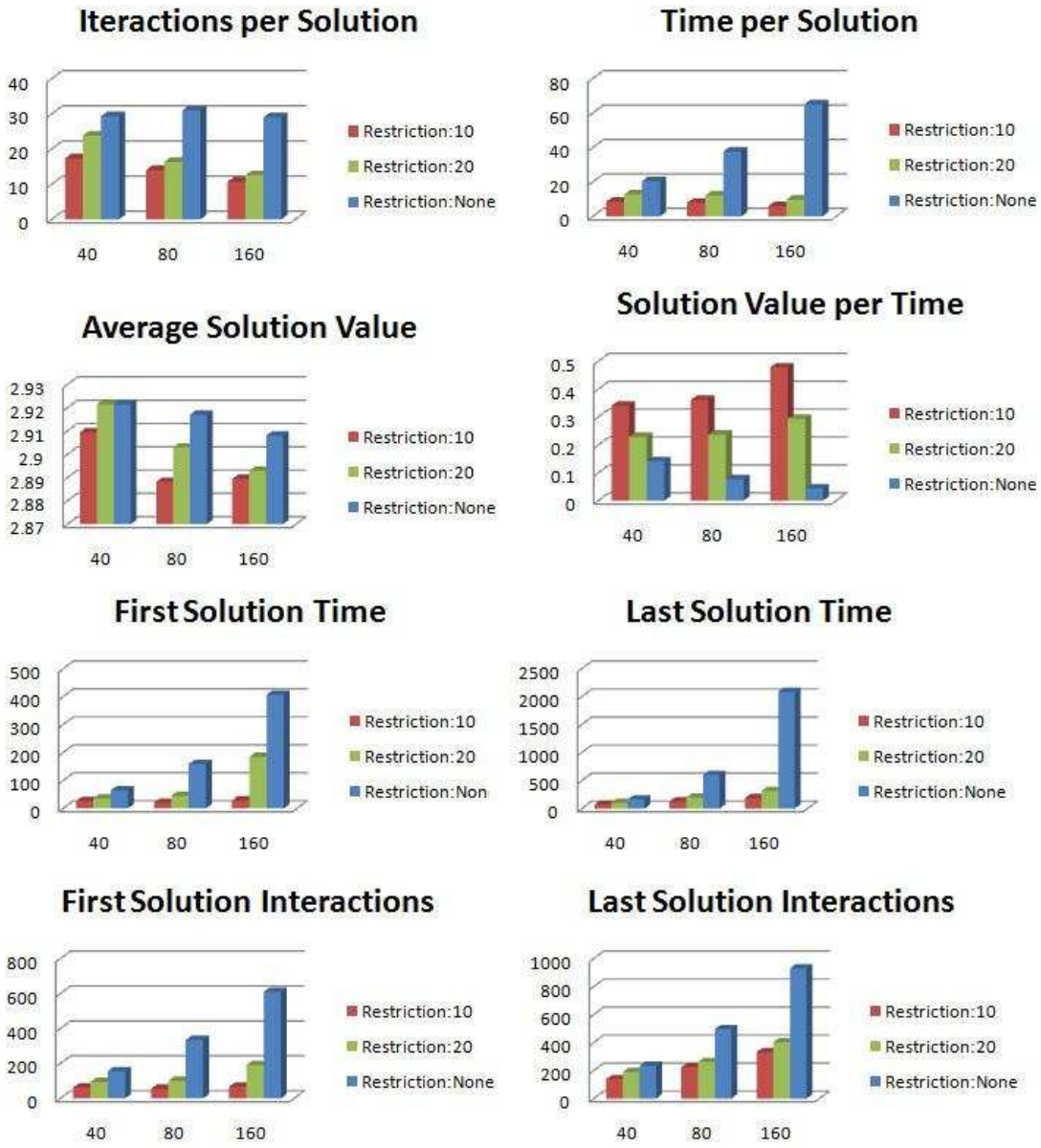


Figure 7.28 –Results Comparison for Tested Interaction Restriction in Agent Environment

The comparison of the results clearly shows that the restriction of the number of interactions has a positive impact in the memory consumption required by the self-configuration methodology. Moreover, it has also a positive impact on all the analysed performance characteristics except for one, the solution value. The performance of the methodology does increase significantly, but at the cost of not finding the best solutions. Nevertheless, the solution value per time clearly indicates that the impact of the reduction in quality is clearly surpassed by the impact on time.

The fact is for slightly better solutions the time cost is quite significant. This is more important for the last configuration solutions found in the system, since these take significantly longer with no restrictions.

7.3.4 Verification of Architecture Overall Behaviour Performance in Distributed Environment

The verification of the architecture overall behaviour performance in a distributed environment provides us with the results that enabled the conclusions on distributing the computational load of the configuration methodology. Moreover, it provides the technical validation that the architecture works in a distributed environment.

The biggest impact on distributing the computational load is the ability to execute more computational operations in a given timeframe. This would have a very positive result in the exhaustive approach, since it has the biggest computational load. This means that the impact of the distribution of the environment across different computers will be significantly higher in the worst case scenario considered. Therefore the verification of the performance in a distributed environment will focus on the scenario with no interaction restrictions. **Figure 7.29** provides an overview of the computation effort results for 160 modules for one of the computers in the defined scenario of distribution across three computers.

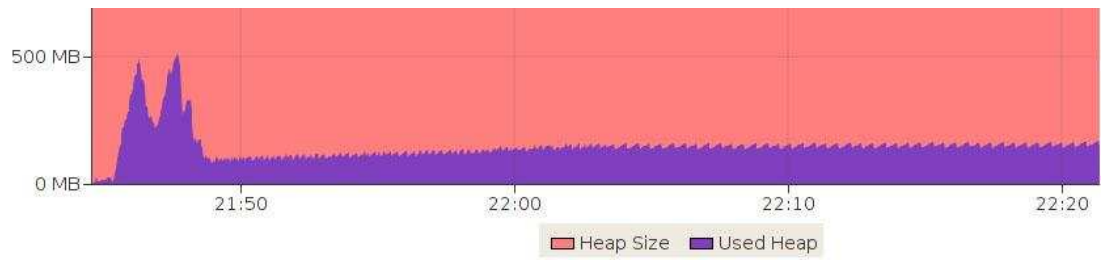


Figure 7.29 - Memory Consumption on One of the Computers used in the Distributed Scenario Testing (for 160 Equipment Module Agents distributed across two computers)

The comparison of these results with the ones obtained in **Figure 7.22** clearly shows that the distributed environment has a positive impact on the performance of the methodology. However, by comparing these results with the results provided in the no restrictions on interactions seen in **Figure 7.22**, it is clear an increase in time to reach all solutions. This increase can be justified by the use of a wireless network as a basis for these experiments, but also because the messaging across different

computers uses http which takes longer than normal messages between agents running in the same platform (FIPA [103]; FIPA [104]).

7.3.5 Analysis of Operational Validation Results of Agent Architecture for Distributed Self-Configuration Methodology for Modular Assembly Systems

The proposed agent architecture has demonstrated the ability to reach configuration solutions through the defined agent interactions. Therefore it is clear that the bottom up approach to achieve MAS configurations is viable through the use of this architecture. Furthermore, the architecture design caters for the future scalability of MAS, through the ability to distribute the load across different computers or by restricting the agent interactions.

The results of running the methodology with no interactions limitations and in a single computer have shown the expected exponential growth in the memory consumption when the pool of available agents increases. This highlights the need for options to cater for this as the main challenges of this approach. The architecture design does provide the means to deal with this issue, which was also tested.

The results for the restriction of the interactions between agents does result in the containment of the growth of the memory consumptions and has a positive impact on the speed solutions are found, however, this is done at the cost of the quality of the configuration solutions. As expected, the bigger the limitation on the interaction the faster solutions are found, but lesser is their value in relation to the potential best solutions. It is important to notice that results do show a significant reduction in time, but not a significant loss of quality of the solutions. This suggests that the approach is quite suitable.

The distribution of the environment across different computers does also produce the expected impact in the reduction of memory consumption. However, it comes with the negative side effect of a slight increase of the time to find solutions. This was expected since the messaging between agents on different platforms requires extra software libraries that do consume time in the making of the messages. Therefore, despite this being a solution to deal with the memory consumption problem, it is only thought to be the solution if time to reach solution is not a

problem. Considering the current configuration process time which takes weeks, the issue of time is not that significant. Nevertheless, if this becomes an issue in the future, one can use a combination of the distribution with the limitation of the agent interactions, since the architecture caters for this as well.

7.4 Operational Validation of Distributed Behaviour

The operational validation of the distributed behaviour will focus on the individual agent's behaviour. This entails the description of the agent decision making capabilities, which are triggered by the architectural defined interactions. The core of the configuration methodology lies in the equipment module agents. Therefore, this is the most important local behaviour for enabling the configuration architecture.

The requirements agent uses the same calculation model as the equipment module agents for ranking the configurations solutions. The difference lies in the established beliefs, which in the case of the requirements agent are defined by the system integrator. Because of this the verification of the local behaviour of the requirements agent can be seen as part of the verification of the equipment module agent's local behaviour. The difference lays in who defines the beliefs, which in the case of equipment module agents is the equipment supplier. So a step-by-step assessment of the decision-making process of an equipment module agent will provide the necessary results to validate both its operation in the requirements agent.

The other two agents provide support for the configuration decision-making process which is carried out by the equipment module agents. Because this task can be decoupled out of the methodology, its verification requires a scenario where more equipment module agents are involved, which would make the validation quite complex. Therefore, the verification of their contributions to the configuration methodology will be decoupled through the creation of an independent scenario where the potential for their use is highlighted.

7.4.1 Validation Scenario

The analysis of the operational validation of the distributed behaviour's objectives identifies the need for two scenarios. The first scenario is a subset of the scenario presented in **Sub-chapter 7.3.1**. The scenario is the same as defined there, however

here the focus will be on the perspective of one of the ten equipment modules. Manipulator 2 has been selected for a step-by-step decision-making verification of its behaviour. The use of the same scenario is intended to provide clarity on the overall decision-making process. All the other agents will have similar decision-making processes varying according to the interaction in their established beliefs.

The second scenario is quite different and it will focus on one of the performance's characteristics assessment, the repeatability. This choice provides a scenario to test both the performance simulation agent and the MAS expert agent using the characteristic that provides a more complex behaviour of the performance simulation method. The scenario consists of a workstation which is composed of three central modules: a feeder module, a conveyor module and a manipulator module. The workstation has an optional vision system module to demonstrate the results of the repeatability synthesis for two different setups, which will be suggested by the MAS expert agent. For simplicity's sake, the repeatability of the workstation will be treated as a two dimensional problem for this example. **Figure 7.30** provides an overview of this conceptual solution.

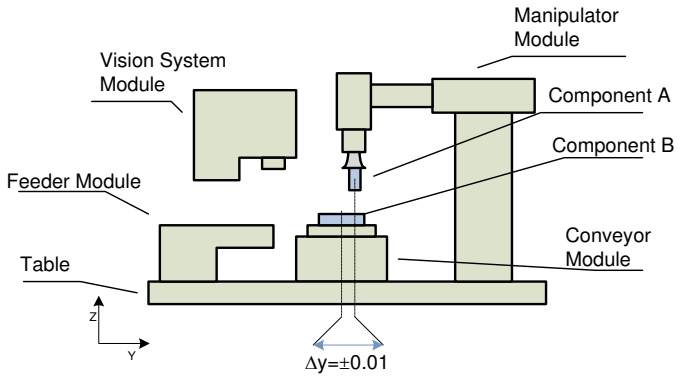


Figure 7.30 - Overview of Conceptual MAS Workstation

For this verification process, the manipulator module has been assumed to use a vacuum gripper and has a movement repeatability of: $\Delta y=\pm 0.01$ and $\Delta z=\pm 0.01$. It is also worth noting that due to the gripper type this equipment module compensates values only in the z axis. The feeder module supplies component A with a repeatability of: $\Delta y=\pm 0.05$. The conveyer module supplies component B with a repeatability of: $\Delta y=\pm 0.01$ and $\Delta z=\pm 0.01$. Finally the vision system module determines the location of the component A on the y axis with an accuracy of: $\Delta y=\pm 0.001$. The workstation will assemble the two components following the

assembly process sequence which highlights the assembly process classification proposed in **chapter 4**. This is shown in **Figure 7.31**.

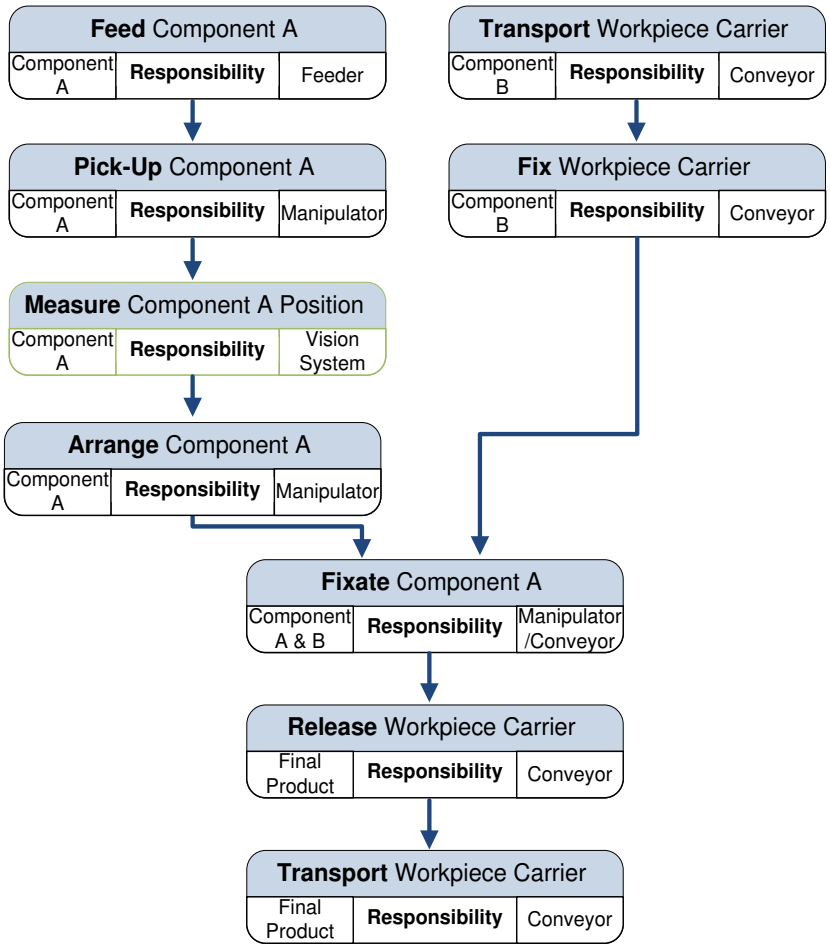


Figure 7.31 - Assembly Process Sequence Including the Required Information for Simulation

This scenario consists of a situation where the assessment of the performance characteristic fails to achieve the requirements after simulation due to the required repeatability. The equipment module agents involved in the configuration solution would contact the MAS Expert Agent which recommends, based on its knowledge, adding a measuring assembly process which would serve to compensate for the error in the repeatability. Then the equipment module agents would identify new collaborations for this new requirement. Once a new configuration solution that fulfils these extended requirements is available, the agents would be sent for simulation by the performance simulation agent.

7.4.2 Verification of Equipment Module Agent Local Behaviour

The verification of the equipment module agent local behaviour is done through the analysis of its interactions and decision-making process. This will provides insight into how the agent establishes collaborations and makes decisions. This was done using the Sniffer agent provided by the JADE platform in conjunction with debug tools.

The results for the simple scenario from the perspective of manipulator 2 show the reception of MAS configuration requirements. This was followed by the internal method to establish if the equipment module has any interest in his requirements. The agent identifies that it has an interest in data handling assembly process requirement, thus it expresses its interest to the requirements agent. Once this is done the manipulator 2, equipment module agent, waits for the requirements agent to send the list of potential interested agents.

The reception of the list with all interested agents is followed by the internal method that generates new requirements, which are the original requirements minus what this agent can do. In this scenario the results show that the agent generated requirements that maintain requirements for the feeding assembly process and gripping assembly process.

After the generation of the requirements this agent randomized the list of interested equipment module agents. The agent then checks the number of allowed contacts, which in this case comprises them all since the list is rather small. Nevertheless the agent follows the same procedure which verifies its behaviour under conditions that establish limits on the amount of contacts it can establish. This is followed by the sending of the new requirements to these agents and waiting for their answer. For each positive answer the agent updates an internal table that combines the different capabilities to assess if the original requirements are achievable. **Table 7.2** provides the core of the internal table for this agent after all agents have replied as well as the details for the value of each of the potential configuration solutions.

Table 7.2 - Manipulator 2 Equipment Module Agent Solution Table

Potential Solution Table			Solution Value	Rejections By:
Manipulator 2	Feeder 2	Gripper 1	2.700133333	Gripper 1
Manipulator 2	Gripper 1	Feeder 1	2.901	Gripper 1
Manipulator 2	Feeder 1	Gripper 2	2.51	
Manipulator 2	Feeder 2	Gripper 2	2.309133333	
Manipulator 2	Gripper 1	Feeder 3	2.716	Gripper 1
Manipulator 2	Feeder 3	Gripper 2	2.325	
Manipulator 2	Gripper 1	Feeder 4	3.055133333	Feeder 4/Gripper 1
Manipulator 2	Gripper 2	Feeder 4	2.664133333	Feeder 4
Manipulator 2	Feeder 1	Gripper 3	2.608	
Manipulator 2	Feeder 2	Gripper 3	2.407133333	
Manipulator 2	Feeder 3	Gripper 3	2.423	
Manipulator 2	Gripper 1			
Manipulator 2	Gripper 2			
Manipulator 2	Feeder 4	Gripper 3	2.762133333	Feeder 4
Manipulator 2	Feeder 1			
Manipulator 2	Feeder 2			
Manipulator 2	Feeder 3			
Manipulator 2	Feeder 4			

After all agents have replied the internal table is trimmed to remove incomplete solutions. This is followed by the request detailed information on the equipment module characteristics for each of the potential solutions. Once this information arrived, the agent proceeded with the calculation for the value of each solution according to its beliefs. This calculation involves the calculation of the assembly characteristics for each configuration solution, which is outside of requirements and makes the agent discard the potential solution. This is followed by the ranking method which for this very simple case selects all possibilities.

The agent then validates the potential solutions with MAS Expert agent, which for the given example validates them as “OK”. This is followed by the formal establishment of collaboration with all the agents in the potential configurations solutions. Once this is done the agents interact to deploy the performance simulation agent and provide them with descriptions for the simulation. Once the results were obtained the agent ranks the solutions based on the information received and its internal beliefs, as shown in **Table 7.2**.

The agent then contacts all agents in its top rank solution while putting on hold responses to requests of unique collaboration. In the given example, Manipulator 1, Gripper 1 and Feeder 4 agree on a collaboration therefore rejecting the collaboration requests made by Manipulator 2, as seen in **Table 7.2**. This means that Manipulator 2, equipment module agent, is rejected from his top rank solutions, which subsequently results in its configuration solution being only its seventh top choice. Once he gets a confirmation of all agents he contacts the requirements agent for proposing the configuration solution.

A final note for results was the omission of all intermediate steps information to the requirements agent.

7.4.3 Verification of Performance Simulation Model

The verification of the performance simulation model starts with the creation of the virtual Petri Net style network that is able to represent the given scenario. To do this the first step is the determination of the number of start tokens for the given workstation and assembly process. In this case two tokens are generated for the two assembly process flows as two component tokens. Each component token will be generated with an error matrix which will accumulate all the errors throughout the different process steps. The next step is to generate all the process placeholders, transitions and tokens for the equipment. The repeatability characteristics are assigned to the equipment tokens and process placeholders respectively. At the end of this process the complete repeatability performance simulation network for the given system is completed. The full network based on the model presented in **Chapter 6** can be seen in **Figure 7.32**. This already includes the network with the vision system as an optional assembly process. A brief analysis of the model provides us with a clear correlation with the assembly process sequence. It is important to note the classification of the processes within the model to understand its behaviour. The bases for the execution of the model are the transitions and the place holders. The behaviour of these is described in **Chapter 6**. For the purpose of simplification a section of the model has been highlighted for the description of these behaviours in **Figure 7.32**.

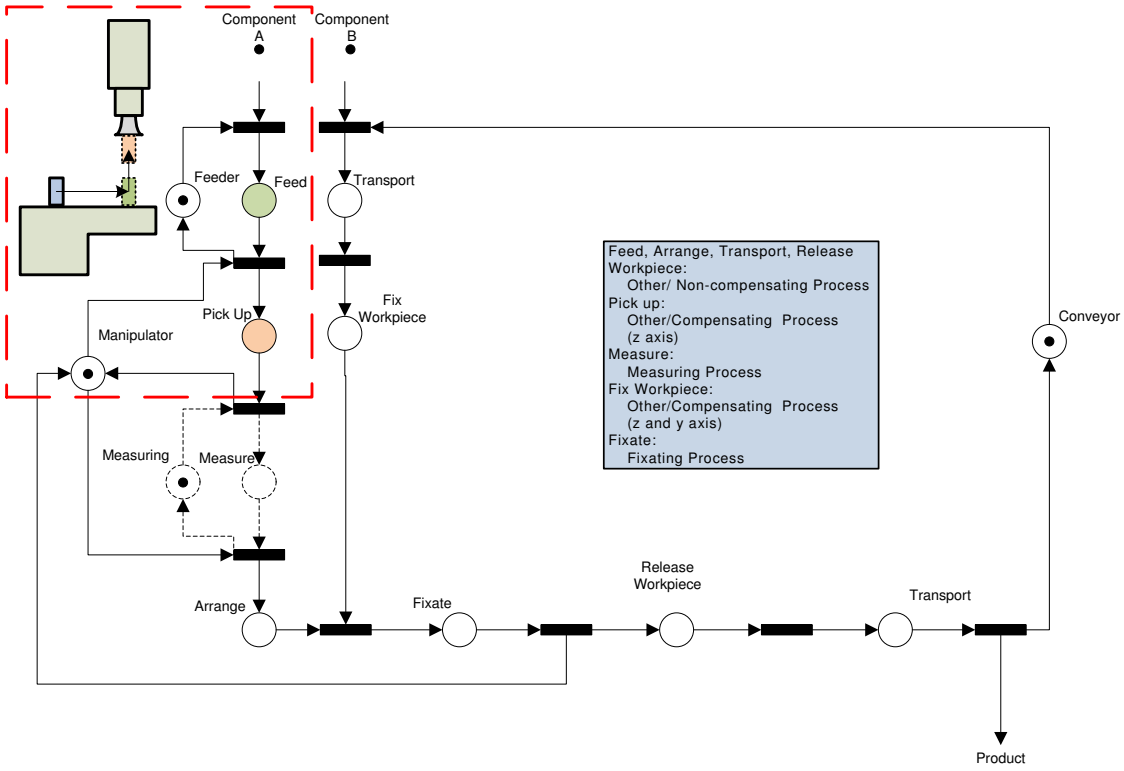


Figure 7.32 - Simulation Model for Given Scenario

In this description there is the correlation between the model and the physical system. Firstly component A is fed and thus moved to the pickup position. This is then followed by the pick up process performed by the manipulator. The underlining logic of the model requires the existence of a feeder token to enable the feed process and the manipulator token to enable the pick up process. **Table 7.3** goes through the different steps for this small section showing what the tokens contain and the different stages. The first transition requires the component A token and the feeder token. Once they are present both tokens move through to the feed place holder. In this place holder the relevant values are activated in the feeder token, for this example the y axis accuracy deviation.

Table 7.3 - Token Behaviour through Component First Steps

Step \ Token	Feeder Token	Component A Token	Manipulator Token
Transition 1	Δy=±0.05	Δy=±0.0	Δy=±0.01
Feed	Δy=±0.05	Δy=±0.0	Δy=±0.01
Transition 2	Δy=±0.05	Δy=±0.05	Δy=±0.01
Pick Up	Δy=±0.05	Δy=±0.05	Δy=±0.01
Transition 3	Δy=±0.05	Δy=±0.06	Δy=±0.01

Active Token	Inactive Token	Token Regeneration

The next transition requires the tokens from the feed place holder and the manipulator token. The manipulator token which was dormant is activated while the feeder token is stacked into the component error. Once the error has been passed on to the component, the feeder token is regenerated to enable the next component A. The pick up place holder operates in a similar fashion to the feed place holder. The manipulator token is affected and once this is done, this will trigger the next transition.

Once the system behaviour model is available, it can be run with random error values to simulate the emerging repeatability of the MAS configuration. The component tokens are passed through the model to accumulate all the errors until they are finally combined into one assembly represented by the product token. The results for both the repeatability of the workstation with and without the measurement system are given in **Figure 7.33**. Assuming that the output is required with six sigma accuracy, the derived repeatability of the workstation in y-direction is ± 13 µm and ± 52 µm with and without measurement system respectively.

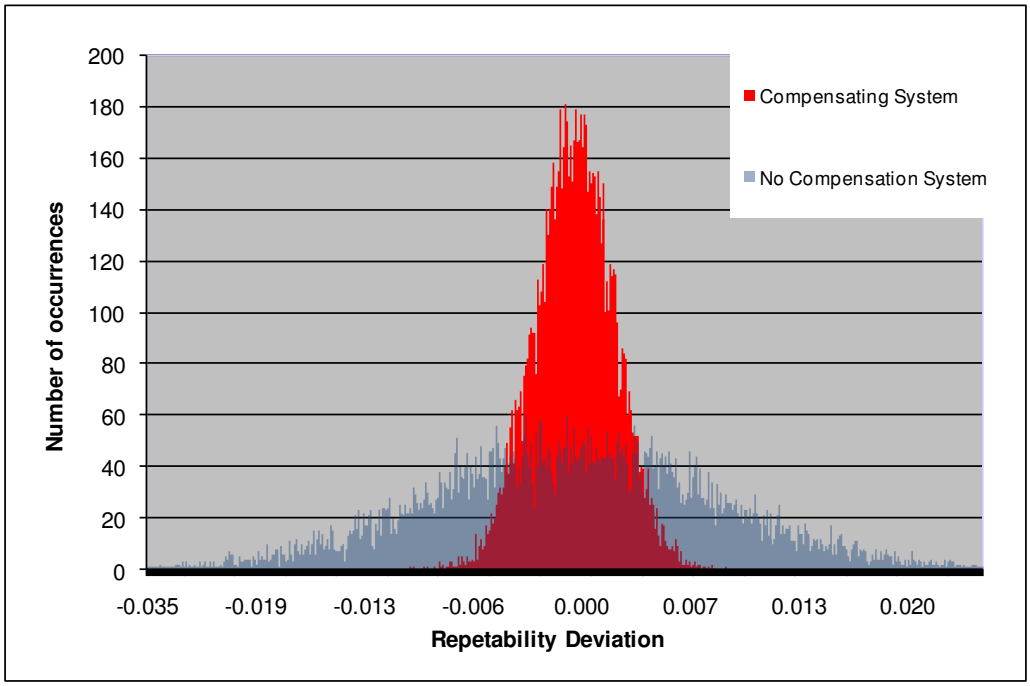


Figure 7.33 – Repeatability Simulation Results for y-direction

These results are then supplied to the individual equipment module agents, which use these in their models as described in **Chapter 6**, which finally result in the ranking of the configuration solutions.

7.4.4 Analysis of Operational Validation of Distributed Behaviour

The distributed local behaviour shows the implementation of the behaviours described in chapter 6. The results show that the implementation of the designed behaviour does provide the configuration methodology with the necessary decision making means to achieve solutions using the different scenarios. It is important to note that the results contained in **Subchapter 7.3** were obtained using the same distributed behaviour as the one described in this section. In this section, the results focused on important aspects of the proposed distributed local behaviour to better understand the inner workings of the proposed approach. In that sense the results clearly shown how decisions are reached using only local knowledge.

For the **Equipment Module Agent** it clearly shows how it needs to adjust to rejections of its preferred configuration solution, which is only possible due to the built in mechanisms not to discard potential solutions until it has made decisions.

The performance simulation results are presented in a non agent format to highlight the independence of the proposed model. The simulation of performance methodology was designed to work also outside of the configuration methodology in a standalone fashion. This provides it with the potential to be used in the current manual configuration process; the difference would only be in the deployment as the results show. The results focus on the repeatability aspects since this is the aspect that requires the highest complexity in the local behaviour. The results show the potential of this simulation methodology for early assessment of MAS configuration which can provide a big impact early in the configuration process to avoid mistakes at latter stages that are harder to correct.

7.5 Chapter Summary

In this chapter the validation scenarios and results of the Model for Agent-Based Self-Configuration of Modular Assembly Systems, Agent Architecture for Distributed Self-Configuration Methodology for Modular Assembly Systems and Behaviour of Distributed Self-Configuration Methodology are presented.

The Model for Agent-Based Self-Configuration of Modular Assembly Systems was validated using academic and industrial experts in the field, through its incorporation

in an expert tool as part of the EUPASS project. An illustrative example of this is presented in this chapter, which emphasises the potential of this model.

The chapter provides illustrative example of how the proposed Self-Configuration Methodology for Modular Assembly Systems works, while highlighting the impact of the restriction or non restriction of the number of agent interactions. The results for the performance of the methodology are presented and analysed.

Finally the chapter provides the details on the inner workings of the agents, providing illustrative examples of both the potential and the operation of the agents under the proposed configuration methodology.

8 Conclusion and Future Work

8.1 Introduction

In this chapter the conclusions for the work contained in this thesis are presented. The knowledge contributions will be highlighted and analysed. The chapter will also contain a perspective on future work as well as an overall future perspective of MAS.

This work targeted a main knowledge gap of the absence of a comprehensive approach for an automatic self-configuration methodology for MAS. Therefore the purpose of this work was to provide an automatic configuration methodology for MAS. To achieve this, a bottom-up approach using distributed decision making methods that enable the overall behaviour of finding MAS configuration solutions was developed. Moreover, it was identified that Agent Technology was the most suitable solution to fulfil this objective. The best practice codes of the use of agent technology have identified that the definition of the agent architecture is the basis for a successful solution. Subsequently, this work proceed as so, by designing an agent architecture that is described in **chapter 5**, where the overall behaviour of the agent's environment is defined, as well as the agent types, roles and overall interactions.

The specific models and methods for the local behaviour of each agent were also developed and are presented in **chapter 6**. These enable the actual decision making method for the agents to achieve configuration solutions. Furthermore, a new methodology was also introduced for early performance simulation of MAS

characteristics that can be used in conjunction with the configuration methodology or as a standalone contribution.

The achievement of an automatic configuration methodology also required a formal model that is able to accurately represent the requirements for any configuration methodology. This was addressed with the introduction of such a model contained in **chapter 4.**

Finally it is important to recap the main hypotheses of this work:

- If a structured and transparent model can be defined which formalises the physical and assembly process constraints of equipment model and a model that enables the definition of MAS requirements using the same concepts, it will be possible to establish automatic configuration methods. - The model described in Chapter 4 was used and it enabled the creation of an automatic configuration method.
- The self-configuration of MAS is better achieved through the use of a distributed bottom up approach.
- By creating a multi-agent solution for the bottom up solving of configuration problems, it will maximise the parallel computation and take advantage of negotiation protocols to achieve goal-oriented behaviour of the overall configuration environment.
- The collaboration of the agents using basic rules will enable the emergence of complex solutions.

8.2 Key Knowledge Contributions

The first fully distributed **Model for Agent-Based Self-Configuration of Modular Assembly Systems** was introduced, which caters not only for the definition of all the information required for the automatic configuration of MAS but also it establishes clear, structured and transparent means to describe MAS configuration solutions. This contribution enabled the development of the automatic configuration methodology but also has proved valuable in the current manual configuration process. It provides the means to exchange information between different tools if

required, namely in terms of early assessments of potential configuration solutions within the scope of the EUPASS project.

A new **Agent Architecture for Distributed Self-Configuration Methodology for Modular Assembly Systems** was introduced as a solution for a bottom up approach that enables the automatic configuration of MAS. This resulted from the analysis of the configuration process which led to the design of agents which accurately represent the different actors that are involved. The idea was to create virtual representation of the involved actors and define specific roles that they will carry out. This required a clear definition of agent types and individual roles for the creation of a multi agent environment that was able to provide solutions for the configuration of MAS. Furthermore, in the architecture it was also introduced a new agent organization model for MAS, as well as agent interactions that enable the overall behaviour of finding configuration solutions based on a set of requirements. Moreover, the possibility to restrict the number of interaction between agents was built into the solution, which enabled the approach to have either exhaustive or heuristic solutions that result in a huge increase in the performance at the cost of the quality of the configuration solutions.

New **Local Behaviour Models for Distributed Self-Configuration Methodology** were equally introduced and presented for the different agents defined in the architecture in **Chapter 6**. The main contribution of this chapter was the introduction of a formal method to establish the value of a given MAS configuration using performance characteristics. This method used a weighted approach which enabled different valuing of systems based on the knowledge of the different actors involved in the configuration process. The ability for changing the valuing of the system was viewed as one of the most important advantages of using a distributed approach, since different agents will have different beliefs and will act according to those. This method was incorporated into a new configuration method that was based on the exchange of information between agents and the establishment of a value for each of the configuration solutions which, in turn, are the basis for the decisions made by the each individual agent. Additionally a new normalization method for MAS characteristics was introduced, where it is possible to adjust the normalization function to reflect changes in trends, or successful solutions. This introduced a new approach to incorporate into the agents the ability to self-adjust its internal

mathematical model (beliefs) based on its prior experiences. Furthermore, this method also introduced a more accurate normalization, since it provided the possibility to have linear or exponential functions for normalizations

A new **Performance Simulation Methodology** was introduced which simulated repeatability, accuracy, cycle time and power consumption of MAS. This new methodology offered the basis for evaluation of these aspects to support the configuration and reconfiguration of MAS. In addition, a Petri net based model was adapted for the synthesis and simulation of a systems' behaviour.

8.3 Areas of Application

The results of this work are expected to be relevant for a wide range of applications in the MAS domain. The proposed models and methodology are likely to provide a big benefit for system integrators during the design and configuration of MAS. The models for describing the domain provide transparent means of exchanging information, which ultimately benefits all stakeholders of the MAS configuration process. This model has been already extensively used during the European project EUPASS and it is expected that its partners will continue to use this model on other endeavours.

The proposed configuration methodology is expected to be integrated in other MAS supporting tools. The literature clearly identifies that advances in supporting tools are expected in the near future. The openness and adaptability of the proposed method will allow it to be integrated into automatic configuration and deployment of solutions in the control domain.

The proposed MAS configuration methodology proposes an agent architecture that can be used on other modular domains to address configuration problems. Therefore it is expected that new input models will be developed to enable the use of the architecture concepts in other domains.

The proposed performance simulation methodology is expected to have a big impact in the early assessment of MAS configurations, since it can be used as a standalone methodology to support the current manual driven configuration process.

8.4 Critical Review

The work presented in this thesis provides a bottom up configuration methodology for MAS. This methodology was used and tested in a contained validation scenario and requires a wider dissemination to assess its true potential. The main drawback of using this methodology is the need for a common model that is shared by all involved parties, namely the system integrators and the equipment suppliers, which despite the current progress is still not a reality. Nevertheless, the introduction of such methodologies will provide better support to achieve a common model, since it provides a clear benefit.

The other drawback of this work is the fact that it targets a wide and complex domain, which makes it impossible for any person to establish a detailed model that covers all aspects. For the realisation of the vision of such system a conjoined effort from several people is required. The EUPASS project provided a step forward in this direction and all the work developed there also provides better support for the future of MAS. This is an area that has managed to bring together a variety of academics and industrialists sharing the same vision.

The main difficulty of doing research in this field is the aversion to change that one may encounter, even in unexpected places. Academics at times are as adverse to change as industrialists; this was one of the greatest learned lessons during this process.

Through experience the author realises that, it is highly recommended the discussion of ideas with other people, there is no better way to see the merit of an idea than to discuss it with others. This work has sparked several discussions, and yet, in insight one must conclude that more discussions would have helped to avoid mistakes. In the beginning of this work the lack of confidence prevented the posing of certain questions, or make suggestions which could have saved time and effort. This conclusion has led to the biggest lesson learned in this process; all ideas have merit, even if the merit is seeing the limitations of an idea. All in all, a PhD is supposed to make one think, and help one make others think. One should not be afraid to of criticism, good or bad, the more criticism, the better the chances to produced high

quality work. A big part of the quality of the work presented in this thesis is due to constructive criticism, which always leads to a constant improvement.

8.5 Future Work

The reported research provides a methodology for the self-configuration of MAS which is designed to be open to enhancements. Therefore it is only logical that the creation of this methodology provided a deep insight into potential future developments.

The introduction of the MAS Expert agent into the approach was only briefly addressed in this work to highlight its potential. The capture of expert knowledge and its incorporation into this agent is viewed as one of the things that will have the biggest impact in the future performance enhancement of the proposed configuration methodology. This will avoid the exploration of inaccurate solutions but also provide solutions to deal with problems affecting given sets of solutions, therefore salvaging solutions that otherwise get lost.

The possibility to restrict the number of interactions introduces the possibility to introduce criteria for interactions. This means the establishment of the concept of neighbourhoods into the agents which means the creation of preferential clusters. This is viewed as a possible avenue to explore for achieving better solutions once the solution's pool increases. However, it is not clear at this stage that this is the way forward; nevertheless it should be further investigated.

The scope of this work only allowed for restricted number of tests, therefore it is important to test the methodology further to better assess the proposed solution. Furthermore, future tests will provide a better understanding into the real impact of the agents' ability to adjust its internal beliefs, based on successful and unsuccessful solutions, in affect learning from past experience. It is expect that this work can be further tested within the context of the IDEAS project targets advancements in this domain (IDEAS [33]).

The proposed configuration methodology should be further tested to consider its performance with other industrial solutions. Further tests on the distribution of the

approach should be brought forward and the methodology should be extensively used in the MAS domain to further prove its value.

The integration of the proposed configuration methodology into multi level configuration approach, in affect breaking down the requirements into parts, might produce positive impact in achieving configuration solutions. Therefore it will be beneficial to explore this possibility in future works.

The performance simulation agent should be extended to include results on kinematics, namely collision assessments, work envelop operations, etc.

The MAS performance simulation should be explored as standalone solution, and should be further tested. Furthermore it should also be analysed against other simulation solutions to assess its true impact in the simulation domain.

8.6 Concluding Remarks

This work was motivated by the growing trend towards modular assembly system which is supported by the existence of a European project EUPASS which targeted such systems. The participation in this project was also quite valuable for the definition of the main knowledge gap, the lack of an automatic MAS configuration method. The analysis of the modular concept provided a clear possibility for the creation of a methodology that was able to cater for this. Furthermore, it was also clear that a lack of formal description for such systems was still a big concern in the domain. This led to the conclusion, that in order to have a MAS configuration method, one would need to formalise the required information for such a methodology. It was clear if that information could be formalized then it would be possible to analyse the configuration process to create an automatic solution.

The definition of a model to capture the required information was one of the challenges of the EUPASS project. In the project a more comprehensive model was required due to project aims and objectives. The model proposed in this work is a lightweight model that was built using the knowledge obtained throughout the participation in the project and the interaction with all its partners. The model was used and proved useful for the exchange of information between different stakeholders in the project, which provides, in my opinion, the best validation

possible for such a model. Furthermore, the creation of this model was the main enabling factor for the definition of a MAS configuration methodology as stated in the research hypothesis.

The analysis of the MAS concept clearly identified a problem of complexity and future scalability. On one hand, the complexity of the domain made it quite complicated to come up with an overall top down solution. In addition to this, the fact that the domain is in constant evolution would render any top-down solution potentially useless. Therefore it was decided that a bottom up approach would be better to tackle the problem. Furthermore, the nature of modular system is quite in line with this type of approach.

The use of agent technology was the natural way forward to implement the concepts of a bottom up approach. Agent technology has been extensively used in literature to tackle a series of different problems, as described in **Chapter 2**. An analysis of this lead to the identification of an appropriate multi agent environment design methodology. This led to the breakdown of the problem into the agent architecture and the distributed local behaviour. The proposed architecture was designed to cater for future enhancements, since the domain is expected to evolve.

The research hypothesis states that the multi agent solution would be able to provide a bottom up self-configuration methodology for MAS. In line with the hypothesis, this work provided a first approach to create a solution for the configuration of MAS using a multi agent solution. Furthermore, this work was designed in order to cater for future enhancements. Subsequently, this was always viewed as an evolving approach as one could expect from an evolving domain. Importantly, this work provided the first steps in the creation of a comprehensive MAS configuration methodology that is able to deal with all aspects of the MAS domain.

The proposed work is viewed to have a big impact on the MAS configuration process, which is still mostly a manual process. This work provided the means to support the process by providing possible configuration solution and ranking them based on the weights the system integrator attributes to the system performance characteristics. This is deemed as a big step forward speciality considering the growing number of available equipment modules, which makes the task of checking all possibilities manually virtually impossible.

The choice for a distributed approach and the use of agent technology does, however, pose a limitation to the system, which is the need for agent platforms running agents across computers. Ideally these computational resources would be supplied by equipment module vendors running their agents; however this is not a reality at the moment. In fact the implementation of such a system would require addressing a lot of security issues that agent technology has. Nevertheless, it is still possible to run such a system in places where the security of the network is not essential, and the methodology will provide the necessary support for the configuration of MAS.

The solution presented did not analyse all aspects of the configuration process but it is viewed as a significant contribution for the reduction of the MAS configuration process time, which is significantly important with the growing number of equipment modules.

References

1. Shen, W., D.H. Norrie, and J.-P.A. Barthès, Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing. 2001, London and New York: Taylor & Francies.
2. Koren, Y., et al., Reconfigurable Manufacturing Systems. CIRP Annals - Manufacturing Technology, 1999. **48**(2).
3. Arai, T., et al., Holonic assembly system with Plug and Produce. Computers in Industry, 2001. **46**: p. 289-299.
4. EUPASS. Evolvable Ultra-Precision Assembly Systems. 2008 [cited; Available from: http://www.eupass-fp6.org/.
5. NACFAM, Potentially disruptive Advanced Manufacturing Technologies. 2003, NACFAM.
6. Kratochvíl, M. and C. Carson, Growing Modular. 2005, Berlin: Spinger
7. Onori, M., et al., European precision assembly roadmap 2012. The Assembly-Net Consotium, 2002.
8. Hollis, R.L. and A. Quaid, An Architecture for Agile Assembly, in Proc. Am. Soc. of Precision Engineering. 1995: Austin.
9. Alsterman, H. and M. Onori. Process-Oriented Assembly System Concepts - The MarkIV Approach. in ISATP 2001. 2001. Fukuoka, Japan.
10. Gaugel, T., M. Bengel, and D. Malthan, Building a mini-assembly system from a technology construction kit. Assembly Automation, 2004. **24**(1): p. 43-48.
11. Jennings, N.R. and M.J. Wooldridge, Applications of intelligent agents. Agent Technology: Foundations, Applications, and Markets, ed. E. N.R. Jennings and M.J. Wooldridge. 1998: Springer
12. Bi, Z.M., et al., Reconfigurable manufacturing systems: the state of the art. International Journal of Production Research, 2007. **46**(4): p. 967-992.
13. Bi, Z.M., et al., Development of reconfigurable machines The International Journal of Advanced Manufacturing Technology, 2007.
14. Chryssolouris, G., Manufacturing Systems - Theory and Practice. Second Edition ed. 2006, New York: Springer.
15. Bi, Z.M., L. Wang, and S.Y.T. Lang, Current status of reconfigurable assembly systems. International Journal of Manufacturing Research, 2007. **2**(3): p. 303 - 328.
16. Bellgran, M. and C. Johansson, A method for the design of flexible assembly systems. International Journal of Production Economics, 1995. **41**: p. 93-102.
17. Bukchin, J., E.M. Dar-El, and J. Rubinovitz, Team-oriented assembly system design: A new approach. European Journal of Operational Research, 1997. **156**: p. 326-352.
18. Onori, M. and J. Oliveira, Outlook report on the future of European assembly automation. Assembly Automation, 2010. **30**: p. 7 - 31.

19. Mehrabi, M.G., A.G. Ulsoy, and Y. Koren, Reconfigurable manufacturing systems and their enabling technologies. Int. J. Manufacturing Technology and Management, 2000. **1**(1): p. 114–131.
20. Edmondson, N.F. and A.H. Redford, Generic flexible assembly system design. Assembly Automation, 2002. **22**(2): p. 139 - 152.
21. Yusuf, Y.Y., M. Sarhadi, and A. Gunasekaran, Agile manufacturing: the drivers, concepts and attributes. International Journal of Production Economics, 1999. **62**: p. 33-43.
22. Michelini, R.C., et al., Computer-Integrated Assembly for Cost Effective Developments, Computer-Aided Design, Engineering and Manufacturing – Systems techniques and applications, Vol. II: Computer Integrated Manufacturing Boca Raton, FL. 2001: CRC Press LLC.
23. Weber, A., Is Flexibility a Myth? Assembly, 2004. **May**: p. 50-59.
24. Tichem, M., Position report on flexible assembly automation. Laboratory for Production Engineering and Industrial Organisation Delft University of Technology, Landbergstraat 3, NL-2628 CE Delft. The Netherlands, 2000. **January**.
25. Feldmann, K. and S. Slama, Highly flexible assembly – scope and justification. Annals of the CIRP, 2001. **50**(2): p. 489–498.
26. Bodine, W.E., Making agile assembly profitable. Manufacturing Engineering, 1998. **121**(4): p. 60–68.
27. Ulrich, K., The role of product architecture in the manufacturing firm. Research Policy, 1995. **24**(3): p. 419-440.
28. Martin, M.V. and K. Ishii, Design for Variety: Developing Standardized and Modularized Product Platform Architectures. Research in Engineering Design 2002. **13**: p. 213-233.
29. Gunnar, E., Modular function deployment – a method for product modularisation. 1998, Royal Institute of Technology: Stockholm.
30. Blackenfelt, M. and R.B. Stake. Modularity in the context of product structuring – a survey. in The Second Nord Design Seminar, KTH, Stockholm. 1998.
31. Bi, Z.M. and W.J. Zhang, Concurrent optimal design of modular robotic configuration. Journal Robot Systems, 2000. **18**(2): p. 77–87.
32. Heisel, U. and M. Meitzner, Process in reconfigurable manufacturing systems, in Second International CAMT Conference. 2003: Wroclaw. p. 129-136.
33. IDEAS, Instantly Deployable Evolvable Assembly Systems. 2010.
34. Arai, T., et al., *Agile Assembly System by "Plug and Produce".* CIRP Annals - Manufacturing Technology, 2000. **49**(1): p. 1 - 4.
35. Vyatkin, V., IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design. 2007: ISA.
36. Boër, C.R., et al., Integrated Computer Aided Design for Assembly Systems. Annals of the CIRP, 2001. **50**(1): p. 17-20.
37. Chen, I.-M., Rapid response manufacturing through a rapidly reconfigurable robotic cell. Robotics and Computer Integrated Manufacturing, 2001. **17**: p. 199-213.
38. Giusti, F., et al., A Reconfigurable Assembly Cell for Mechanical Products. Annals of the CIRP, 1994. **43**(1): p. 1-4.

39. EasterBrook, S.M., Elicitation of Requirements from Multiple Perspectives, in Imperial College of Science, Technology and Medicine. 1991, University of London: London.
40. Hirani, H.J., Knowledge Based Requirements Specification for Reconfigurable Assembly Systems, in M3. 2005, University of Nottingham.
41. Faulkner, D., B. Levy, and T. Garner, Open-architecture platforms. Circuits Assembly, 1999. **January**: p. 50-55.
42. Grondahl, P. and M. Onori, Standardised flexible automatic assembly - evaluating the mark IV approach. Assembly Automation, 2000. **20**(3): p. 243-253.
43. Gardan, N. and Y. Gardan, An application of knowledge based modelling using scripts. Expert Systems with Applications, 2003. **25**: p. 555-568.
44. Lohse, N., Towards an Ontology Framework for the Integrated Design of Modular Assembly Systems. 2006, University of Nottingham.
45. Perremans, P., Feature-based description of modular fixturing elements: the key to an expert system for the automatic design of the physical fixture. Advances in Engineering Software 1996. **25**(1): p. 19 - 27
46. Tsai, Y.-T. and K.-S. Wang, The development of modular-based design in considering technology complexity. European Journal of Operational Research., 1999. **119**: p. 692–703.
47. Levin, M.S., Towards combinatorial analysis, adaptation, and planning of human-computer systems. Applied Intelligence, 2002. **16**(3): p. 235 - 247
48. Hong, N.K. and S. Hong, Entity-based models for computer-aided design systems. Journal of computing in civil engineering, 1998. **12**(1): p. 30 - 41.
49. Watson, I., Case-based reasoning is a methodology not a technology. Knowledge-Based Systems, 1999. **12**(303 - 308).
50. Paredis, C.J.J. and P.K. Khosla, Serial link manipulators from task specifications. Int. Robot. Res., 1993. **12**(3): p. 274–287.
51. Chen, I.-M. and J.W. Burdick. Determining task optimal modular robot assembly configurations. in IEEE International Conference on Robotics and Automation. 1995. Nagoya, Aichi, Japan.
52. Sims, K., Evolving virtual creatures, in International Conference on Computer Graphics and Interactive Techniques. 1994. p. 15 - 22.
53. Parunak, H.V.D., Workshop Report: Implementing manufacturing agents, in Sponsored by the SFA project of NCMS in conjunction with *PAAM'96*. 1996, NCMS.
54. Bi, Z.M., On Adaptive Robot Systems for Manufacturing Applications. 2002, University of Saskatchewan: Canada.
55. Son, S.Y., Design principles and methodologies for reconfigurable machining systems. 2000, University of Michigan: Ann Arbor.
56. Spicer, J.P., A design methodology for scalable machining systems. 2002, University of Michigan Ann Arbor.
57. Tang, L., et al., Concurrent Line-Balancing, Equipment Selection and Throughput Analysis for Multi-Part Optimal Line Design. The International Journal for Manufacturing Science & Production 2004. **6**: p. 71-81.
58. Zhao, X., K. Wang, and Z. Luo, A stochastic model of a reconfigurable manufacturing system Part I, a framework. International Journal of Production Research, 2000. **38**(10): p. 2273–2285.
59. Ohiro, T., et al. A stochastic model for deciding an optimal production order and its corresponding configuration in a reconfigurable manufacturing

system with multiple product groups. in International Conference on Agile, Reconfigurable Manufacturing. 2003. Ann Arbor.
60. Oliveira, J.A.B.d., Coalition Based Approach for Shop Floor Agility: MultiAgent Approach. 2005, Amadora: Edições Orion.
61. Maturana, F., W. Shen, and D.H. Norrie, MetaMorph: an adaptive agent-based architecture for intelligent manufacturing. International Journal of Production Research, 1999. **37**(10): p. 2159 - 2173.
62. Nwana, H.S., Software Agent: An Overview. Knowledge Engineering Review, 1996. **11**(2): p. 205 - 244.
63. Ferber, J., Multi-Agent Systems: An introduction to distributed Artificial Intelligence. 1999, London: Addison-Wesley.
64. Shen, W., et al., Applications of agent-based systems in intelligent manufacturing: An updated review. Advanced Engineering Informatics 2006. **20**(4): p. 415 - 431
65. Jennings, N.R., J.M. Corera, and I. Laresgoiti. Developing industrial multi-agent systems. in *Proc. of ICMAS'95*. 1995. San Francisco, CA, .
66. Shen, W. and D.H. Norrie, Agent-based systems for intelligent manufacturing: a state-of-the-art survey. KAIS 1, 1999. **2**: p. 129–156.
67. Wooldridge, M. and N. Jennings, Intelligent agents: Theory and practice, . The Knowledge Engineering Review 1995. **10**(2): p. 115–152.
68. Deen, S.M., Agent-Based Manufacturing – Advances in the Holonic Approach. 2003, Heidelberg, Germany Springer-Verlag.
69. Azevedo, A., C. Torscano, and J.P. Sousa. An order planning system to support networked supply chains. in Proc. of PRO-*VE'02*. 2002.
70. Barry, J., et al. NIIIP-SMART: an investigation of distributed object approaches to support MES development and deployment in a virtual enterprise. in *Proc. of EDOC'98*. 1998. La Jolla, CA.
71. Fox, M.S., J.F. Chionglo, and M. Barbuceanu, The integrated supply chain management system, Internal Report. 1993, Univ. of Toronto.
72. McEleney, B., G.M.P. O'Hare, and J. Sampson. An agent-based system for reducing changeover delays in a job-shop factory environment. in Proc. of *PAAM'98*. 1998. London, UK.
73. Peng, Y., et al., A multi-agent system for enterprise integration, in Proc. of *PAAM'98*. 1998: London, UK. p. 155–169.
74. Sadeh, N., D.W. Hildum, and D. Kjenstad, Agent-based e-supply chain decision support. Journal of Organizational Computing and Electronic Commerce, 2003. **33** (3–4): p. 225–241.
75. Shen, W., F. Maturana, and D.H. Norrie, MetaMorph II: an agent-based architecture for distributed intelligent design and manufacturing. Journal of Intelligent Manufacturing, 2000. **11**(3): p. 237-251.
76. Yen, B.P.C. and O.Q. Wu, Internet scheduling environment with market driven agents. IEEE TSMC-A 2003. **34**(2): p. 281–289.
77. Butler, J. and H. Ohtsubo, ADDYMS: Architecture for distributed dynamic manufacturing scheduling, in Artificial Intelligence Applications in Manufacturing, A. Famili, D.S. Nau, and S.H. Kim, Editors. 1992, The AAAI Press. p. 199–214.
78. McDonnell, P., et al., A Cascading Auction Protocol as a Framework for Integrating Process Planning and Heterarchical Shop Floor Control. International Journal of Flexible Manufacturing Systems, 1999. **11**(1): p. 37-62.

79. Parunak, H.V.D., A. Baker, and S. Clark. The AARIA agent architecture: From manufacturing requirements to agent-based system design. in Working Notes of the ABM Workshop. 1998. Minneapolis, MN.
80. Shen, W. and D.H. Norrie, Dynamic manufacturing scheduling using both functional and resource related agents. ICAE, 2001. **8**(1): p. 17–30.
81. Lu, T.-P. and Y. Yih, An agent-based production control framework for multiple-line collaborative manufacturing. International Journal of Production Research, 2001. **39**(10): p. 2155 - 2176.
82. Usher, J.M., Negotiation-based routing in job shops via collaborative agents. Journal of Intelligent Manufacturing, 2003. **14**(5): p. 485-499.
83. Wang, C., W. Shen, and H. Ghenniwa, An adaptive negotiation framework for agent-based dynamic manufacturing scheduling, in IEEE SMC 2003, P.o. 2003, Editor. 2003: Washington DC, USA. p. 1211–1216.
84. Bremer, C.F. and W.M. Molina, Global virtual business – a systematic approach for exploiting business opportunities in dynamic markets. IJAM, 1999. **1**(12).
85. Nigro, G.L., et al., Coordination policies to support decision making in distributed production planning. RCIM 2003. **19**(6): p. 521–531.
86. Hao, Q., et al., Towards an Internet enabled cooperative manufacturing management framework, in Proc. of PRO-*VE'03*. 2003: Lugano, Switzerland. p. 191–200.
87. Parunak, H.V.D., A.D. Baker, and S.J. Clark, The AARIA Agent Architecture: From Manufacturing Requirements to Agent-Based System Design Integrated Computer-Aided Engineering 2001. **8**(1): p. 45 - 58
88. Camarinha-Matos, L.M.A., H.; Rabelo, R. J., Infrastructure developments for agile virtual enterprises. Journal of Computer Integrated Manufacturing, 2003. **16**(4 - 5): p. 235-254.
89. Leeuwen, E.H.v. and D. Norrie, Holons and holarchies. Manufacturing Engineer, 1997. **76**(2): p. 86-88.
90. Bussmann, S., An agent-oriented architecture for holonic manufacturing control, in Proc. of IMS1998. 1998: Lausanne, Switzerland. p. 1–12.
91. Burke, P. and P. Prosser, The distributed asynchronous scheduler, in Intelligent Scheduling,, M. Zweben and M.S. Fox, Editors. 1994, Morgan Kaufman Publishers: San Francisco, CA p. 309–339.
92. Fischer, K., The design of an intelligent manufacturing system, in Proc. of *CKBS'94*. 1994: University of Keele, England,. p. 83–99.
93. McGuire, J., et al., SHADE: Technology for knowledge-based collaborative engineering. CEAR, 1993. **1**(3).
94. Petrie, C., et al., Next-Link: An experiment in coordination of distributed agents, in Position paper for the AID-94 Workshop on Conflict Resolution. 1994: Lausanne, Switzerland,.
95. Ouelhadj, D., C. Hanachi, and B. Bouzouia, Multi-agent system for dynamic scheduling and control in manufacturing cells, in Working Notes of the ABM Workshop. 1998: Minneapolis, MN. p. 96–105.
96. Shen, W. and J.-P. Barthès, An experimental multi-agent environment for engineering design. IJCIS, 1996. **5**(2–3): p. 131–151.
97. Babayan, A. and D. He, Solving the n-job 3-stage flexible flowshop scheduling problem using an agent-based approach. International Journal of Production Research, 2004. **42**(4): p. 777 - 799.

98. Kraus, S., Strategic negotiation in multiagent environments. 2001, Cambridge: MIT Press.
99. Lesser, V.R., Reflections on the Nature of Multi-Agent Coordination and Its Implications for an Agent Architecture Autonomous Agents and Multi-Agent Systems, 1998. **1**(1).
100. Rosenschein, J.S. and G.Z. Zlotkin, Rules of encounter: designing conventions for automated negotiation among computers. 1994, Cambridge: MIT Press.
101. Fatima, S.S., M. Wooldridge, and N.R. Jennings, An agenda-based framework for multi-issue negotiation. Artificial Intelligence, 2004. **152**.
102. Finin, T., Y. Labrou, and J. Mayeld, KQML as an agent communication language, in Conference on Information and Knowledge Management 1994, ACM: Gaithersburg.
103. FIPA, F.f.I.P.A., FIPA ACL Message Structure Specification. 2002, FIPA: Geneva.
104. FIPA, F.f.I.P.A., FIPA Abstract Architecture Specification. 2002, FIPA: Geneve.
105. Krothapalli, N.D., Abhijit Design of Negotiation Protocols for Multi-Agent Manufacturing Systems. International Journal of Production Research, 1999. **37**(7).
106. Smith, R.G., The contract net protocol: high-level communication and control in a distributed problem solver. IEEE Transactions on Computers, 1980. **C-29**(12): p. 1104–1113.
107. Duffie, N.A. and R.S. Piper, Non-hierarchical control of manufacturing systems. JMS 1986. **5** (2): p. 137–139.
108. Parunak, H.V.D., Manufacturing experience with the contract net, in Distributed Artificial Intelligence, M.N. Huhns, Editor. 1987, Pitman. p. 285–310.
109. Ow, P.S. and S.F. Smith, A cooperative scheduling system, in Proc. of the First International Conference on Expert Systems and the Leading Edge in Production Planning and Control. 1988. p. 43–56.
110. Shaw, M.J., Dynamic scheduling in cellular manufacturing systems: A framework for networked decision making. Journal of Manufacturing Systems, 1988. **7**(2): p. 83-94.
111. Saad, A., et al., Evaluation of contract net-based heterarchical scheduling for flexible manufacturing systems, in Proc. of Intelligent Manufacturing *Workshop at IJCAI'95*. 1995: Montreal, QC. p. 310–321.
112. Baker, A.D., Manufacturing control with a market-driven contract net. 1991, Rensselaer Polytechnic Institute: NY, USA.
113. Lin, G.Y.-J. and J.J. Solberg, INTEGRATED SHOP FLOOR CONTROL USING AUTONOMOUS AGENTS. IIE Transactions, 1992. **24**(3): p. 57 - 71.
114. Matos, N., C. Sierra, and N.R. Jennings. Determining Successful Negotiation Strategies: An evolutionary Approach. in Proceedings of the 3rd International Conference on Multi-Agent Systems 1998: IEEE Computer Society
115. Rahwan, I., et al., Stratum: A methodology for designing heuristic agent negotiation strategies. International Journal of Applied Artificial Intelligence, 2007. **26**.

116. Guan, Z., et al., Application of decentralized cooperative problem solving in dynamic flexible scheduling, in Proc. of SPIE. 1995: Bellingham, WA, . p. 179–183.
117. Henderson-Sellers, B.G., Paolo, Agent-Oriented Methodologies. 2005, Hershey, USA: Idea Group Publishing.
118. Bernon, C., et al., Engineering Adaptive Multi-Agent Systems: The ADELFE Methodology, in Agent-Oriented Methodologies, B. Henderson-Sellers and P. Giorgini, Editors. 2005, Idea Group Publishing: London, UK. p. 172-202.
119. Cossentino, M., From Requirements to Code with the PASSI Methodology, in Agent-Oriented Methodologies, B. Henderson-Sellers and P. Giorgini, Editors. 2005, Idea Group Publishing: London, UK. p. 79-106.
120. Garijo, F.J., J.J. Gomez-Sanz, and P. Massonet, The MESSAGE Methodology for Agent-Oriented Analysis and Design, in Agent-Oriented Methodologies, B. Henderson-Sellers and P. Giorgini, Editors. 2005, Idea Group Publishing: London, UK. p. 203-235.
121. Iglesias, C. and G. Mercedes, The Agent-Oriented Methodology MAS-CommonKADS, in Agent-Oriented Methodologies, B. Henderson-Sellers and P. Giorgini, Editors. 2005, Idea Group Publishing: London, UK. p. 46-78.
122. Padgham, L. and M. Winikoff, Prometheus: A Practical Agent-Oriented Methodology, in Agent-Oriented Methodologies, B. Henderson-Sellers and P. Giorgini, Editors. 2005, Idea Group Publishing: London, UK. p. 107-135.
123. Zambonelli, F., N.R. Jennings, and M. Wooldridge, Multi-Agent Systems as Computational Organizations: The Gaia Methodology, in Agent-Oriented Methodologies, B. Henderson-Sellers and P. Giorgini, Editors. 2005, Idea Group Publishing: London, UK. p. 136-171.
124. Sugumaran, V., Application of agents and intelligent information technologies. 2007, Oakland University, USA: Idea Group Publishing.
125. Inohira, E., A. Konno, and M. Uchiyama, Layered Multi Agent Architecture with Dynamic Reconfigurability, in loternational Conference on Robotics & Automstion. 2003: Taipei.
126. Ryu, K., Y. Son, and M. Jung, Modeling and Specification of Dynamic Agents in Fractal Manufacturing Systems. Computers in Industry 2003. **52** (2): p. 161 - 182
127. FIPA, F.f.I.P.A., FIPA Ontology Service Specification. 2001, FIPA: Geneve.
128. Ferreira, P., N. Lohse, and S. Ratchev, Repeatability synthesis methodology for modular ultra-precision assembly systems, in IEEE International Symposium I.A.a. Manufacturing, Editor. 2009. p. 280 - 285
129. Snedecor, G.W. and W.G. Cochran, Statistical Methods. 1989, Iowa, USA: Wiley-Blackwell.
130. Maffei, A., Evolvable production systems: A new business environment, in IEEE International Symposium on Assembly and Manufacturing (ISAM). 2011: Tampere
131. Ferreira, P., N. Lohse, and S. Ratchev, Multi-Agent Architecture for Self-Configuring Modular Assembly Systems, in 9th International IFAC Symposium on Robot Control. 2009: Gifu, Japan.
132. Kennedy, J. and R.C. Eberhart, Swarm Intelligence 1st ed. 2001, London, UK: Morgan Kaufmann.
133. Chang, R., Chemistry 9th ed. 2006: McGraw-Hill.

134. Ferreira, P., N. Lohse, and S. Ratchev, Multi-agent Architecture for Reconfiguration of Precision Modular Assembly Systems, in Precision Assembly Technologies and Systems. 2010, Springer Boston. p. 247-254.
135. Bellifemine, F., et al., Jade Administrator's Guide. Last update: 08-April-2010: Boston, MA.

Publications

Ferreira, P., N. Lohse, and S. Ratchev, Multi-Agent Architecture for Self-Configuring Modular Assembly Systems, in 9th International IFAC Symposium on Robot Control. 2009: Gifu, Japan.

Ferreira, P., N. Lohse, and S. Ratchev, Repeatability synthesis methodology for modular ultra-precision assembly systems, in IEEE International Symposium I.A.a. Manufacturing, Editor. 2009. p. 280 – 285

Ferreira, P., N. Lohse, and S. Ratchev, Multi-agent Architecture for Reconfiguration of Precision Modular Assembly Systems, in Precision Assembly Technologies and Systems. 2010, Springer Boston. p. 247-254.

Appendixes

A XSD Model Source Code

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2010 (http://www.altova.com) by jack (un) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
 <xs:element name="AssemblyProcess">
  <xs:annotation>
   <xs:documentation>Assembly Process XSD definition</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:element name="ControlPorts">
     <xs:complexType>
      <xs:sequence>
       <xs:element ref="ControlPort" minOccurs="4" maxOccurs="unbounded"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="ParameterPorts">
     <xs:complexType>
      <xs:sequence>
       <xs:element ref="ParamenterPort" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element ref="Composed"/>
    <xs:element name="AssemblyProcessType">
     <xs:complexType>
      <xs:simpleContent>
       <xs:extension base="xs:anySimpleType">
        <xs:attribute name="Name" use="required"/>
        <xs:attribute name="AssemblyProcessTypeID" use="required"/>
        <xs:attribute name="Description" use="required"/>
        <xs:attribute name="ProductRelatedClassification" use="required"/>
       </xs:extension>
      </xs:simpleContent>
     </xs:complexType>
    </xs:element>
    <xs:element name="ConfigurationCharacteristics ">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="RuningCost"/>
       <xs:element name="Time"/>
       <xs:element name="Accuracy"/>
       <xs:element name="Repeatability"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="Belongs">
     <xs:complexType>
```

```
      <xs:choice>
       <xs:element name="AssemblySystemID"/>
       <xs:element name="WorkStationID"/>
       <xs:element name="PhysicalEquipmentModuleID"/>
      </xs:choice>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
   <xs:attribute name="Name" use="required"/>
   <xs:attribute name="AssemblyProcessID" use="required"/>
   <xs:attribute name="Description" use="required"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="ControlPort">
  <xs:complexType>
   <xs:attribute name="Name" use="required"/>
   <xs:attribute ref="ControlPortID" use="required"/>
   <xs:attribute ref="InterfaceTypeID" use="required"/>
   <xs:attribute name="Description" use="required"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="ParamenterPort">
  <xs:complexType>
   <xs:attribute name="Name" use="required"/>
   <xs:attribute name="Description" use="required"/>
   <xs:attribute ref="ParameterPortID" use="required"/>
   <xs:attribute ref="InterfaceTypeID" use="required"/>
  </xs:complexType>
 </xs:element>
 <xs:attribute name="ParameterPortID"/>
 <xs:attribute name="ControlPortID"/>
 <xs:attribute name="InterfaceTypeID"/>
 <xs:element name="Composed">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Composition" maxOccurs="unbounded">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="Connections">
        <xs:complexType>
         <xs:sequence>
          <xs:element name="Connection" maxOccurs="unbounded">
           <xs:complexType>
            <xs:choice>
             <xs:element name="ParameterPorts">
              <xs:complexType>
               <xs:sequence>
                <xs:element name="ParameterPortID" minOccurs="2" maxOccurs="unbounded"/>
               </xs:sequence>
              </xs:complexType>
             </xs:element>
             <xs:element name="ControlPorts">
              <xs:complexType>
               <xs:sequence>
                <xs:element name="ControlPortID" minOccurs="2" maxOccurs="unbounded"/>
               </xs:sequence>
              </xs:complexType>
             </xs:element>
            </xs:choice>
           </xs:complexType>
```

```
        </xs:element>
       </xs:sequence>
      </xs:complexType>
     </xs:element>
     <xs:element name="AssemblyProcesses">
      <xs:complexType/>
     </xs:element>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
 </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Interface">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Ports" minOccurs="2" maxOccurs="unbounded">
    <xs:complexType>
     <xs:attribute name="Name" use="required"/>
     <xs:attribute name="PortType" use="required"/>
     <xs:attribute name="Description" use="required"/>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
  <xs:attribute name="Name" use="required"/>
  <xs:attribute name="InterfaceID" use="required"/>
  <xs:attribute name="InterfaceType" use="required"/>
  <xs:attribute name="Description" use="required"/>
 </xs:complexType>
</xs:element>
<xs:element name="PhysicalPort">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="RefrenceFrame" minOccurs="0">
    <xs:complexType>
     <xs:attribute name="X" use="required"/>
     <xs:attribute name="Y" use="required"/>
     <xs:attribute name="Z" use="required"/>
     <xs:attribute name="RX" use="required"/>
     <xs:attribute name="RY" use="required"/>
     <xs:attribute name="RZ" use="required"/>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
  <xs:attribute name="Name" use="required"/>
  <xs:attribute name="PhysicalPortID" use="required"/>
  <xs:attribute name="PortType" use="required"/>
  <xs:attribute name="Description" use="required"/>
 </xs:complexType>
</xs:element>
<xs:element name="EquipmentModule">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="ModuleCapabilities">
    <xs:complexType>
     <xs:sequence>
      <xs:element ref="AssemblyProcess" maxOccurs="unbounded"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
```

```
  <xs:element name="ModuleStructure">
   <xs:complexType>
    <xs:sequence>
     <xs:element ref="PhysicalPort" maxOccurs="unbounded"/>
     <xs:element name="Weight"/>
     <xs:element name="Volume"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="BusinessInformation">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="OwnerName"/>
     <xs:element name="ModuleManufacturerID"/>
     <xs:element name="Cost">
      <xs:complexType>
       <xs:sequence>
        <xs:element ref="Buy"/>
        <xs:element ref="Lease"/>
       </xs:sequence>
      </xs:complexType>
     </xs:element>
     <xs:element name="ModuleAvailability"/>
     <xs:element name="DeliveryTime"/>
     <xs:element name="PreferableCollaborations ">
      <xs:complexType>
       <xs:sequence>
        <xs:element name="PreferableCollaboration" maxOccurs="unbounded">
         <xs:complexType>
          <xs:attribute name="AddedValue" use="required"/>
          <xs:attribute name="ModuleManufacturerID" use="required"/>
         </xs:complexType>
        </xs:element>
       </xs:sequence>
      </xs:complexType>
     </xs:element>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element ref="ConfigurationStrategy"/>
 </xs:sequence>
 <xs:attribute name="Name" use="required"/>
 <xs:attribute name="EquipmentModuleID" use="required"/>
 <xs:attribute name="Description" use="required"/>
</xs:complexType>
</xs:element>
<xs:element name="AssemblySystemRequirements">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="AssemblySystemTargets"/>
   <xs:element ref="PhysicalSystemRequirements"/>
   <xs:element ref="AssemblyProcessRequirements"/>
  </xs:sequence>
  <xs:attribute name="Name" use="required"/>
  <xs:attribute name="RequirementsID" use="required"/>
  <xs:attribute name="Description" use="required"/>
 </xs:complexType>
</xs:element>
<xs:element name="AssemblyProcessRequirements">
 <xs:complexType>
```

```
  <xs:sequence>
   <xs:element ref="AssemblyProcess"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name="PhysicalSystemRequirements">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="AssemblySystem">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="WorkStation" maxOccurs="unbounded">
       <xs:complexType>
        <xs:sequence>
         <xs:element ref="PhysicalEquipmentModule" minOccurs="0" maxOccurs="unbounded"/>
         <xs:element ref="PhysicalConnections" minOccurs="0"/>
         <xs:element name="PhysicalPorts">
          <xs:complexType>
           <xs:sequence>
            <xs:element ref="PhysicalPort" maxOccurs="unbounded"/>
           </xs:sequence>
          </xs:complexType>
         </xs:element>
        </xs:sequence>
        <xs:attribute name="Name" use="required"/>
        <xs:attribute name="WorkStationID" use="required"/>
        <xs:attribute name="Description" use="required"/>
       </xs:complexType>
      </xs:element>
      <xs:element ref="PhysicalConnections"/>
      <xs:element name="SpareEquipmentModules">
       <xs:complexType>
        <xs:sequence>
         <xs:element ref="PhysicalEquipmentModule" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
       </xs:complexType>
      </xs:element>
     </xs:sequence>
     <xs:attribute name="Name" use="required"/>
     <xs:attribute name="AssemblySystemID" use="required"/>
     <xs:attribute name="Description" use="required"/>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name="AssemblySystemTargets">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="FixedTargets">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="Time">
       <xs:complexType>
        <xs:sequence>
         <xs:element name="CycleTime"/>
         <xs:element name="CommissioningTime "/>
        </xs:sequence>
       </xs:complexType>
      </xs:element>
```

```
      <xs:element name="Accuracy"/>
      <xs:element name="Cost">
       <xs:complexType>
        <xs:choice>
         <xs:element ref="Buy"/>
         <xs:element ref="Lease"/>
        </xs:choice>
       </xs:complexType>
      </xs:element>
      <xs:element name="Repeatability"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element ref="ConfigurationStrategy"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name="Buy">
 <xs:complexType>
  <xs:attribute name="Value" use="required"/>
  <xs:attribute name="Currency" use="required"/>
 </xs:complexType>
</xs:element>
<xs:element name="Lease">
 <xs:complexType>
  <xs:attribute name="Currency" use="required"/>
  <xs:attribute name="ValuePerMonth" use="required"/>
  <xs:attribute name="DevalueBuyCostPerMonth" use="required"/>
 </xs:complexType>
</xs:element>
<xs:element name="ConfigurationStrategy">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="PercentageForCost"/>
   <xs:element name="PercentageForTime"/>
   <xs:element name="PercentageForFlexibility"/>
   <xs:element name="PercentageForAccuracy"/>
   <xs:element name="PercentageForRepeatability"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name="PhysicalConnections">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Connection" maxOccurs="unbounded">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="Connects">
       <xs:complexType>
        <xs:sequence>
         <xs:element name="PhysicalPortID" minOccurs="2" maxOccurs="unbounded"/>
        </xs:sequence>
       </xs:complexType>
      </xs:element>
     </xs:sequence>
     <xs:attribute name="Name" use="required"/>
     <xs:attribute name="ConnectionID" use="required"/>
     <xs:attribute name="Description" use="required"/>
    </xs:complexType>
   </xs:element>
```

```
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name="PhysicalEquipmentModule">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="PhysicalPorts">
    <xs:complexType>
     <xs:sequence>
      <xs:element ref="PhysicalPort" maxOccurs="unbounded"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
  <xs:attribute name="Name" use="required"/>
  <xs:attribute name="PhysicalEquipmentModuleID" use="required"/>
  <xs:attribute name="Description" use="required"/>
 </xs:complexType>
</xs:element>
</xs:schema>
```

B Sequence Diagrams of Interaction Protocols

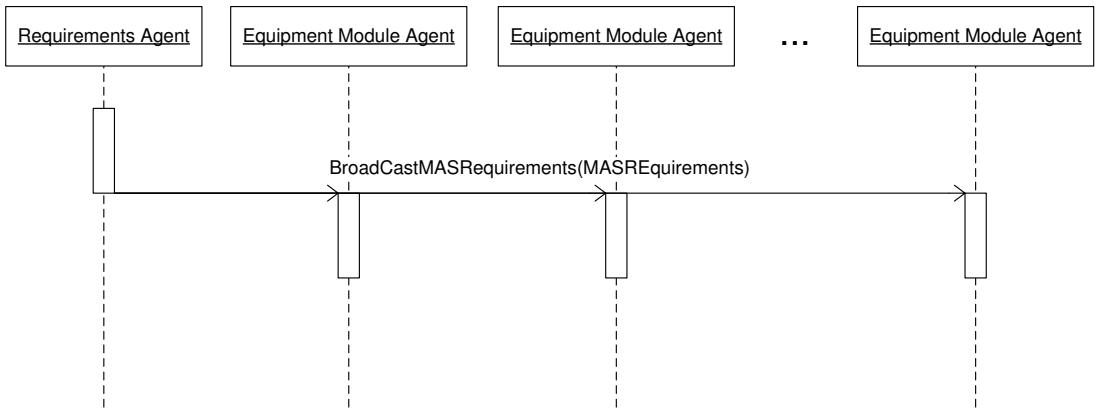


Figure B.1 - Sequence Diagram for the Broadcast of Requirements Protocol

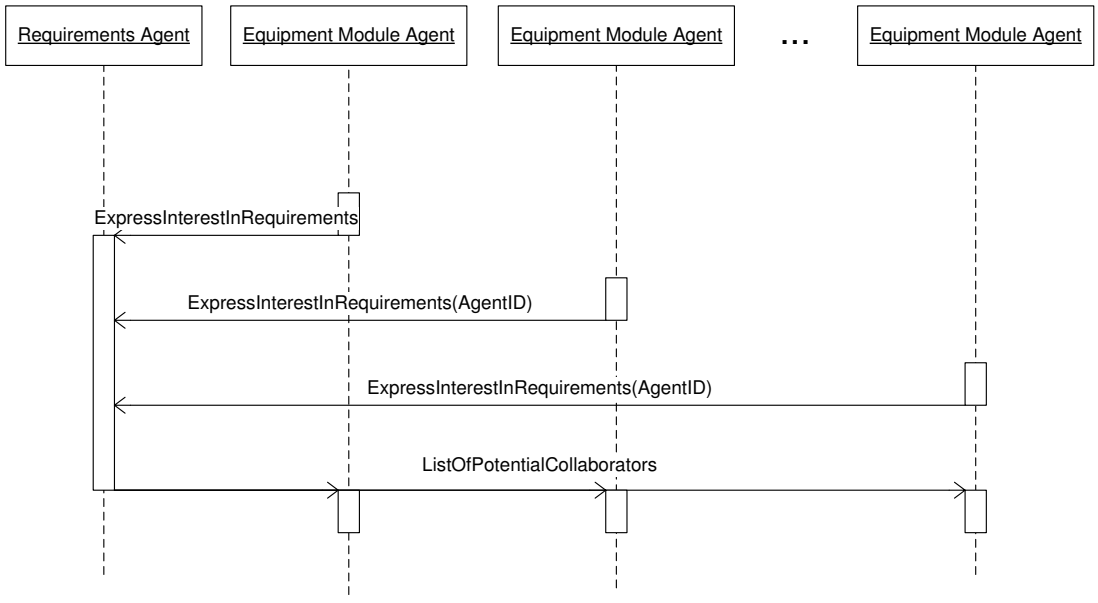


Figure B.2 - Sequence Diagram for the Express Interests in Requirements Protocol

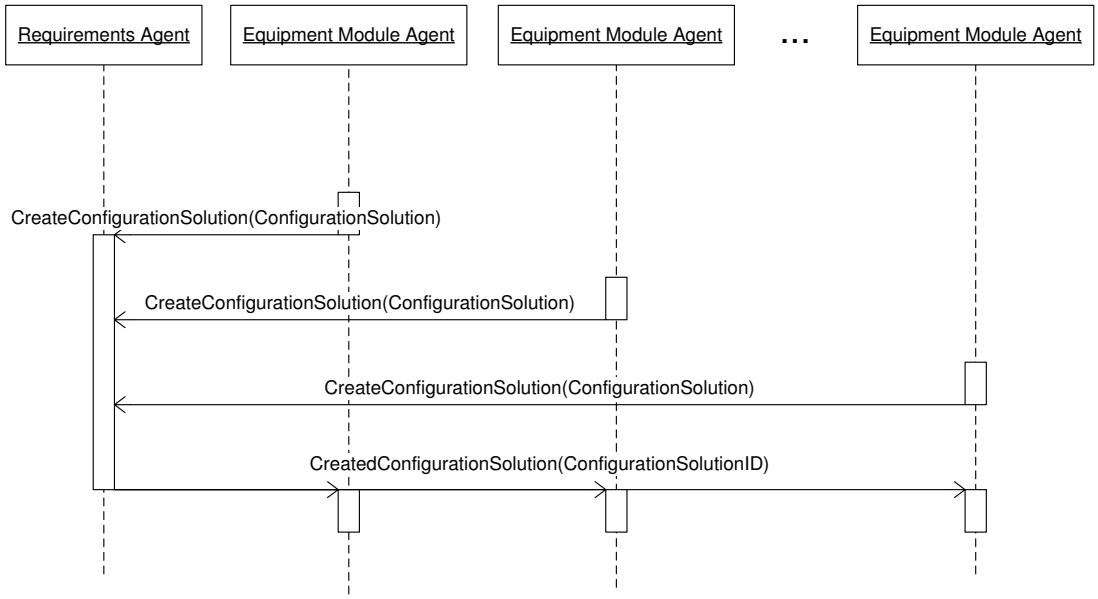


Figure B.3 - Sequence Diagram for the Creation of a Configuration Solution Protocol

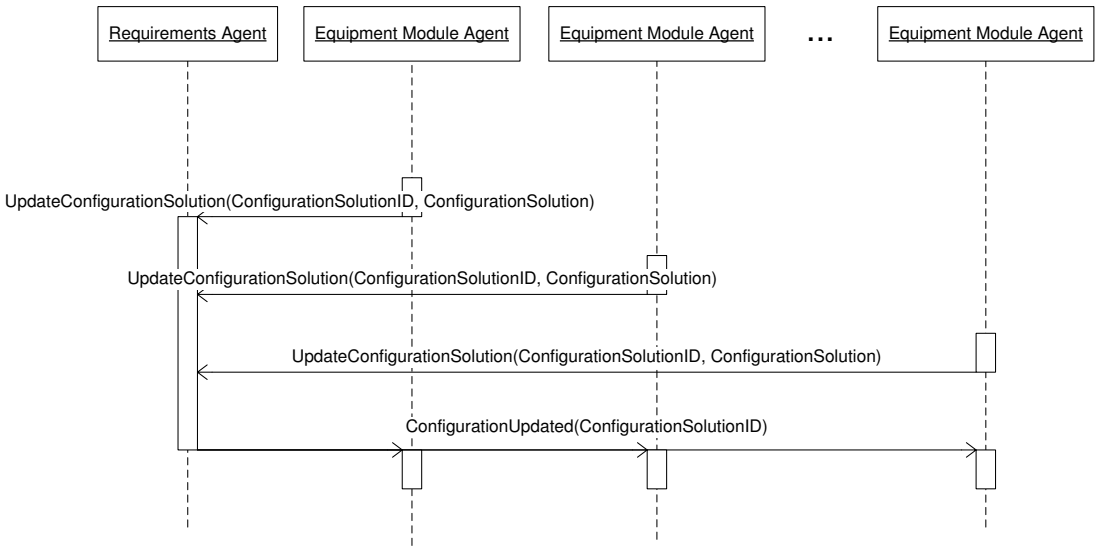


Figure B.4 - Sequence Diagram for the Update of the Configuration Solution Protocol

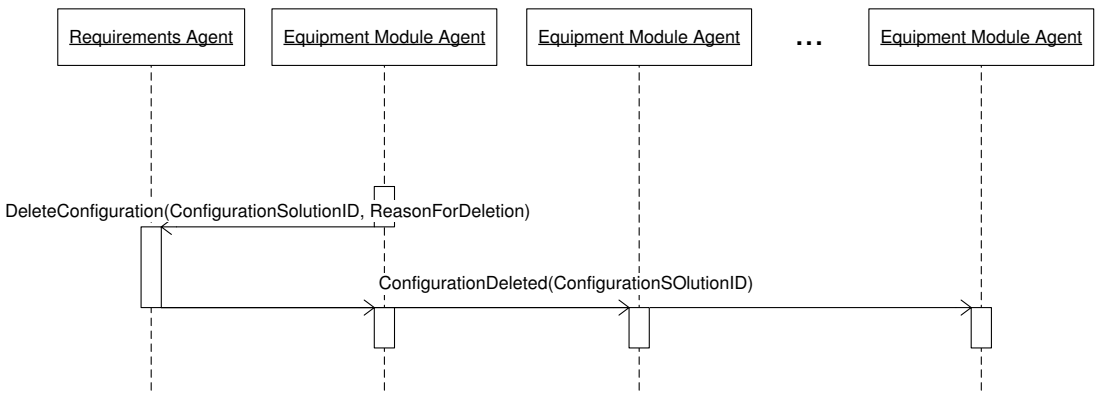


Figure B.5 - Sequence Diagram for the Delete Configuration Protocol

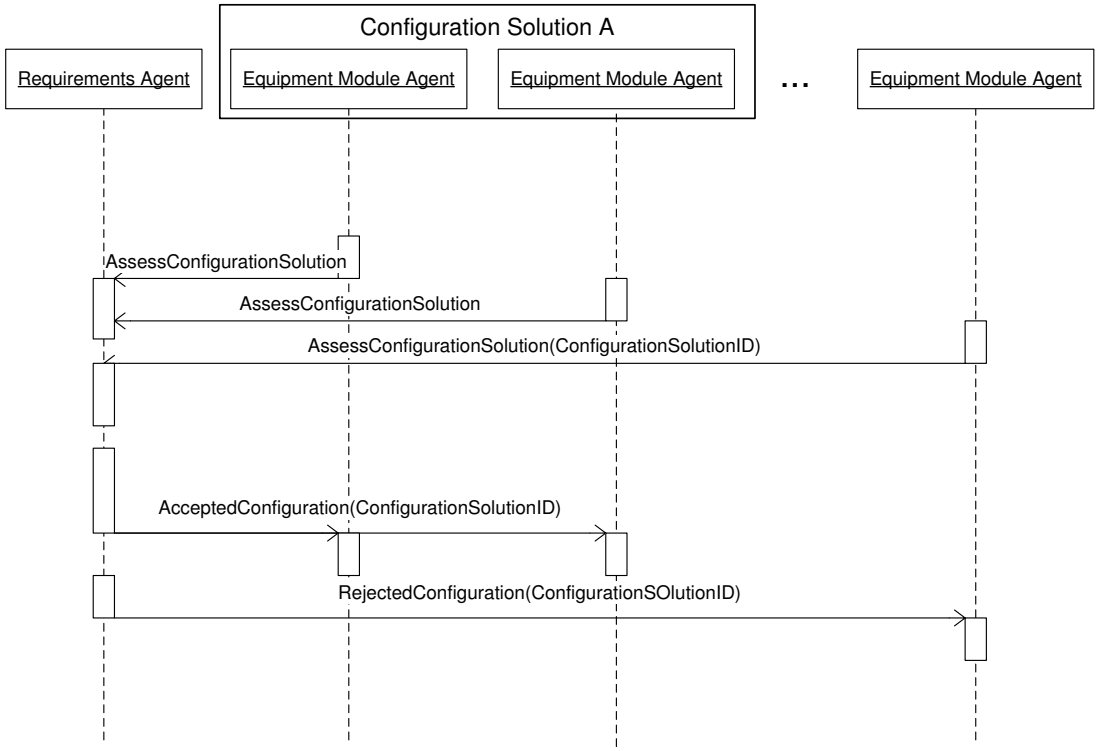


Figure B.6 - Sequence Diagram for the Assessment of Solution Protocol

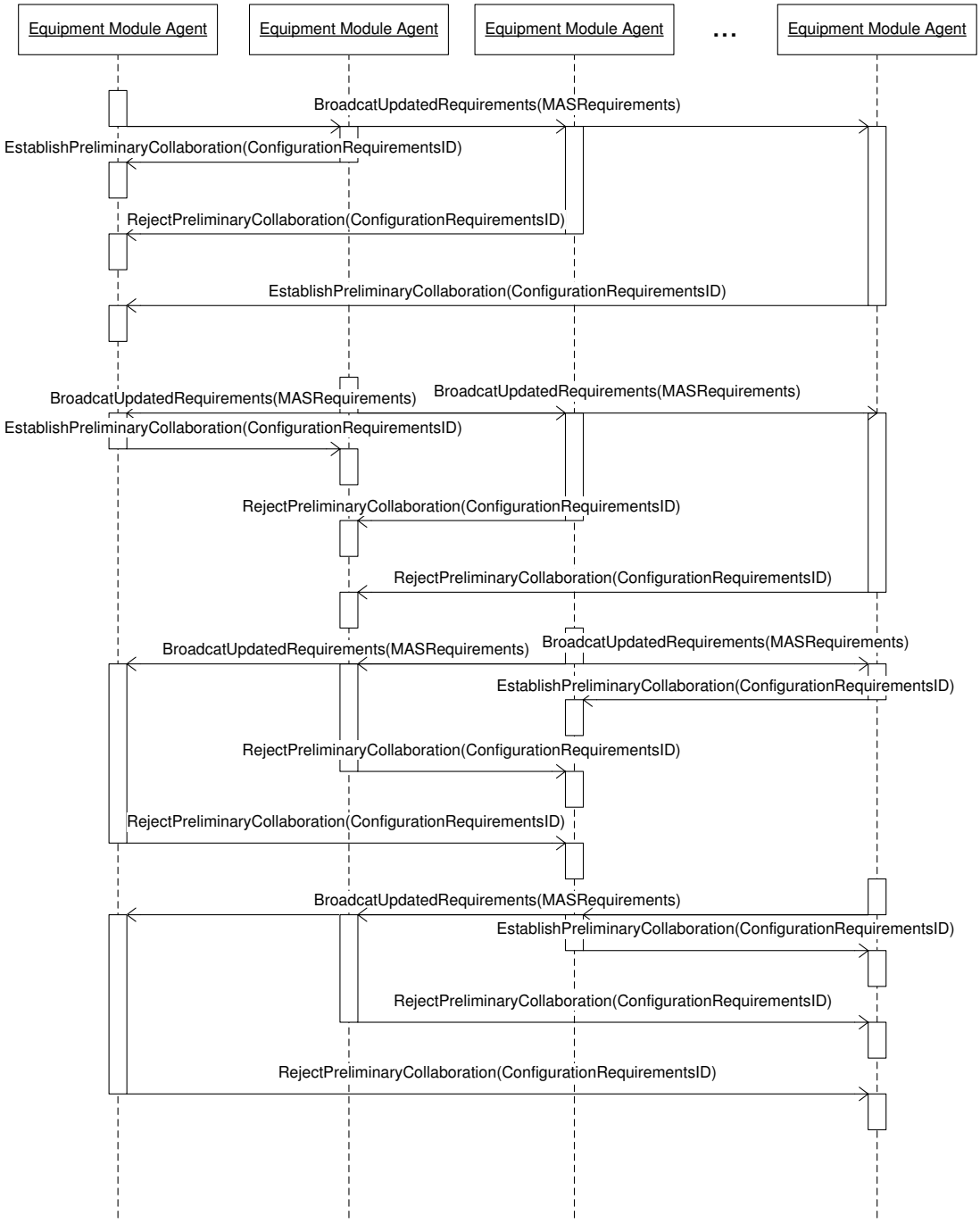


Figure B.7 - Sequence Diagram for the Establish Collaboration Protocol

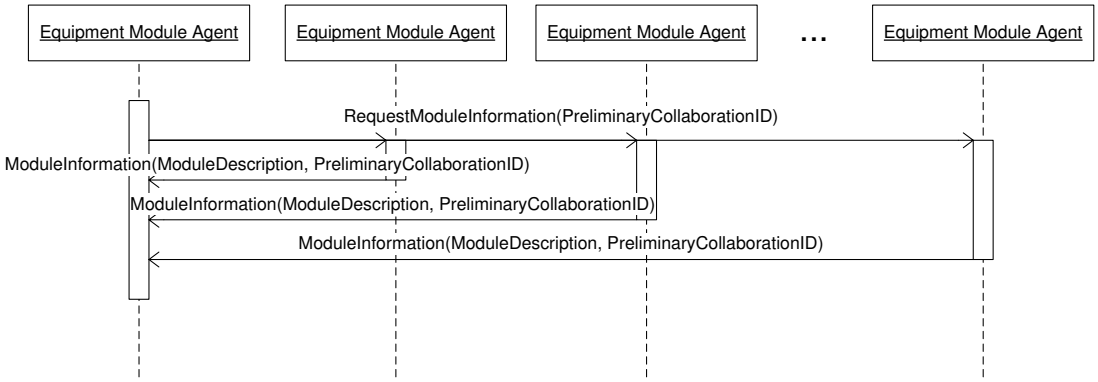


Figure B.8 - Sequence Diagram for the Exchange Module Information Protocol

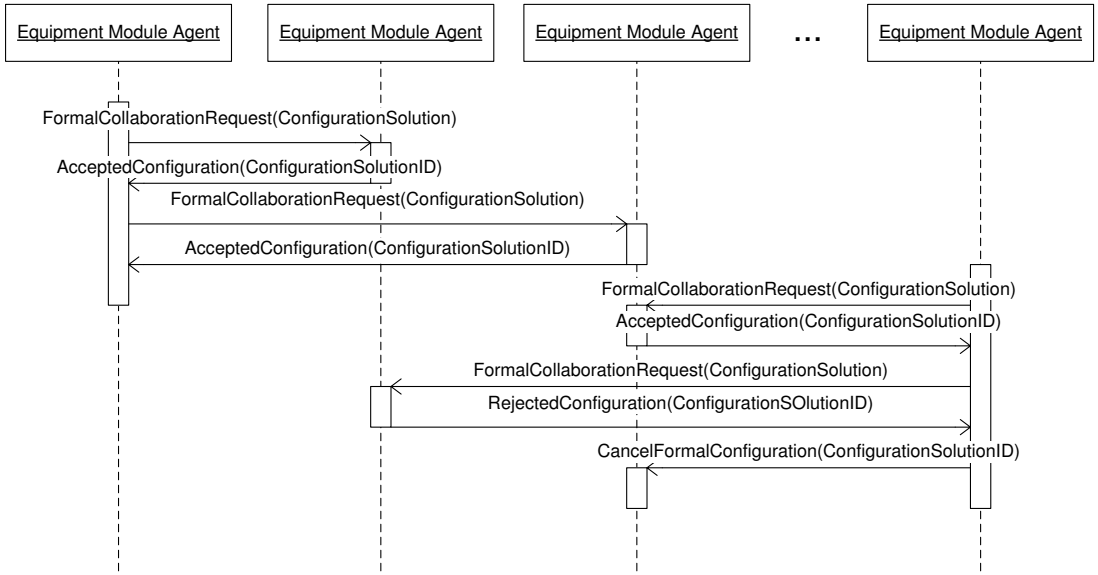


Figure B.9 - Sequence Diagram for the Establish Formal Collaboration Protocol

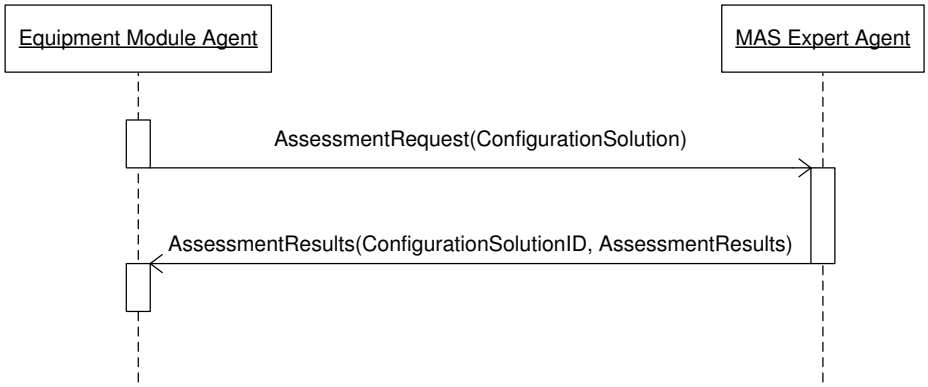


Figure B.10 - Sequence Diagram for the Expert Validation Request Protocol

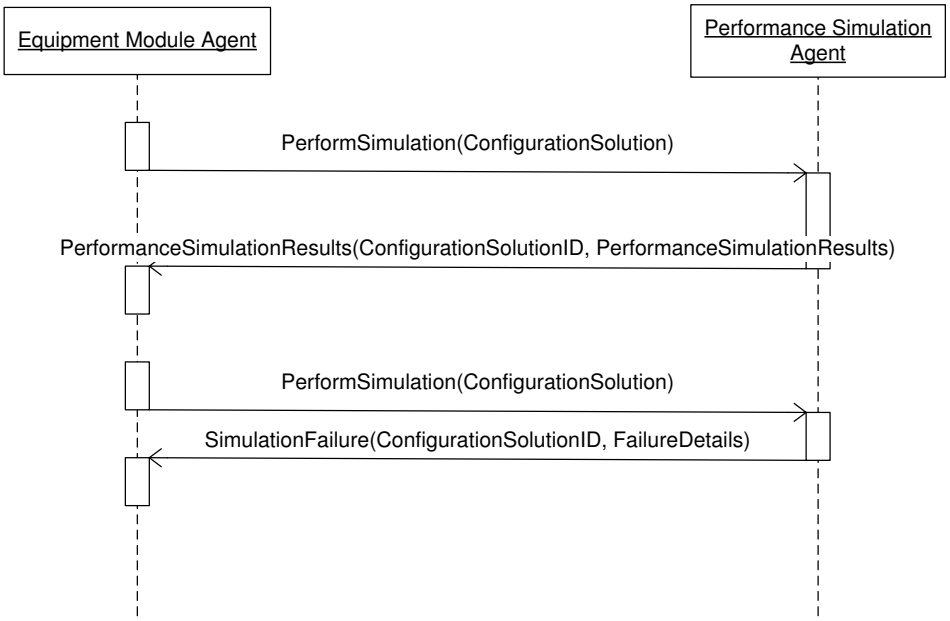


Figure B.11 - Sequence Diagram for the Request for Simulation Protocol

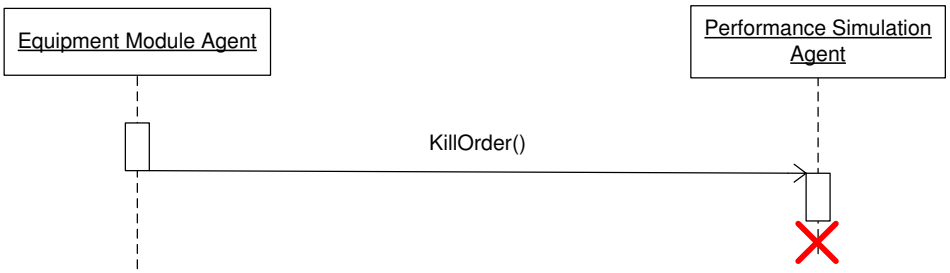


Figure B.12 - Sequence Diagram for the Kill Order Protocol

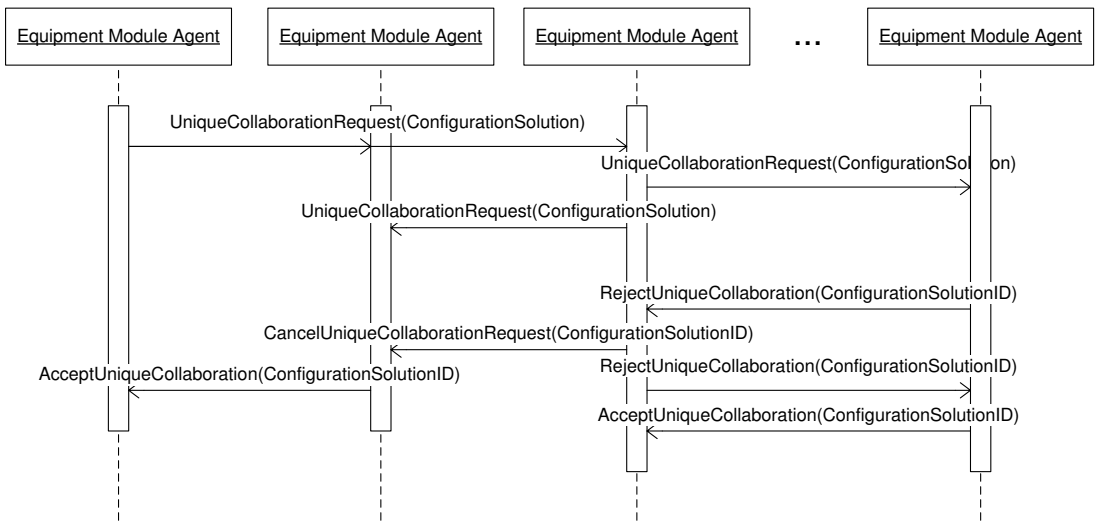


Figure B.13 - Sequence Diagram for the Establish Unique Collaboration Protocol

C Mathematical Normalization Function Deduction and Establishment of Operational Range

Conditions:

$f(Ll) = 1 \xrightarrow[Simplification]{} Ll = 0$

$f(Lu) = 0$

$f[j * Lu] = i\,, where\ 0 < j < 1\ and\ 0 < i < 1$

$f'(x) = 0, for\ Ll < x < Lu$

The number of conditions implies the need for four variables. However this implies a function of third degree which has more than one zero in its derivative form. If we try to contain the two zeros by adding an extra variable the degree of the function would increase and we would have the same problem again. Therefore, the use of a polynomial function of the second degree was used and its operational range will have to be established. The function will be of the following type:

$f[x] = a * x^2 + b * x + c$

The first condition is easily verified:

$f[0] = 1 \Rightarrow a * 0^2 + b * 0 + c = 1 \Longrightarrow c = 1$

Therefore we have:

$$f[x] = a * x^2 + b * x + 1$$

Considering the second condition we have:

$$f[Lu] = 0 \Rightarrow a * (Lu)^2 + b * Lu + 1 = 0 \Rightarrow b = \frac{-1 - a * (Lu)^2}{Lu}$$

Thus, we have:

$$f[x] = a * x^2 + \frac{-1 - a * (Lu)^2}{Lu} * x + 1$$

Considering the third condition, we have:

$$f[j * Lu] = i \Longrightarrow a * (j * Lu)^2 + \frac{-1 - a * (Lu)^2}{Lu} * (j * Lu) + 1 = i \Longrightarrow$$

$$\Longrightarrow a = -\frac{1 - i - j}{(-1 + j)j(Lu)^2}$$

$$\Longrightarrow \mathrm{b} = -\frac{-1 + i + j^2}{(-1 + j)j(Lu)}$$

Thus the function that has the required behaviour is:

$$f(x) = 1 - \frac{(-1 + i + j^2)x}{(-1 + j)j(Lu)} - \frac{(1 - i - j)x^2}{(-1 + j)j(Lu)^2}$$

The next step is to determine the condition on “i” and “j” for the operation of this function that verifies the final condition.

$$f'(x) = 0\,, for\ Ll < x < Lu$$

$$f'(x) = -\frac{-1 + i + j^2}{(-1 + j)jLu} - \frac{2(1 - i - j)x}{(-1 + j)j(Lu)^2} \Longrightarrow$$

$$\Longrightarrow -\frac{-1 + i + j^2}{(-1 + j)jLu} - \frac{2(1 - i - j)x}{(-1 + j)j(Lu)^2} = 0 \Longrightarrow \mathrm{x} = \frac{(-1 + i + j^2)Lu}{2(-1 + i + j)}$$

By establishing the upper and lower limit we have:

$$0 <= \frac{(-1+i+j^2)Lu}{2(-1+i+j)} \leq Lu$$

If we reduce this the conditions are as follows

$$(j < 0 \& ((i \leq 1-j^2 \& Lu \geq 0) || (1-j^2 < i < 1-j \& Lu == 0) || (1-j < i < 1-2j+j^2 \& Lu == 0) || (i \geq 1-2j+j^2 \& Lu \geq 0)))$$

$$|| (j == 0 \& ((i < 1 \& Lu \geq 0) || (i > 1 \& Lu \geq 0)))$$

$$|| (0 < j < 1 \& ((i \leq 1-2j+j^2 \& Lu \geq 0) || (1-2j+j^2 < i < 1-j \& Lu = = 0) || (1-j < i < 1-j^2 \& Lu == 0) || (i \geq 1-j^2 \& Lu \geq 0)))$$

$$|| (j == 1 \& ((i < 0 \& Lu \geq 0) || (i > 0 \& Lu \geq 0)))$$

$$|| (j > 1 \&\& ((i \leq 1-j^2 \&\& Lu \geq 0) || (1-j^2 < i < 1-j \&\& Lu == 0) || (1-j < i < 1-2j+j^2 \&\& Lu == 0) || (i \geq 1-2j+j^2 \&\& Lu \geq 0)))$$

By analysing the conditions, we can eliminate some possibilities based on the requirements. We know that" j" is between zero and one, this means the focus of the analyses is on:

$$(0 < j < 1 \& ((i \leq 1-2j+j^2 \& Lu \geq 0) || (1-2j+j^2 < i < 1-j \& Lu = = 0) || (1-j < i < 1-j^2 \& Lu == 0) || (i \geq 1-j^2 \& Lu \geq 0)))$$

We also know that Lu is larger than zero, thus:

$$(0 < j < 1 \& ((i \leq 1-2j+j^2 \& Lu \geq 0)(i \geq 1-j^2 \& Lu \geq 0)))$$

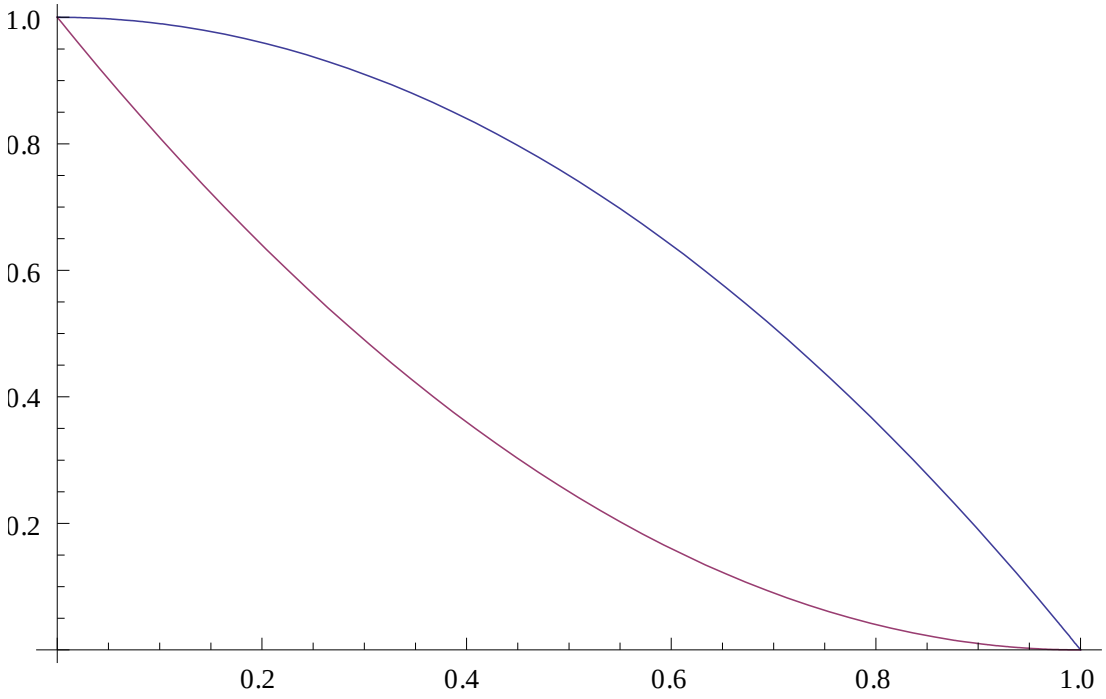
Therefore the range of operation is contained between the two following functions:

$h(j) = 1-2j+j^2$

$g(j) = 1-j^2$

A graphical representation is produce in the following figure:

s



D Requirements Specification Tool

This appendix provides a brief overview of the requirements definition process using the requirements specification tool developed for the EUPASS project (EUPASS [4]).

The role of the process requirements specification largely consists in the definition of the assembly processes required to fulfill the specifications of the defined product. This is done using the skills library which contains the assembly processes and by configuring them in a structure and sequence that realizes a conceptual assembly of the product.

Figure D.14 shows an overview of the assembly processes defined for the Valve test case. This contains several assembly processes with different levels of granularity, namely tasks and operations. Also present are the supply chain processes which are defined in a different color to emphasize their difference.

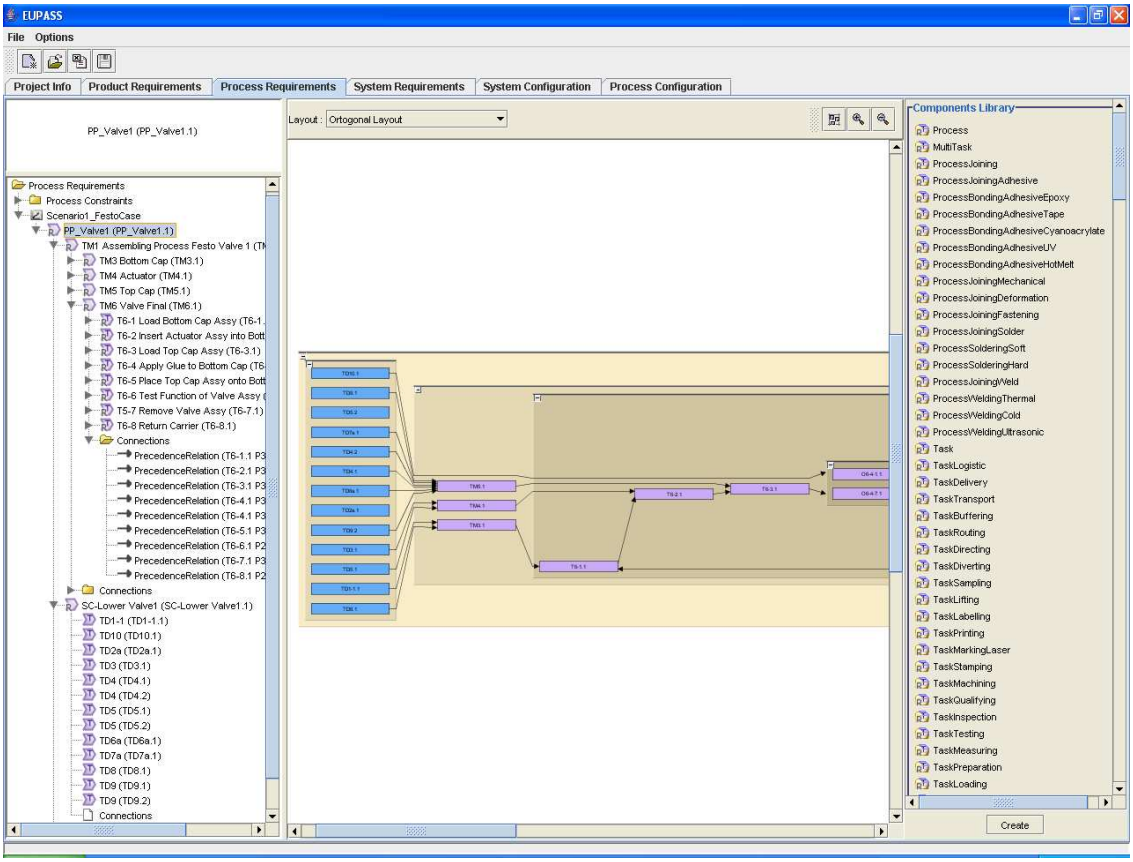


Figure D.14 - Overview of Process Requirements Specification Front End

The process requirements definition starts with the automatic generation of the delivery tasks based on the product definition, more concretely using the components description which includes this information (see **Figure D.15**). The tool also provides the empty tasks for each sub-assembly of the product, thus creating the task structure for the given product. This structure can have several alternative variants which are defined using the support of the sequence generator.

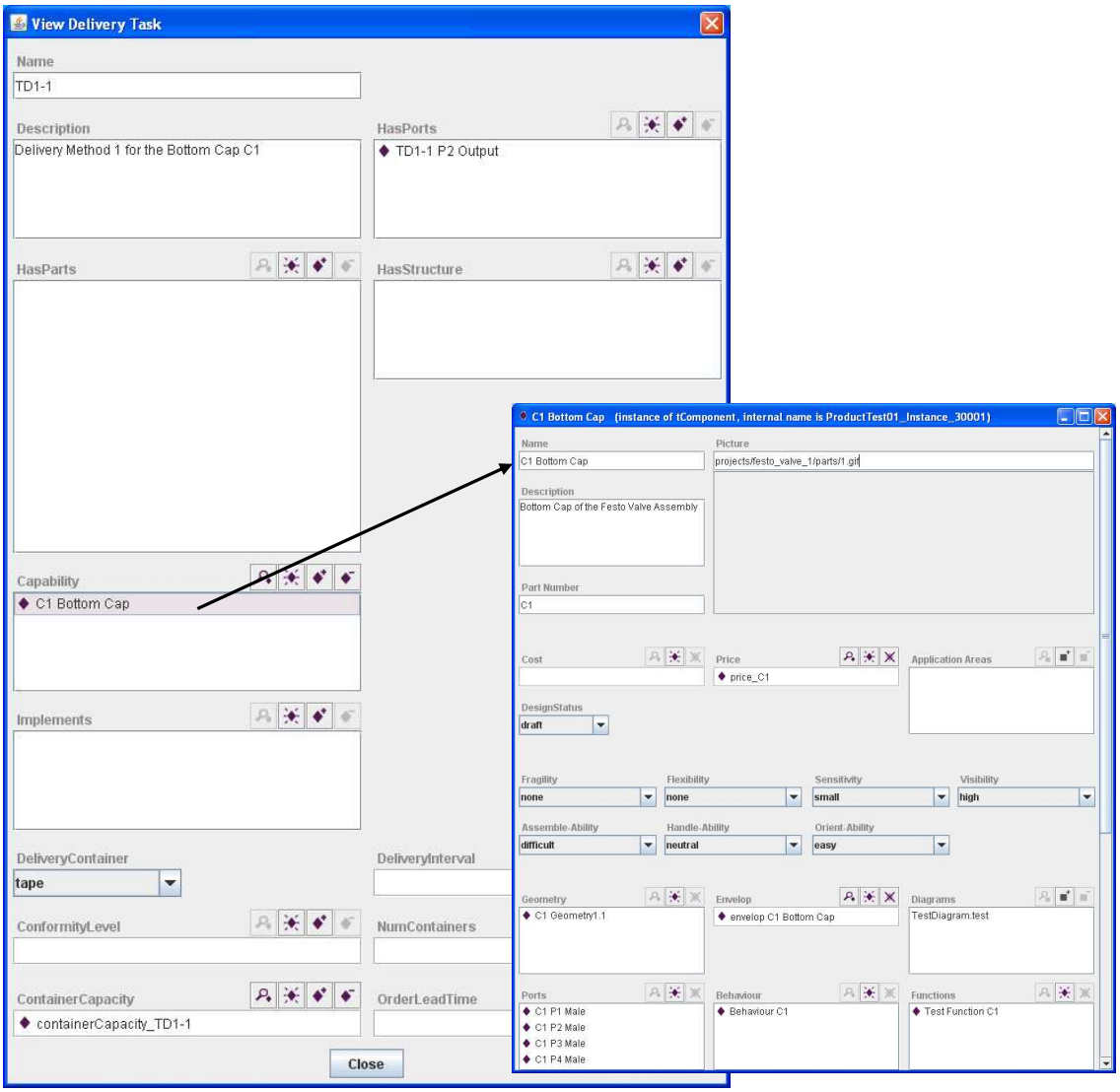


Figure D.15 - Deliver Task Definition

The sequence generator provides a guided creation of the process definitions based on the assembly structure of the product. The user simply chooses which component is first in the sequence and the tool generates the process responsible for it, this component is taken out of the options and the user does repeat this step until the end of the sequence. **Figure D.16** is a screen shot of the sequence generator which includes a visualization of the sub-assembly for a better understanding of the sequence choices.

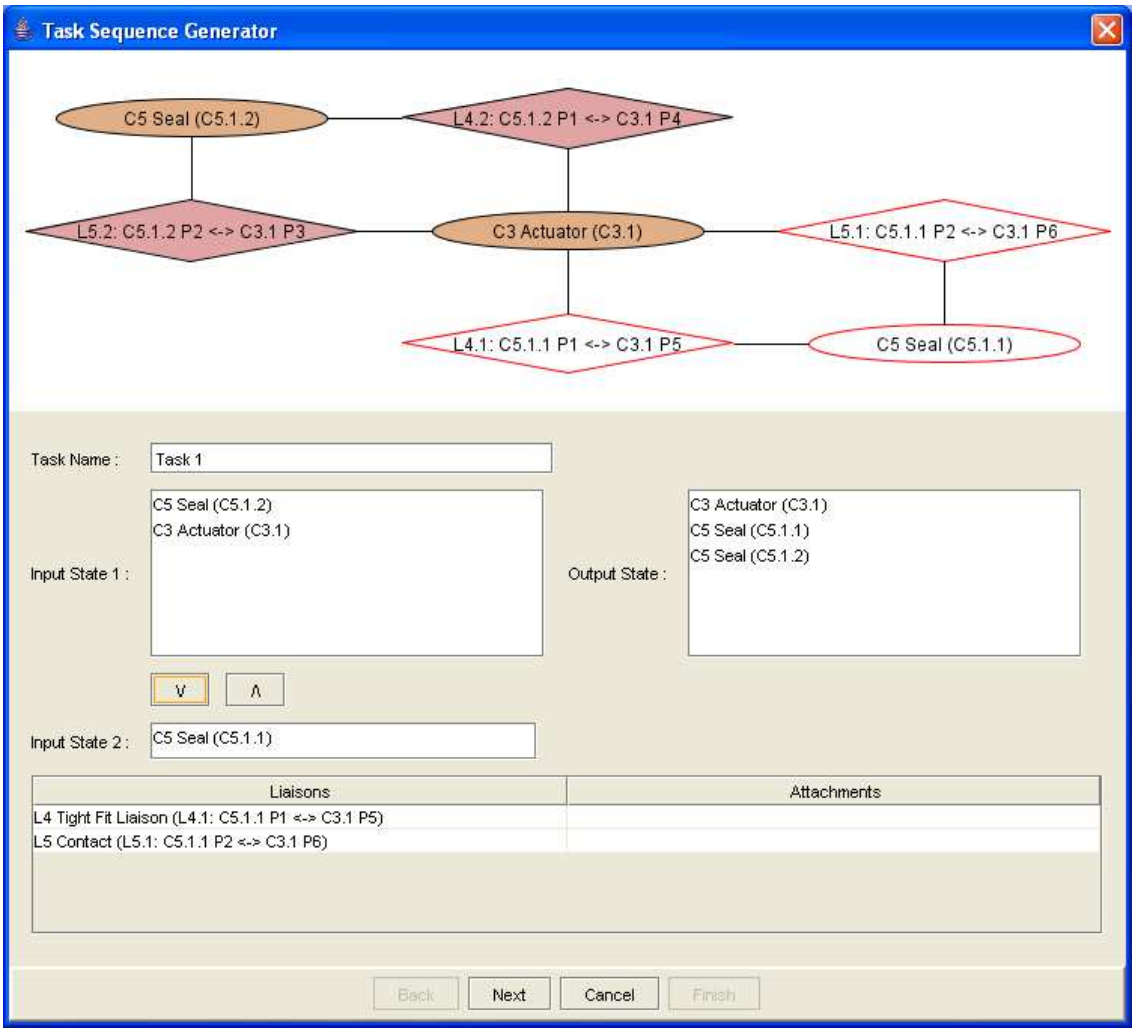


Figure D.16 - Task Sequence Generator

Once this step is concluded the system integrator can edit the created tasks and connections to change the content if required. **Figure D.17** shows an example specification of an assembling process with one possible task sequence. Each alternative sequence is created as a separate variant under the assembling process multi task. This allows the user to assess alternative process sequences and explore them in more detail if required. It is important to note that this definition of alternatives always establishes a preferred variant which is the one used for the exploration of configuration solutions.

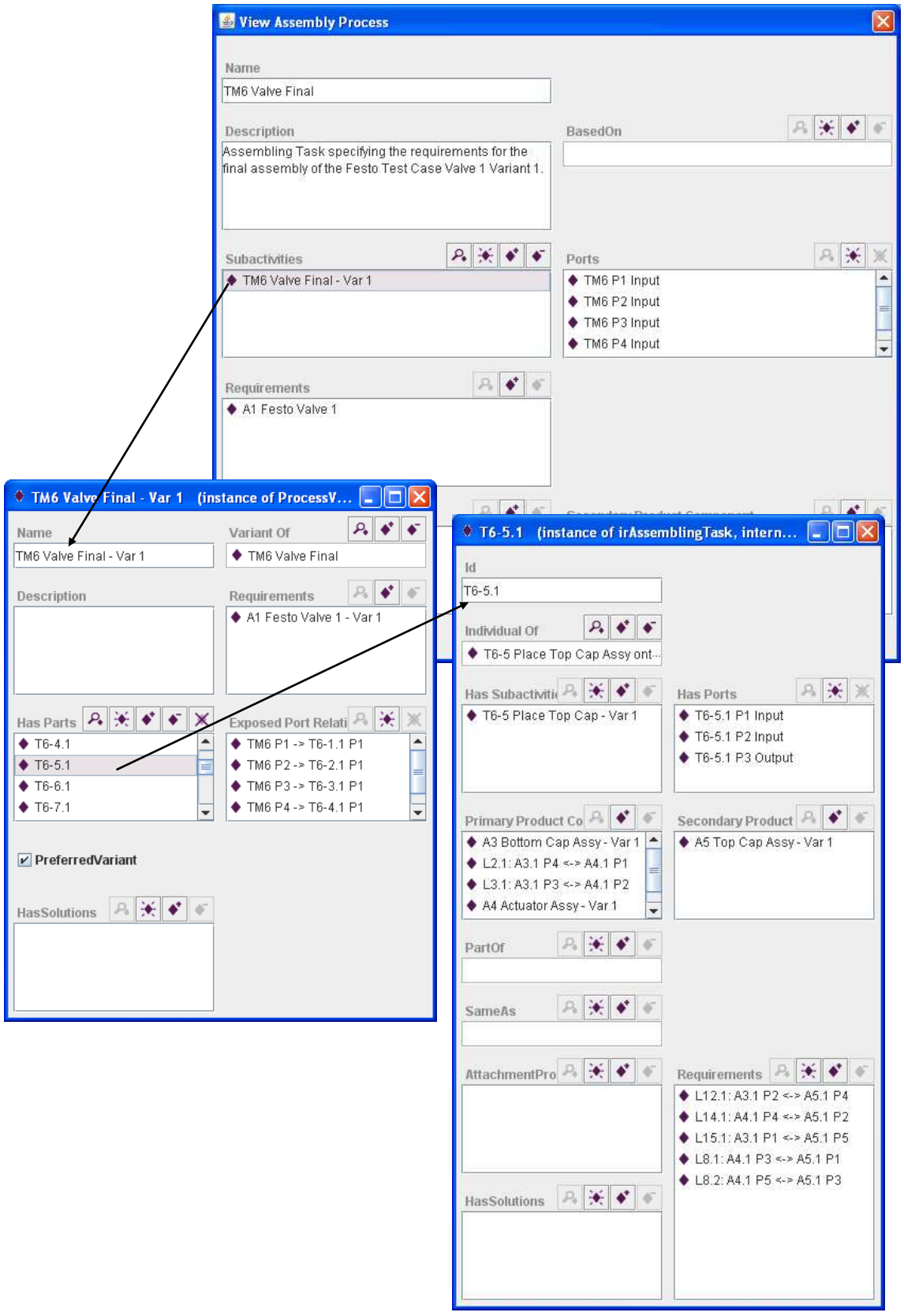


Figure D.17 - Assembling Process and Task Requirements Specification

Each assembling task in the task sequence can be further broken down into required operations and their attributes. Some operations can be derived from the product description and higher level process requirements. Others will only become apparent once further downstream decisions have been taken regarding the needed system configuration. **Figure D.18** gives an example of how tasks are associated to their enabling operations. Each task can be defined through a number of alternative operational requirements much the same as on the assembling process level with lower level tasks. Consequently, the same definition approach can be used to create the link between tasks and operations with variants to express the or-junction in the hierarchy.

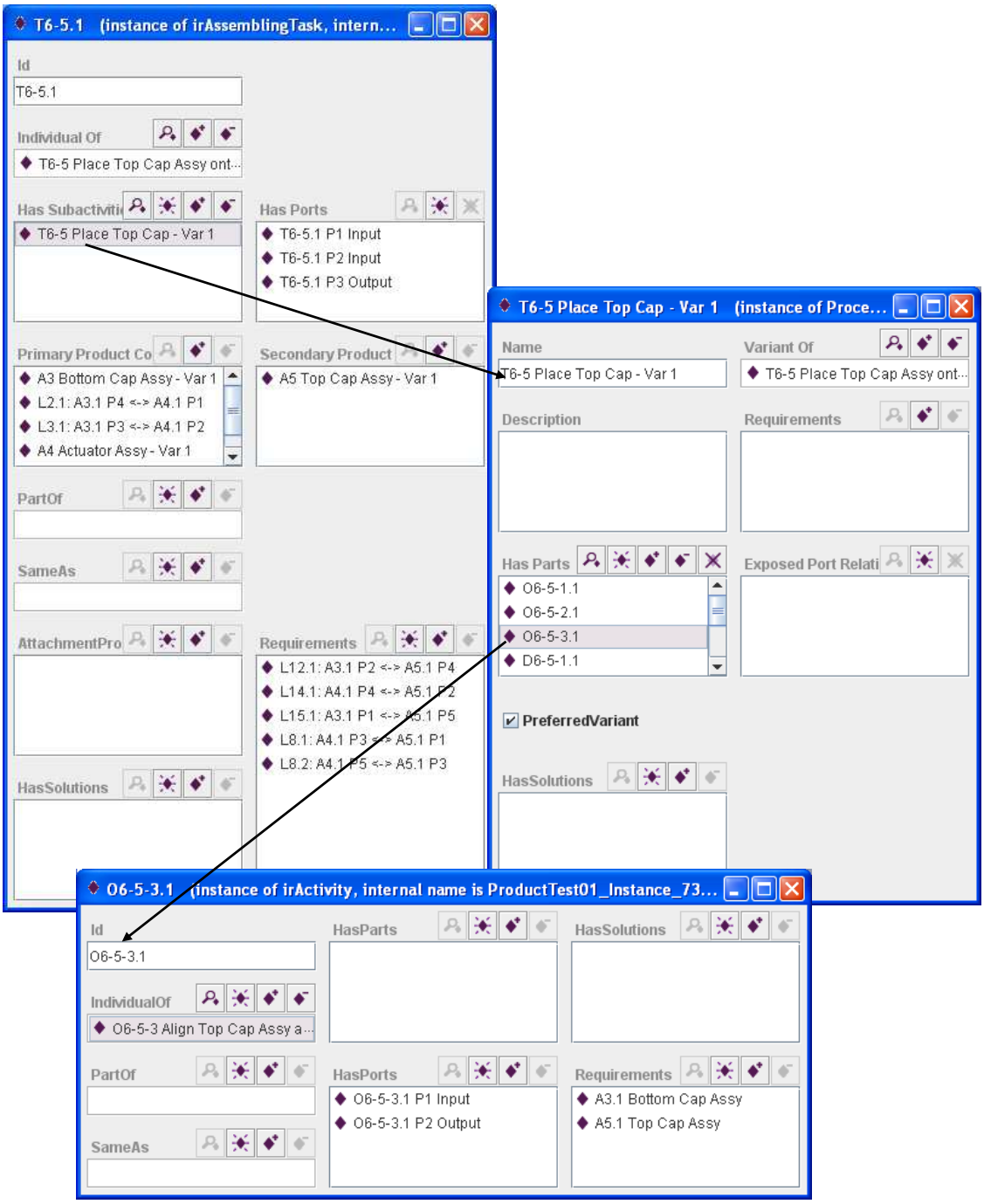


Figure D.18 - Task and Operation Requirements Specification

The system requirements are part of the requirements tool developed for the EUPASS projects. The tool is similar to the assembly process requirements definition and allows for the conceptual assembly system design process which defines an assembly system concept, which in turn fulfils the set of requirements. **Figure D.19** provides a screen shot of this tool where the conceptual system for the valve is shown.

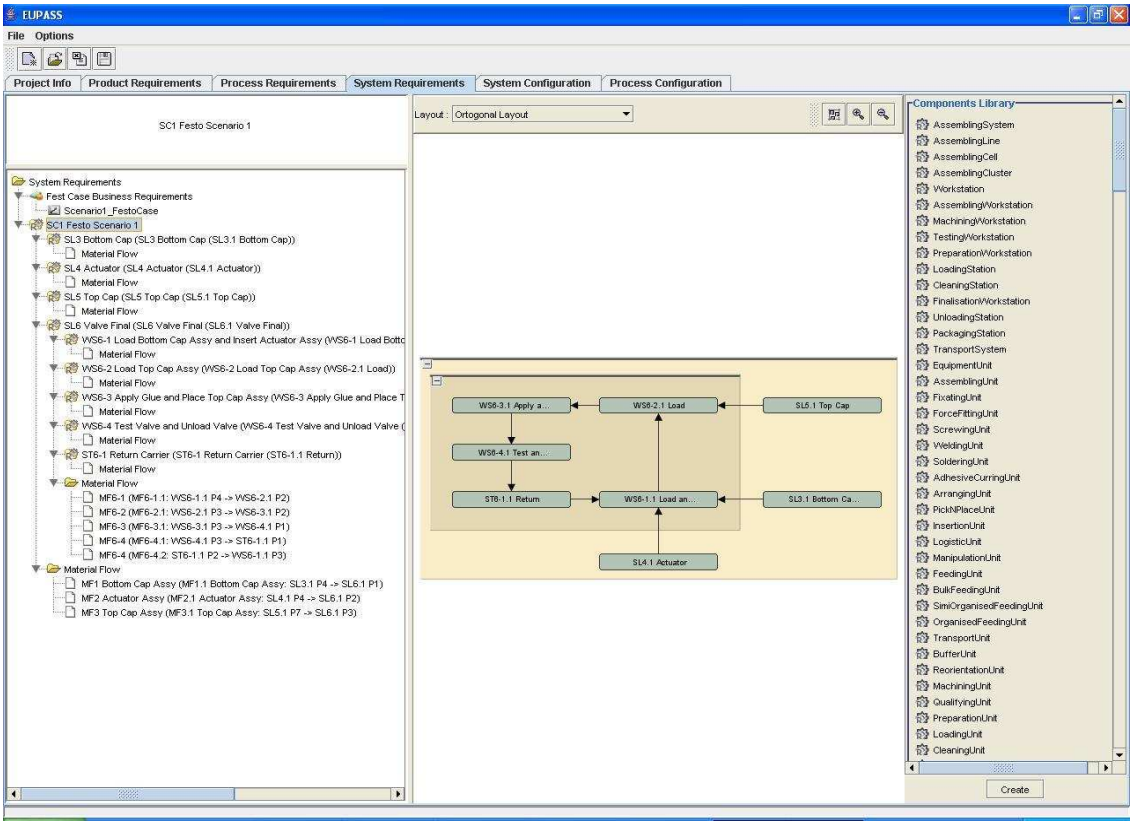


Figure D.19 - Example Assembly System Concept for the Valve Test case

The first step is the definition of the system concept, which is followed by the definition of the conceptual workstations and the conceptual equipment modules (if required). The tool supports all this as well as the definition of the material flow in the system which is closely related to the assembly process requirements. The definition of the system requirements process also provides the link between the defined conceptual system and the assembly process requirements.

E Manual Configuration Tool

This appendix provides a brief overview of the manual configuration process using the manual configuration tool developed for the EUPASS project (EUPASS [4]).

The role of the system configuration tool is to define possible assembly system configurations that realise the process requirements associated to it by the system concept. The tool only supports the configuration of compatible equipment modules descriptions and assumes that they portray their capabilities correctly. The focus in this section is to give an example of how this method has been implemented in the virtual assembly system configuration tool.

The configuration essentially consists of two parts; the selection and configuration of available equipment modules into possible physical system solutions and the configuration of the control logic. The physical configuration of the system focuses on the selection of appropriate equipment modules, based on their skill capabilities and interconnection constraints, and connecting them together. The process logic focuses on defining the sequential order between the skills of the equipment modules selected for a system configuration.

Figure E.20 shows an overview of the main interface used by the virtual assembly configuration tool. The illustrated example shows a possible workstation configuration for the placing and gluing of the Top Cap of a Valve onto the main assembly. The interface shows the hierarchical structure of the configuration and the physical interrelationships between the modules. All equipment modules are integrated by reference only into the underlying assembly system configuration model. The main objective of the tool is to find the most suitable modules, connect them to each other, and trigger the evaluation of the resulting system configuration.

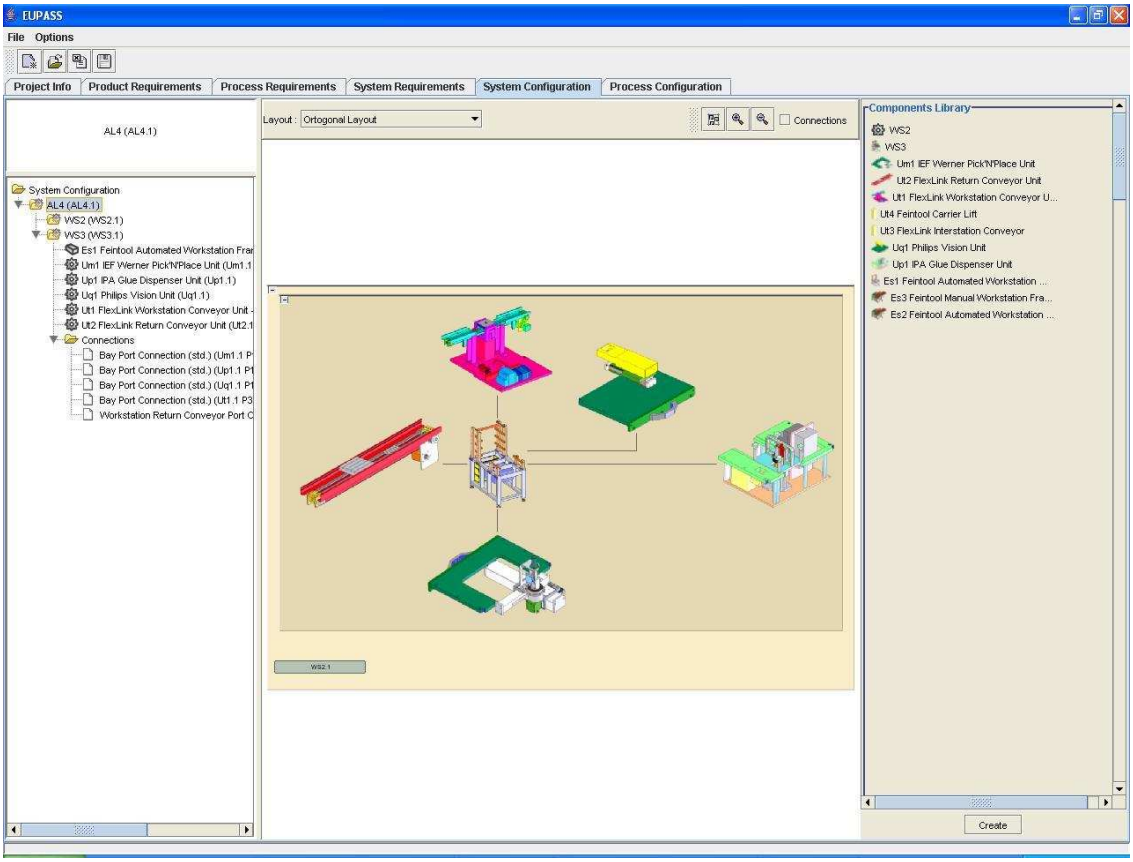


Figure E.20 - Example Workstation Configuration Overview

The configuration of assembly system solutions is a bottom up approach. The tool does however create an empty system structure based on the associated assembly system concept to maintain the consistency of the models. This structure is strictly speaking generated in a top down fashion but remains empty until it is being populated with detail from the lowest level (bottom up).

The interface contains in the "Components Library" the list of available modules to fit in the selected level, meaning that if the selected level is the workstation level then the shown modules will be the ones that can be used to build the workstation. This module list is derived from the set of available XML Files and contains all the relevant information extracted from them allowing for the configuration of the modules. All suitable modules for a given set of requirements will be listed and made available to the user to select from.

The module selection is executed by dragging the selected module from the "Components Library" into the specific place holder, namely a workstation if working on that level. **Figure E.21** shows an example of how the reference or

Individual of the EUPASS base frame module is added into a workstation configuration. The information added to the configuration model only establishes a link to the generic description of the module and replicates individual IDs for its interface ports (connection point). This is required to allow references to the same module to be used within the same configuration without losing the ability to unambiguously connect it. This information is created automatically by the tool maintaining the system validity and simplifying the configurations procedure.

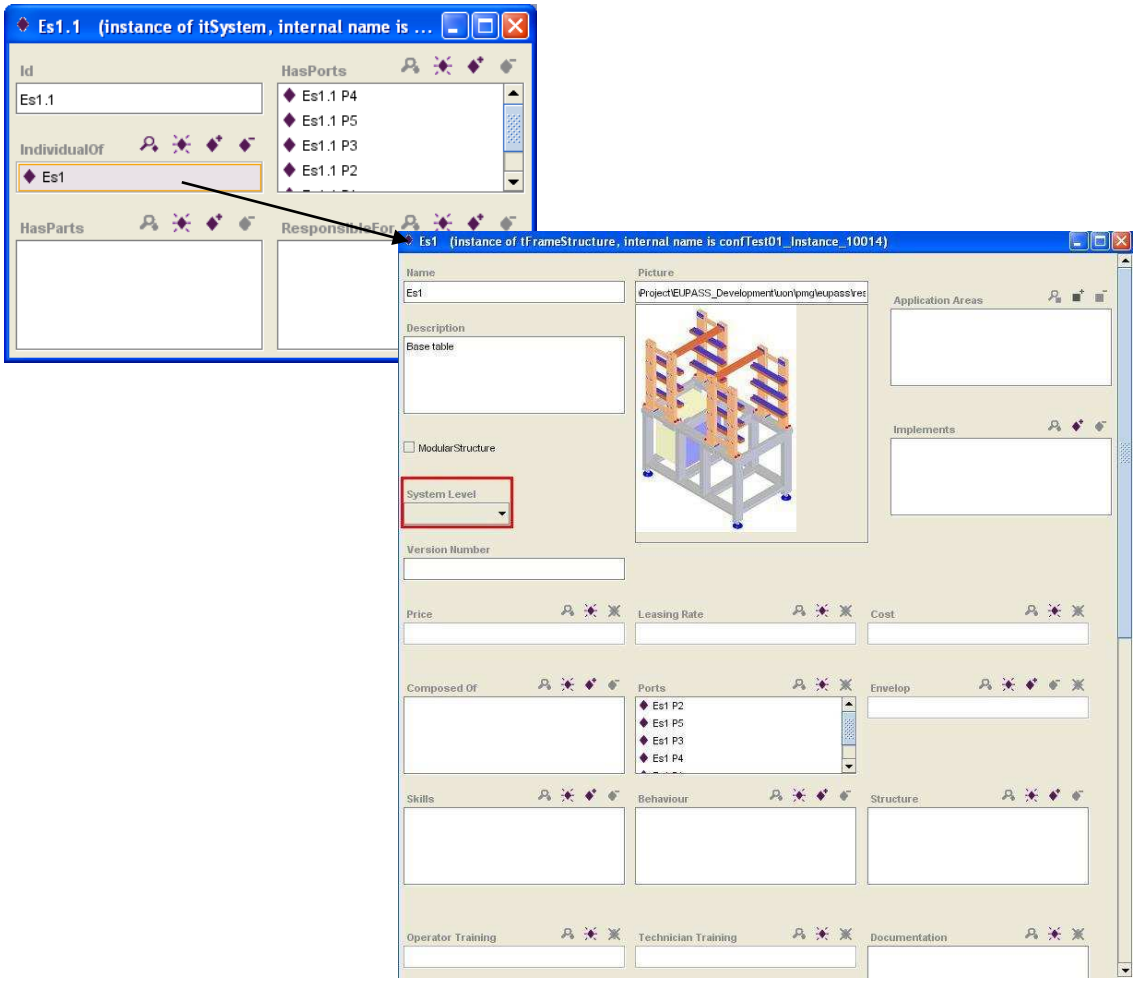


Figure E.21 - Example Individual of an Equipment Module

This is a workstation configuration in this specific case. Each higher level configuration definition has two parts in the same way as the system concepts during the conceptual design. The information directly included in the configuration model is only a reference to the generic description of the workstation. This allows the same workstation configuration to be used as part of other configuration without having to replicate it. Additionally this allows for an easy assessment of the equipment

similarity within a system solution and between different proposals. **Figure E.22** shows the relationships between the different workstation related definitions.

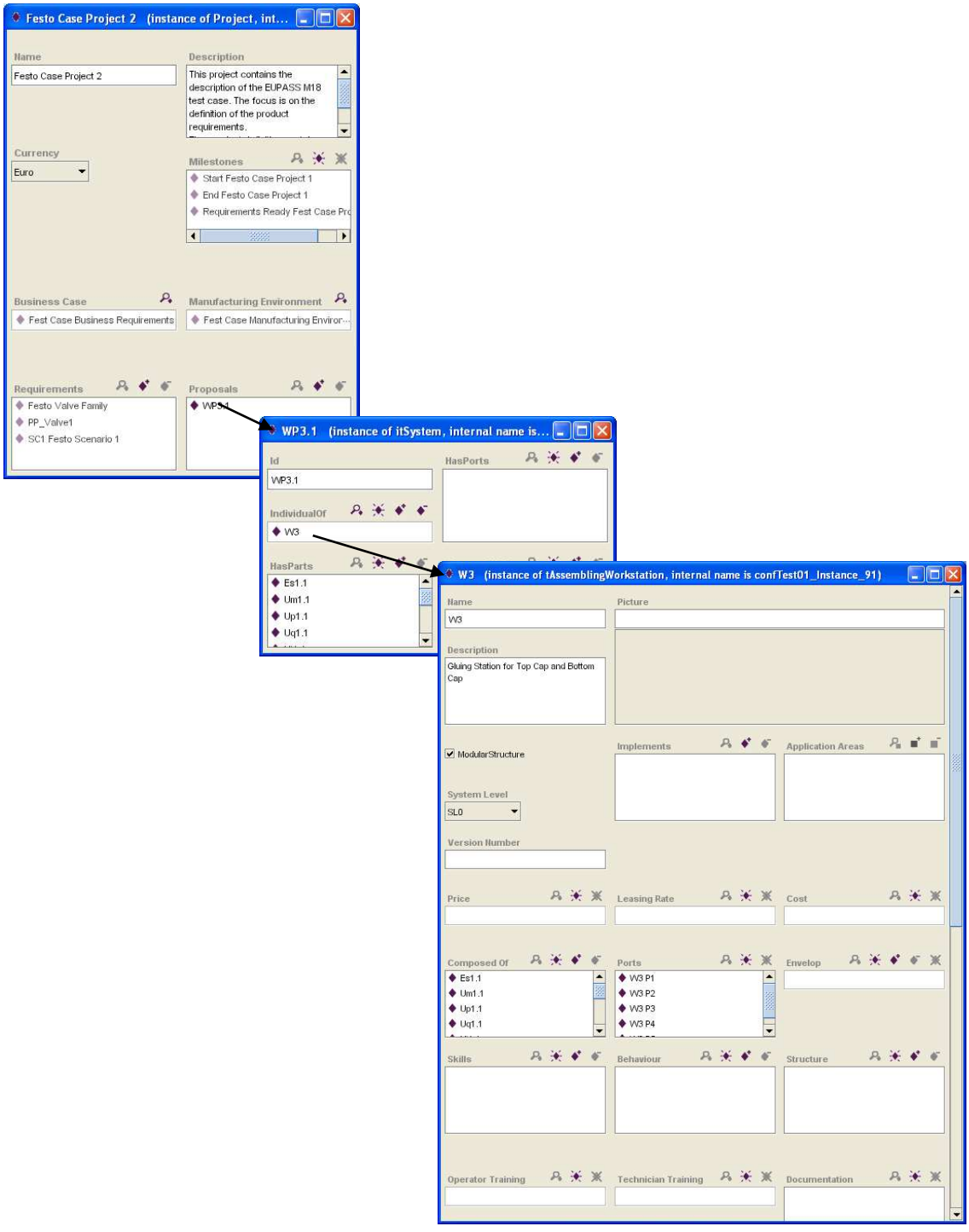


Figure E.22 Example of a Workstation Configuration

Once all the modules have been chosen and added to a higher level configuration, they need to be connected to each other. The creation of a connection is a simple process for the user as the tool maintains all the required constraints towards

achieving a valid system. To establish a connection one needs to tick the "Connections" check box and proceed to select which modules to plug together. **Figure E.23** shows the window that guides the user after selecting both modules. The tool only allows for the connection of same interface types, as well as maintaining the right socket types (male or female).

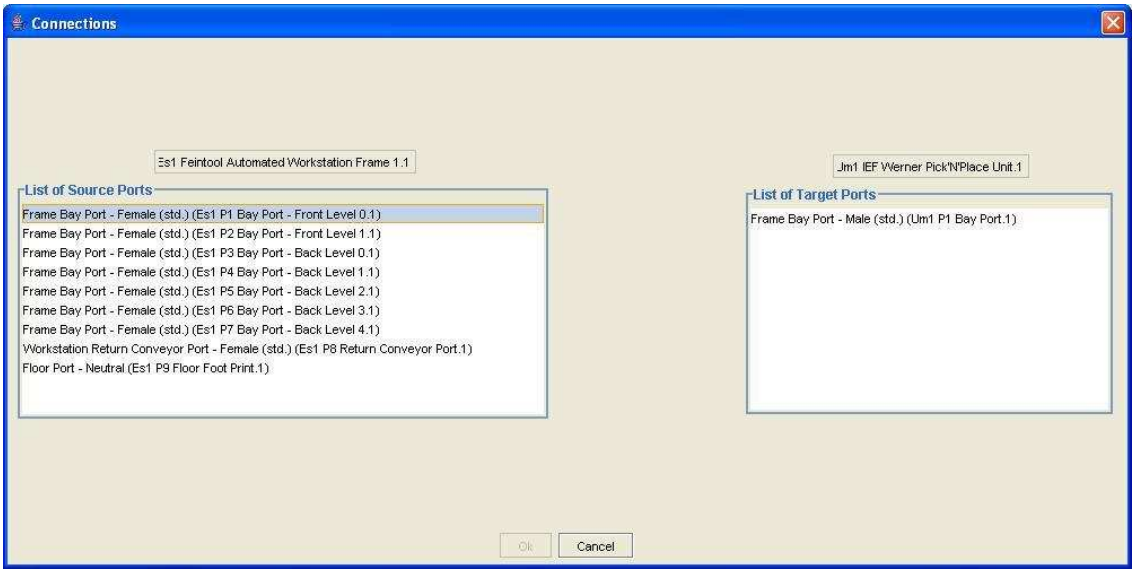


Figure E.23 – Example of the Connection creation

Once the ports for the connection have been selected the tool creates the connection. **Figure E.24** shows how the connection between two modules is defined. The connection is added to the definition of the higher level configuration, in this case the workstation.

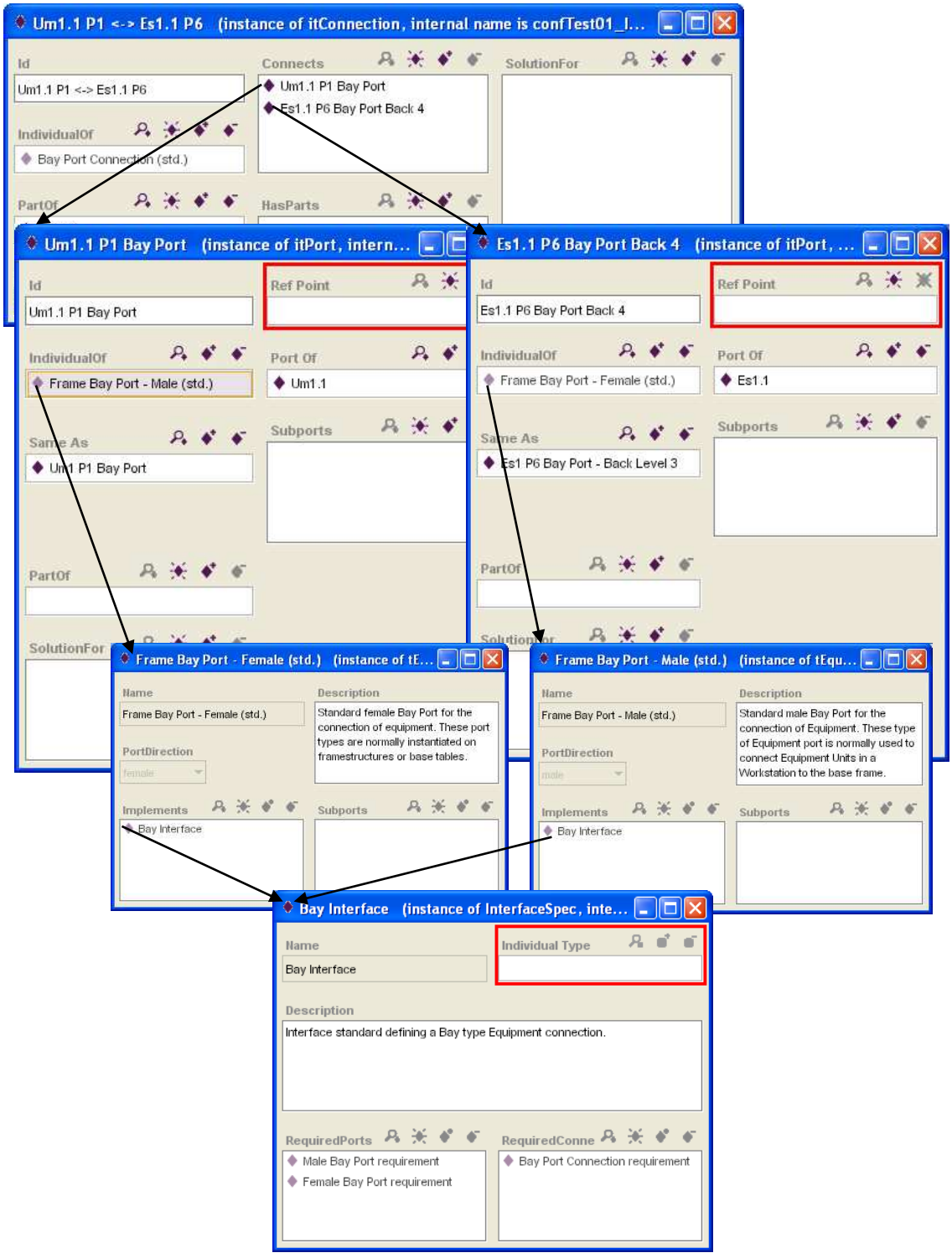


Figure E.24 Example Connection between two Modules

The given illustration is not an exhaustive specification of a whole assembly system. The main purpose is to illustrate the principle means of defining a system.

The role of the process configuration tool is to define the possible process sequences for a specific system configuration. The tool allows the configuration of skills

contained in the system which are provided by the equipment module description files. This section will provide an example of how this method has been implemented in the virtual assembly system configuration tool. The instantiation of the actual assembling process configuration is based on the assembling process requirements defined by the requirements definition tool. The assembling process requirements give a framework for the detailed definition of the actual skill sequence required for the control of a workstation.

Figure E.25 shows an overview of the main interface used for the process configuration tool. The illustrated example shows a process configuration for the proposed workstation configuration for the placing and gluing of the Top Cap of a Valve onto the main assembly shown in the previous section. The interface shows the sequential structure of the configuration as well as the control interfaces between the processes. The main objective of the tool is to find the best possible process configuration for a given system and convey it to the line configurator.

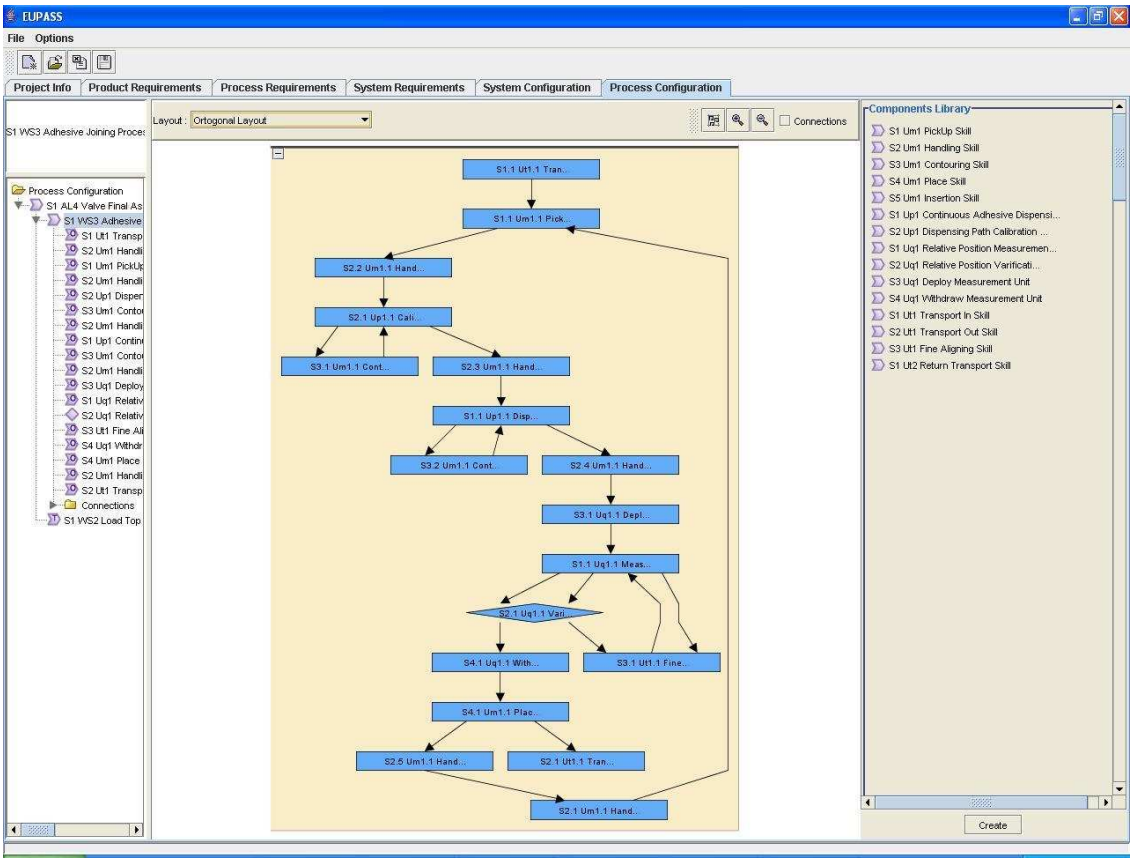


Figure E.25 - Example of Process Configuration Overview

The Process Configuration tool is intrinsically related with the System Configuration since the base structure for process configuration is generated from the system configuration structure in a top down approach and it remains empty until it is being populated with detail from the lowest level (bottom up). **Figure E.26** shows an example of a higher level relation between the system configuration and the process configuration.

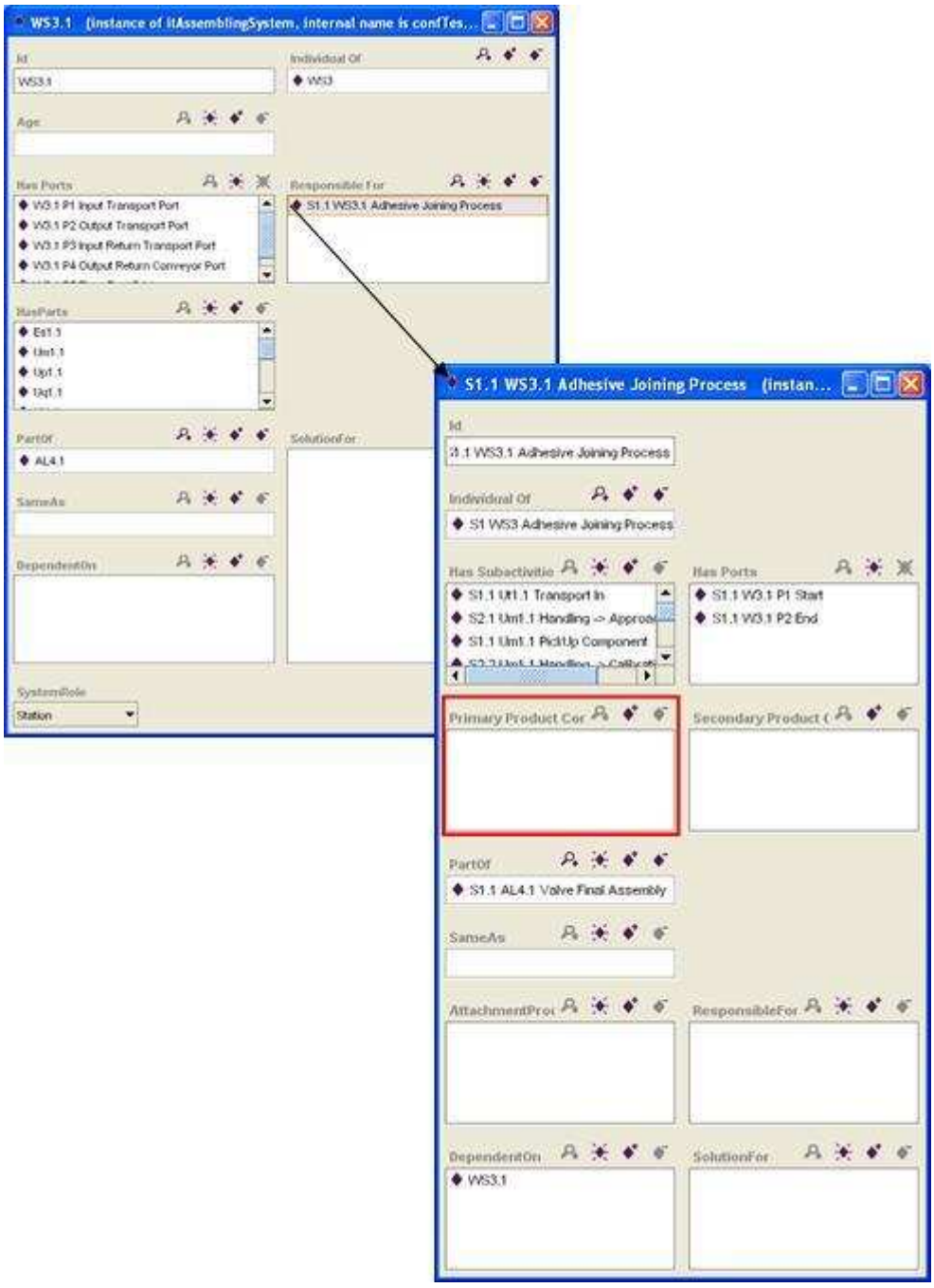


Figure E.26 - Example of Relation between System Configuration and Process Configuration

The main interface also contains in the "Components Library" the list of assembly processes (Skills) available in the selected level which is directly related with the system configuration. The process configuration starts with the selection of the processes (Skills) needed to satisfy the requirements. This is executed by dragging the selected process into the intended place holder. **Figure E.27** shows the instantiated assembly process (Skill) and its relation with the individual assembly process (Skill) that originates in the equipment module description.

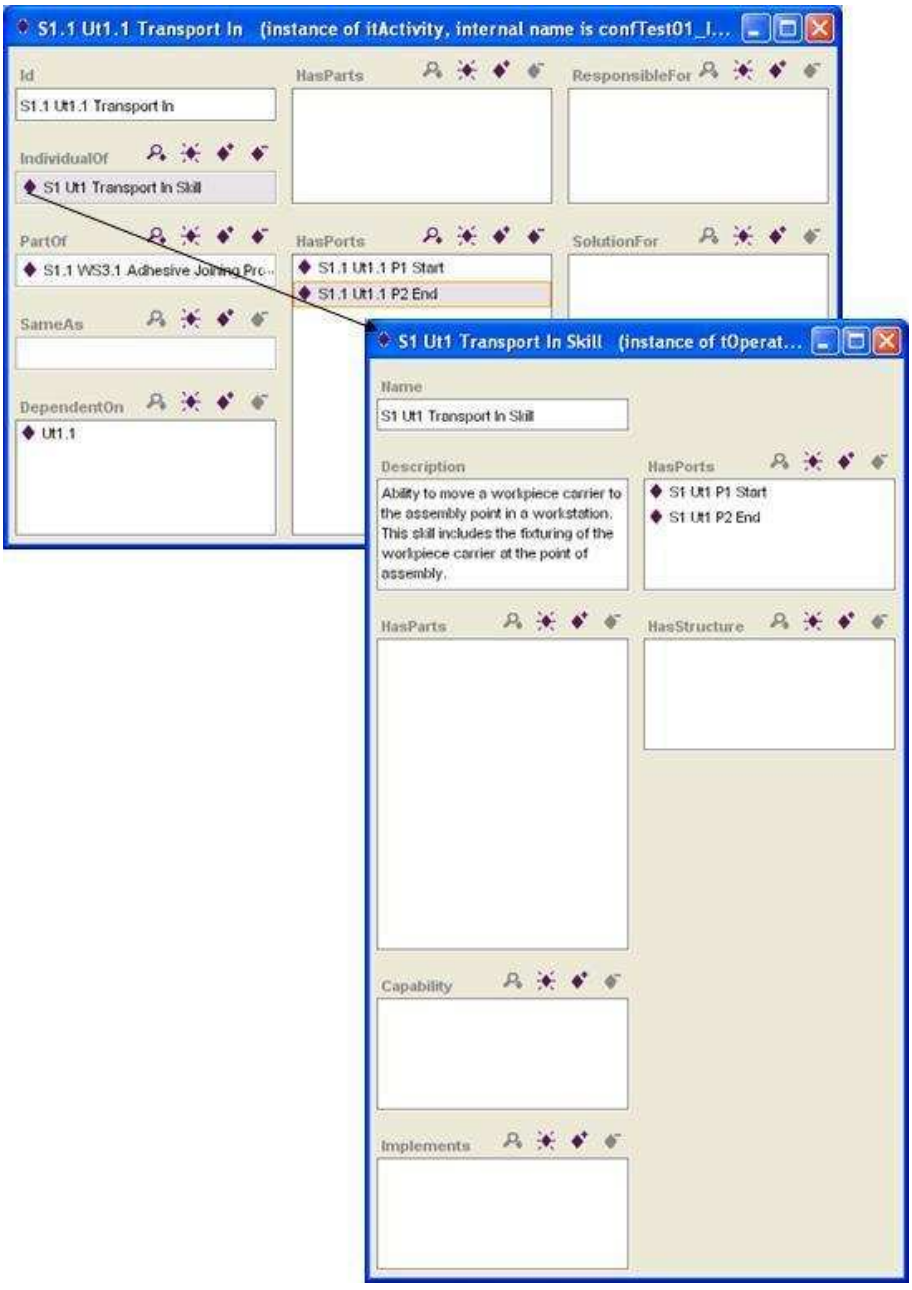


Figure E.27 – Example of a process (Skill) instance

Once the assembly processes (Skills) have been selected and added one needs to establish their sequence. This is achieved by connecting the respective input and output ports of individual process skill. The creation of a connection is a simple process for the user as the tool maintains all the required constraints imposed by the process requirements and semantics of the underlying model. To establish a connection one needs to tick the "Connections" check box and proceed to select which processes (Skills) to plug together. **Figure E.28** shows the window that guides the user after selecting the source process (Skill) and the target process (Skill) establishing the control connection and process (Skill) sequence.

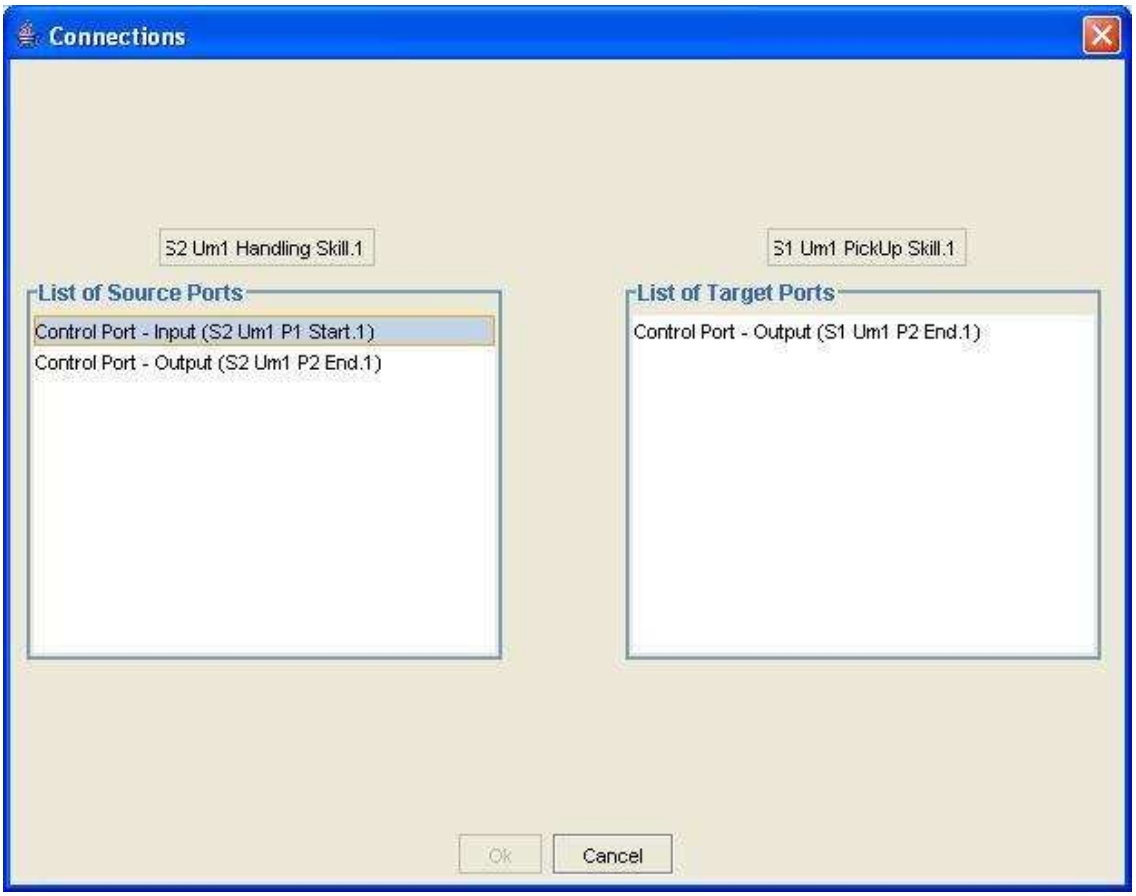


Figure E.28 – Example of the Connection creation

Once the control ports for the connection have been selected the tool creates the connection. **Figure E.29** shows how the connection between two processes (Skills) is defined.

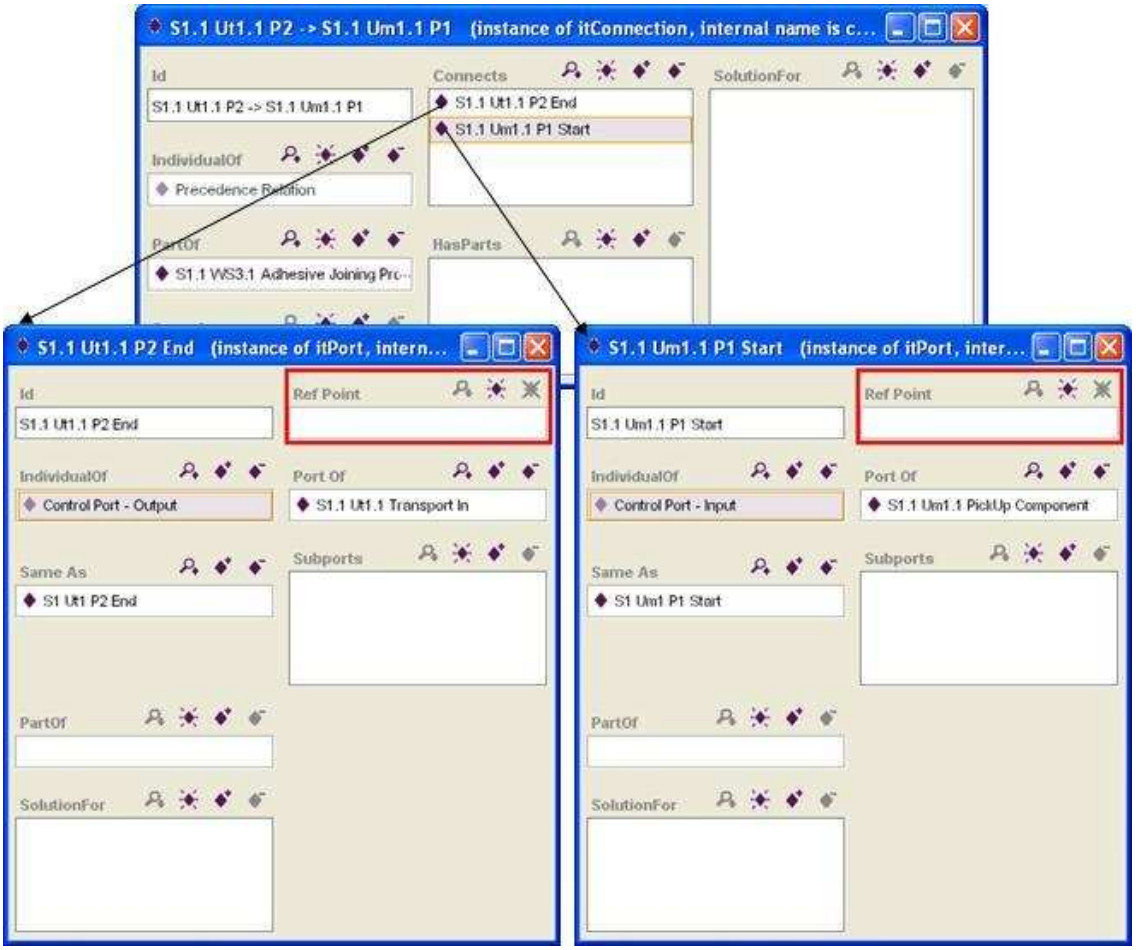


Figure E.29 - Example Connection between two processes (Skills)

The step following the process configuration is to evaluate the system to assess its capabilities and validate it against the requirements. Once a whole system or subsystem has been defined the cost evaluation and simulation tools can be triggered to provide some feedback on the overall cost of the system and its expected performance. Cycle times, expected utilisation, and bottle necks in the system can be identified and used as bases for further fine-tuning or to select alternative system or subsystem configurations. A number of iterations may be required between all the tools, depending on the results of the evaluation and validation, to achieve a more optimal solution.