José António Barata de Oliveira

Coalition Based Approach for Shop Floor Agility – A Multiagent Approach

Dissertation submitted for a PhD degree in Electrical Engineering, speciality of Robotics and Integrated Manufacturing from the Universidade Nova de Lisboa, Faculdade de Ciências e Tecnologia, under the supervision of Professor Luís Camarinha de Matos.

Lisboa

2003

To Tomboco, the small village in the mother Africa where I grew up,
and learned to appreciate the importance
and beauty of the plurality of people.

A word of thanks also to the seven seas and all
the sailors that have sailed upon with me.

Acknowledgements

There are certain individuals that render the journey simpler, others which render it bearable and a few without whom one would probably never make it. I would hereby like to thank all of those that, in one or more of the above mentioned ways, have contributed to this work.

A special recognition should be made to the person that has well and truly been the instigator, motivator and spiritual mentor to my work: Prof. Luís Camarinha de Matos. He is probably the most important reason for the achievement herewith being presented, and nothing I could do would ever thank him. This thesis has received a lot of his valuable criticism and suggestions, without which this work would be far lesser. All the years that I have been working with him have been very rewarding both in scientific and personal terms. He has always been willing to share with me his vast scientific knowledge and experience as well as his indefatigable attitude towards science. My gratitude to all the effort he has made to help me become a better scientist, engineer, and teacher, is endless. But I am also grateful to his friendship and personal support I have had from him since I joined his group. I would like to extend my thanks also to his wife and kids that have been always very helpful and friendly.

Another individual that I also owe big recognition to is Mauro Onori. Nothing I can do will ever thank him for his friendship and brotherhood. He is one of those individuals, who is always there giving his support without expecting anything in return. In addition to reading this thesis thoroughly, giving valuable comments and suggestions, he has also undertaken the hard task of correcting my English. I am also grateful for the various scientific and technical discussions we have had, which have been immensely valuable for the quality of this work. Moreover his continuous encouraging and open mind

to accept completely different views from his own have been a valuable support. Thanks Mauro, soul brother of our mother Africa.

A special word of thanks goes for my students. They have all given me the joy and motivation to pursue this work. Teaching future engineers is a challenge and pleasant activity. In this process of trying to provide them with the right tools and skills to master their future as engineers, and responsible citizens, I am continuously learning with their youth and eagerness towards knowledge. A special thank is addressed to the students that have in a way or another worked with me and contributed to this thesis: João Carlos Silva, Sandra Gadanho, Nuno, Florbela Queiroz, Luís Flores, Rui Monteiro, David Costa, Luís Mendes, Paulo Compadrinho, Rui Rodrigues, Bruno, João Sarraipa, Eduardo Costa, Inês Oliveira, Filipa Ferrada, André Grosso, and João Praça.

My colleagues from the department where I work (DEE) should also be thanked for their support and friendship. Things get easier when the people with whom you need to interact are more than easy to deal with, when they are helpful. The people in this department, from professors to auxiliary personnel, are all responsible for the creation of a pleasant and helpful atmosphere that has been extremely important for my well-being. A warm thanks to all of you.

My current and past PhD colleagues also deserve a special word of thanks. All of them have engaged me in important scientific and technical discussions that proved beneficial for this thesis. On the other hand, they were also a source of encouragement whenever I needed it. Thanks, Hélder Pita, Luís Osório, Ricardo Rabelo, Fernando Martinelli, Octavio Castolo, Celson Lima, and Walter Vieira.

During the realisation of this work I have interacted with many knowledgeable and interesting people. In particular I have worked and learned a lot in European research projects, involving researchers both from Europe and Latin America. I am grateful to all of them and I will not forget how much I have learned with them through the uncountable discussions we have had, that eventually proved an important tool in the building of the work described in this thesis. However, there are some individuals from these projects that I need to individualise because of their help, support, and friendship: Uwe Kirchhoff (ATB Bremen), José Lastra (TUT Tampere), Rolf Bernhardt (IPK Berlin), Volker Katshinsky (IPK Berlin), Gilberto Cunha (UFRGS Porto Alegre), Dagmar Menken (Airbus Hamburg), Raimo Launonen (VTT Finland), and Goran Lange (Univ. of Stockholm).

I am also grateful to the following people from the Portuguese industry that in a way or another have provided me with contacts and visits that have helped me in getting a better picture of the real industrial world: Alberto Gavinhos and António Pires from AutoEuropa, Joaquim Martins and Paulo Compadrinho from Hidromac, and Rui Martins from Grundig.

To Luís Flores I am grateful for all his availability to discuss and help in many of the contents in this thesis related to industrial application.

I am also indebted to my English teachers at the Cambridge School of Almada, who have quite an important share of responsibility in this thesis. Together with my colleagues and supporting staff they have made classes enjoyable and learning much easier. I shall thank Tony Hilzbrich (Bill), Tina Sealey, Joanne Marsden, Robert Hart, and Adrian Matters. In particular I would like to express my gratitude to Joanne Marsden for all her effort in proofreading some chapters of this thesis. Nevertheless, any errors in the English are entirely my responsibility.

My friends João Lisboa and Luís Assunção whose friendship and attention have been important, also deserve a word of recognition.

I would like to thank my parents for being always supportive in this journey. Even at a cost of their own well-being they have always encouraged and supported my studies. I am tremendously grateful for that as well as for all their lessons that encouraged me to pursue learning and knowledge for life.

I cannot also forget my sister Tucha, which, from my earliest memories, has been always there, helping without expecting anything in return.

I left my wife and my three kids to the end not because they have contributed least to this thesis but precisely because they are, in my opinion, the ones to whom I owe more and to whom I shall be more grateful. My kids João, Inês, and José must be thanked for several reasons. First of all, because they have refrained from using the uncanny capacity which kids have to stop any parent from doing anything. They have been refrained too many times from doing many things they could have done, without complaining. My absence ha been always understood with love and tenderness that helped more than anything to pursue my goal. I am eternally grateful to you João, Inês, and Zezito. To the person that was most affected by my work and absence, there are probably no words that can be said that really express what she deserves. My wife Celia has been overcharged by the work she burdened in order for me to finish this thesis. Without her love, tenderness, and support in dealing with almost everything related to the family, not connected with my work, it would be impossible for me to finish this work, as well as having the necessary emotional stability that is required. The only word I can say is thanks my northern star, I owe you so much that nothing in my power could ever return the courtesy.

Abstract

This thesis addresses the problem of shop floor agility. In order to cope with the disturbances and uncertainties that characterise the current business scenarios faced by manufacturing companies, the capability of their shop floors needs to be improved quickly, such that these shop floors may be adapted, changed or become easily modifiable (shop floor reengineering).

One of the critical elements in any shop floor reengineering process is the way the control/supervision architecture is changed or modified to accommodate for the new processes and equipment. This thesis, therefore, proposes an architecture to support the fast adaptation or changes in the control/supervision architecture. This architecture postulates that manufacturing systems are no more than compositions of modularised manufacturing components whose interactions when aggregated are governed by contractual mechanisms that favour configuration over reprogramming.

A multiagent based reference architecture called Coalition Based Approach for Shop floor Agility – CoBASA, was created to support fast adaptation and changes of shop floor control architectures with minimal effort. The coalitions are composed of agentified manufacturing components (modules), whose relationships within the coalitions are governed by contracts that are configured whenever a coalition is established. Creating and changing a coalition do not involve programming effort because it only requires changes to the contract that regulates it.

Glossary of Abbreviations

ACL	Agent Communication Language
AI	Artificial Intelligence
AMI	Agent Machine Interface
API	Application Programming Interface
AUML	Agent Unified Modelling Language
BA	Broker Agent
BAS	Balanced Automation Systems
BIS	Business Information Systems
BPR	Business Process Reengineering
CA	Coordinating Agent
CAC	Cluster Adhesion Contract
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CAQ	Computer Aided Quality
CAPP	Computer Aided Process Planning
CIM	Computer Integrated Manufacturing
CMgA	Cluster Manager Agent
CNC	Computer Numerical Controls
CoBASA	Coalition Based Approach for Shop Floor Agility
DAI	Distributed Artificial Intelligence
ERP	Enterprise Resources Planning
FAA	Flexible Automatic Assembly

FAS	Flexible Assembly System
FMS	Flexible Manufacturing System
ICT	Information and Communication Technology
IGES	Initial Graphics Exchange Standard
IT	Information Technology
JADE	Java Agent Development Environment
JIT	Just In Time
MAS	Multiagent Systems
MCC	Multilateral Coalition Contract
MES	Manufacturing Execution System
MRA	Manufacturing Resource Agent
MRP	Material Requirements Planning
MRP II	Manufacturing Resources Planning
NC	Numerical Control
OKBC	Open Knowledge Base Connectivity
PMC	Physical Manufacturing Component
PPC	Production Planning and Control
PVC	Professional Virtual Communities
PDM	Product Data Management
PDT	Product Data Technology
RMS	Reconfigurable Manufacturing Systems
SFCS	Shop Floor Control Systems
SME	Small and Medium Enterprise
UML	Unified Modelling Language
VE	Virtual Enterprise
VO	Virtual Organisation

Table of Contents

Table of Figures

Table of Tables

1 Introduction

This thesis addresses the problem of shop floor agility. In order to cope with the disturbances and uncertainties that characterise the current business scenarios manufacturing companies need to improve the capability of their shop floors in order to be to be quickly adapted or easily modifiable (agile shop floor reengineering). One of the critical elements in any shop floor reengineering process is the way the control/supervision system is changed or modified. Addressing this need, an approach and system's architecture is proposed to support the fast adaptation of agile shop floor control/supervision systems. According to this approach manufacturing systems are no more than compositions of modularised manufacturing components whose interactions, when aggregated, are governed by contractual mechanisms that favour configuration over reprogramming.

1.1 RESEARCH PROBLEM

Shop floor agility is a central problem in current manufacturing companies. Internal and external constraints, such as growing number of product variants and volatile markets, are changing the way these companies operate by requiring continuous adaptations or reconfigurations of their shop floors. This need for continuous shop floor changes is so important that finding a solution to this problem would offer a competitive advantage to contemporary manufacturing companies.

The central issue is, therefore, which techniques, methods, and tools are appropriate to address shop floors whose life cycles are no more static but show high level of dynamics. In other words, how to make the process of changing and adapting the shop floor fast, cost effective, and easy. The long history of industrial systems automation shows that the problem of developing and maintaining agile shop floors cannot be solved without an integrated view, which accommodate the different

perspectives and actors involved in the various phases of the life cycle of these systems. Moreover, supporting methods and tools should be designed and developed to accommodate the continuous evolution of the manufacturing systems along their life cycle phases – a problem of shop floor reengineering. The design and development of a methodology to address shop floor reengineering is thus an important research issue aiming to improve shop floor agility, and, therefore, increasing the global competitiveness of contemporary manufacturing companies.

Before detailing the focus of the present work in the context of shop floor agility, the concept of agility will be further detailed. Agility is a fundamental requirement for modern manufacturing companies in order to face challenges provoked by the globalisation, changes on environment and working conditions regulations, improved standards for quality, fast technological mutation, and changes of the production paradigms. The turbulent and continuous market changes have impacts at different levels, from company management to shop floor. Only companies that exhibit highly adaptable structures and processes can cope with such harsh environments. Furthermore, the capability to rapidly change the shop floor infrastructure is a fundamental condition to allow participation of manufacturing enterprises in dynamic cooperative networks. Networked enterprise associations, such as virtual enterprises, advanced supply chains, etc. are examples of cooperative structures created to cope with the mentioned aspects. Manufacturing companies wishing to join these networked structures need to be highly adaptable in order to cope with the requirements imposed by very dynamic and unpredictable changes. In such scenarios, agility means more than being flexible or lean. Flexibility in this context means that a company can easily adapt itself to produce a range of products (mostly predetermined), while lean essentially means producing without waste. On the other hand, agility corresponds to operating efficiently but in a competitive environment dominated by change and uncertainty (Goldman, Nagel, & Preiss, 1995), which means adaptation to conditions that are not determined or foreseen a-priori. The participation in dynamic (and temporary) organisations requires agile adaptation of the enterprise to each new business scenario, namely in terms of its manufacturing capabilities, processes, capacities, etc.

It is worth noting that the need of methods and tools to manage the process of change was first felt at the company's higher management levels. This is not surprising because the external business conditions are initially felt at managerial levels. Therefore, in past research the processes of change (reengineering/adaptation) have been addressed mostly at the level of business process reengineering and information technology infrastructures. Little attention, however, has been devoted to the changes needed at the manufacturing system level and, yet, the shop floor suffers a continuous evolution along its life cycle and it is subject to ever increasing demands on its flexibility. In fact, despite the efforts put in the creation of agile organisational structures, little attention has been devoted to the agility of the shop floor, even if many research works have been focused on flexible assembly and flexible manufacturing systems (Gullander, 1999; Onori, 1996; Vos, 2001; Zwegers, 1998). There are some

research works (Huff & Edwards, 1999; Koren et al., 1999; Mehrabi, Ulsoy, & Koren, 2000), in which shop floor agility is achieved by focusing on the reconfigurability of the individual equipment rather than considering a global agility approach. Nevertheless the situation is that a non-agile shop floor seriously limits the global agility of a manufacturing company even if its higher levels are agile. A good indication of how great the demand for agile shops-floors is within manufacturing companies is the increasing number of shop floor alteration projects (Barata & Camarinha-Matos, 2000). As long as people in the shop floor are faced with the need to often change (adapt) their production systems, the need to have methods and tools to cope with such challenge increases significantly.

Consequently, the work described in this thesis, besides being important in terms of the mentioned practical need, is pioneering because to the author's knowledge it is the first one that attempts a holistic approach to the agility problem.

Nonetheless, it must be pointed out that the aim of this thesis cannot be that of solving the global shop floor agility problem since this would demand the development of several methods and tools, which would demand a scale of resources that goes beyond the scope of a PhD work. The goal is therefore to present an approach to the problem which applies a holistic perspective and in which partial contributions to the global solution are expected. Once the global problem is understood, and a sub-problem is focused upon, the thesis attempts to propose a solution to it.

A particularly critical element in a shop floor reengineering process is the control system. Current control/supervision systems are not agile because any shop floor change requires programming modifications, which imply the need for qualified programmers, usually not available in manufacturing SMEs. To worsen the situation, the changes (even small changes) might affect the global system architecture, which inevitably increases the programming effort and the potential for side-effect errors. It is therefore vital to develop approaches, and new methods and tools that eliminate or reduce these problems, making the process of change (re-engineering) faster and easier, focusing on *configuration* instead of *codification*. Hence the decision to focus this PhD work on the reengineering aspects required by the control/supervision architecture, which covers an important part of any global life cycle support methodology.

The proposed contributions to improve the shop floor reengineering task aim at accommodating the following requirements:

- **Good level of modularity**. Manufacturing systems should be created as compositions of modularised manufacturing components, which become basic building blocks. The building blocks should be developed on the basis of the processes they are to cater for.
- **Configuration rather than programming**. The addition or removal of any manufacturing component (basic building block) should be done smoothly, without or with minimal

programming effort. The system composition and its behaviour are established by configuring the relationships among modules, using contractual mechanisms.

- **High reusability**. The building blocks should be reused for as long as possible, and easily updated for further reuse.
- **Legacy systems migration**. Legacy and heterogeneous controllers should be considered in the global architectures and a process should be found out to integrate them in the new agile architecture.

Reducing the programming effort that is usually required whenever any changes or adaptations take place in the shop floor becomes one of the most important requirements for this thesis.

The research question addressed by this work can be summarised as:

Research question

Which methods and tools should be developed to make current manufacturing control/supervision systems reusable and swiftly modifiable?

The adopted work hypothesis to address the research question is defined bellow:

Research Hypothesis

Shop floor control/supervision reengineering agility can be achieved if manufacturing systems are abstracted as compositions of modularised manufacturing components that can be reused whenever necessary, and, whose interactions are specified using configuration rather than reprogramming.

The approach followed to tackle the problem raised in the research question was the following:

Approach

- The life cycle of shop floor manufacturing systems should explicitly include a new phase: the reengineering phase that captures the time frame in which the systems are being changed or adapted (reengineered).
- Multiagent based systems seem to be a good modelling and implementation paradigm because of their adequacy to create cooperative environments of heterogeneous entities.

- Manufacturing components are agentified (transformed from physical manufacturing components into agents) to become modules that can be used and reused to compose complex systems.
- The different types of manufacturing systems are represented by coalitions or consortia of agentified manufacturing components, which are essentially societies of self-interested and heterogeneous agents whose behaviour is governed by contracts.
- Contract negotiation is the configuration basis required whenever a control/supervision system needs to be changed or adapted.

The main scientific contributions that have been developed under the framework of this thesis are:

Main Contributions

1. Shop floor agility is achieved by using coalitions/consortia of agentified manufacturing components (basic building blocks), whose members are chosen from an entity (cluster) that contains all the manufacturing components of a certain cell, using a broker agent as an intermediary.
2. A multiagent based architecture, called **Co**alition **B**ased **A**pproach for **S**hop Floor **A**gility – **CoBASA**, was created to support the reengineering process of shop floor control/supervision architectures. In an innovative way, CoBASA uses contracts to govern the relationships between coalitions members (manufacturing agents).
3. The postulation of a new methodological approach in which the reengineering process is included within the life cycle. Note that the multiagent approach requires the manufacturing community to structure and classify the process involved, thus leading to a more systematic or structured methodological approach.

The work described in this thesis assumes that there is a similarity between the proposed reengineering process and the formation of consortia regulated by contracts in networked enterprise organisations. The problems a company faces in order to join a consortium are analogous to the shop floor adaptation problem. In other words, the formation of a coalition of enterprises to respond to a business opportunity is analogous to the organisation of a set of manufacturing resources in order to perform a given job. The proposed approach is therefore to use the mechanisms and principles developed to support the enterprise integration into dynamic enterprise networks as inspiration for an agile shop floor reengineering process.

1.2 OUTLINE OF THE THESIS

This thesis is organised in six chapters: Introduction, Manufacturing Trends and Shop Floor Reengineering, Supporting Concepts, System Architecture to Support Shop Floor Reengineering, Prototype and Validation, and Conclusions and Future Work.

The **Introduction** i.e. the current chapter, motivates the reader for the domain under analysis, and briefly outlines the research problem as well as the approach followed in the research. In the end of the chapter the thesis is outlined.

Chapter 2 (**Manufacturing Trends and Shop Floor Reengineering**) presents an historical perspective of the evolution of manufacturing/business paradigms (section 2.1) to show that there is a direct connection between these paradigms and manufacturing systems requirements. This description shows that the requirements for shop floor agility start to become determinant only in the case of the more recent business/manufacturing paradigms, what highlights the importance of shop floor reengineering, and consequently, of this thesis. The enablers and barriers that constrain the evolution of business paradigms are also discussed in order to give the reader some insight on which factors, in the end, constrain the requirements of control/supervision architectures. After the historical perspective the current challenges faced by manufacturing companies are emphasised, and, in particular, it is clarified why manufacturing companies really need to solve the shop floor reengineering problem (section 2.2). After this, a suggested shop floor reengineering methodology scenario to face the indicated manufacturing challenges is described in section 2.3. Finally some concluding observations are discussed in section 2.4.

Chapter 3 (**Supporting Concepts**) describes the underlying concepts used to framework the architecture being proposed (CoBASA). It provides the reader with insight on the themes that contributed to structure and build CoBASA, which facilitates the reader's understanding of the architecture and the concepts behind it. Moreover, it shows that the problem of shop floor reengineering, or more generically shop floor agility, can only be addressed by integrating or considering a large number of domains.

In this context the most relevant aspects about multiagents are described in section 3.2 to enable the reader to better understand the main concepts behind CoBASA. Next, ontologies are described in section 3.3 since they are the basic mechanism used by CoBASA agents to have a common understanding of concepts. Since contracts are the key mechanism that regulates the interaction among agents in the coalitions of manufacturing agents, section 3.4 briefly introduces the basic principles and mechanisms of the law of contracts. Shop floor control is described in section 3.5 to give the reader information about the current state of the art on the main focus of the work being described in this thesis. Collaborative organisations are briefly described in section 3.6 because some of the CoBASA

concepts were inspired on them. Finally in section 3.7 integration infrastructures are described to emphasise the importance of integration in this context.

Chapter 4 (**System Architecture to Support Shop Floor Reengineering**) describes the most important contribution from this thesis: the CoBASA architecture. This multiagent based architecture and methodology supports the shop floor control/supervision reengineering. The basic approach followed by CoBASA is described in section 4.1. Section 4.2, which introduces the global architecture, details the basic components that compose the architecture, and the cluster and coalition concepts. At the end of the section the life cycle phases of a coalition are described. Finally, in section 4.3 the architecture of each individual CoBASA component is detailed.

Chapter 5 (**Prototype and validation**) includes the aspects related to the implementation of a validation prototype based on the CoBASA architecture, and the validation of the proposed approach. The approach followed to validate the work is discussed in section 5.1. The implementation of a multiagent based software architecture cannot be successfully achieved in a feasible time frame without the help of supporting tools. Therefore, it was of outmost importance to find two supporting tools for two relevant activities: multiagent development environment, and the ontology support. Section 5.2 briefly describes the selected tools and why they were chosen. In section 5.3 the implementation details of the prototype are outlined, in which snapshots of user interfaces of the basic agents that compose CoBASA are shown. Section 5.4 describes the main steps required to operate CoBASA, mainly by describing what are the steps required to transform manufacturing components into agents that, later on, might be used as candidates to participate in future coalitions, as well as how coalitions are created, changed, and deleted. In sections 5.5, 5.6, and 5.7 the contributions of this thesis are validated. Section 5.5 is about the experiments that have been developed to support experimental validation, while section 5.6 is about external validation. Finally, section 5.7 discusses the validation of the thesis being proposed using a theoretical approach.

Chapter 6 (**Conclusions and Future Work**) discusses the main conclusions of this work. The main results and contributions are summarised, and some possible directions for further research are mentioned.

2 Manufacturing Trends and Shop Floor Reengineering

This chapter presents an historical perspective of the evolution of manufacturing business paradigms (section 2.1) to show that there is a direct connection between these paradigms and manufacturing systems requirements. This description shows that the requirements for shop floor agility are important in recent manufacturing business paradigms, which highlights the importance of shop floor reengineering and consequently of this thesis. The enablers and barrier factors that constrain the evolution of business paradigms are also discussed to give the reader some insight into which factors, in the end, constrain the requirements of control and supervision architectures. After the historical perspective, the current challenges faced by manufacturing companies are emphasise, and, in particular, it is clarified why there is a need to solve the shop floor reengineering problem (section 2.2).

2.1 HISTORICAL PERSPECTIVE

Before detailing the evolution of manufacturing it may be worthwhile to briefly discuss what influences manufacturing requirements and how these requirements affect the control and supervision architecture design. Figure 2-1 illustrates that the manufacturing requirements imposed on manufacturing companies at a certain time are based on a set of external factors. These factors can be broadly grouped into human factors, business trends, and technological evolution. “Human Factors” include aspects related to the impact customers, as well as workers and entrepreneurs, have on the definition of a manufacturing system. Economic recessions and booms and their consequences are included in “Business Trends”.

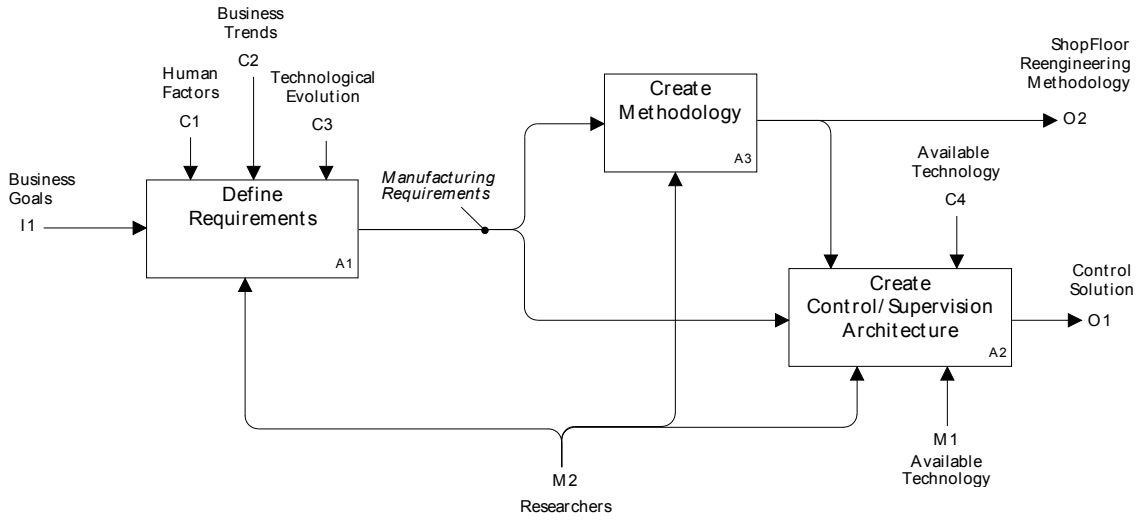

Figure 2-1 – Activities to create a manufacturing control/supervision architecture

"Business Trends" group all those external business factors that directly and indirectly influence the way manufacturing systems are organised. Market trends, for instance, are included here. Finally, "Technological Evolution" represents those factors that are related to how technology influences manufacturing requirements. The technological evolution suffered by a certain product, for instance, has a direct impact on the way this product is manufactured (process) (Erixon, 1996).

To create a control and supervision architecture for a manufacturing system, which is able to respond to the company's business goals, it is first necessary to define the requirements needed by that manufacturing system. These requirements are then used to create a suitable methodology that will be applied to the development of the control and supervision architecture, which, in turn, transforms the manufacturing requirements into a suitable control architecture. This activity is also influenced by the available technology. The set of manufacturing requirements as well as the architecture varies according to the period being considered. An architecture might be totally adequate for one period and completely inadequate in another one, just because the factors that constrain the process have changed.

The various manufacturing business paradigms and their temporal evolution are indicated before detailing the various external factors, and their temporal evolution. While some factors are typically associated with a period or paradigm, others cover more than one period.

A remark should be made about the discrepancies between what is technologically possible and what is in fact applied. In flexible automation, for instance, although many advanced control and supervision architectures have been developed since the 1980s, few of them are being used on the shop floor. System vendors, in many cases, impose technology and, unfortunately, they tend to restrain technological evolution for the sake of profit. This is particularly true for SMEs that do not usually have enough engineering resources able to spend time thinking and analysing the most adequate solutions for their problems. Big companies, on the other hand, tend to let their shop floor managers come to a compromise with big system vendors.

2.1.1 Manufacturing business paradigms

Figure 2-2 shows the manufacturing business paradigms that characterise the various periods from the nineteenth century to the twenty-first century. Only the most important ones are shown for the sake of clarity. On the other hand, this section's goal is more about giving an historical perspective for the framework of this thesis than providing a detailed record of historical facts. The different important ages in the considered time frame as considered by several authors (Clarke & Clegg, 1998; De Masi, 2000; Negroponte, 1996; Toffler, 1981) are also indicated in the figure.

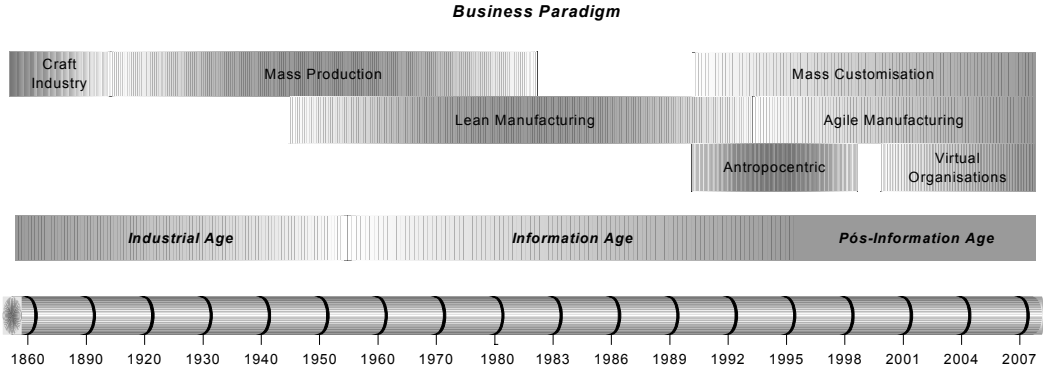

Figure 2-2 – Business paradigms

The transition from one age to another is not abrupt, in the sense that there is a clear point in time in which one age is considered terminated and the other starts. On the contrary, during the transition from one age to another the two ages might overlap for a long time. Nevertheless, this transition time is, in general, turbulent because while one age is not really superseding the other there is a lot of ambiguity. The same happens when transiting from one paradigm to another. Turbulence as a landmark for paradigm changing was pioneered by Thomas Khun (Kuhn, 1970). A key feature of his theory is the emphasis placed on the revolutionary character of scientific progress, where a revolution involves the abandonment of one theoretical structure and its replacement by another incompatible one. The disorganised and diverse activities that precedes the formation of a new age or paradigm eventually becomes structured and directed when a new age or paradigm becomes adhered to by a group of people (Chalmers, 1999).

Changing from one age to another involves many aspects: business, political, and social. While the manufacturing business paradigms are only related to business and manufacturing, ages involve other aspects.

The **industrial age** is considered to have started in the middle of the eighteenth century, with the advent of the industrial revolution, up to the middle of the twentieth century (De Masi, 2000).

The disruptive factor that leads to this age was the invention of the steam engine. Another relevant aspect of this age is the greater importance of science, which in fact greatly contributed to its development. The most important theoretical work behind this age is Adam Smith's book "Wealth of

Nations" (Smith, 2000), which was written in 1776. The following factors contributed to the rise of this age:

- Availability of massive supplies of raw materials, such as timber, iron ore, oil and other resources.
- Development of new inventions and technologies.
- Existence of a large labour force constantly replenished by immigration.
- Emergence of highly talented, but often unscrupulous business leaders.

The generic characteristics of the industrial age, according to a sociological/political view (De Masi, 2000; Toffler, 1981), are indicated in Table 2-1.

❑ Concentration of employed workers in factories ❑ Predominance of workers from the secondary over the primary and tertiary sectors ❑ The gross national product (GNP) is predominantly formed by industry ❑ Application of scientific discoveries in the industry ❑ The work is progressively organised and scientific methods are used in its organisation ❑ Division of labour and its fragmentation ❑ Clear separation between the living place and the working place ❑ Progressive urbanisation and increased education	❑ Fewer social inequalities ❑ More geographical and social mobility ❑ Increased mass produced products and increased consumption ❑ The life of humans is now more synchronised with the pace of machines than with nature ❑ Thinkers are separated from those that do the work (doers) ❑ The only important criteria to optimise the production are productivity and efficiency ❑ Labour relations are characterised by conflicts between employers and employees ❑ Nations are organised in a hierarchy based on the GNP, the ownership of raw materials and means of production

Table 2-1 – Main characteristics of the industrial age

A management view of the important aspects of the industrial age, adapted from (Hames, 1999) are indicated in Table 2-2.

An important characteristic of the industrial age, identified by (Clarke & Clegg, 1998; De Masi, 2000; Hames, 1999), is the diffusion of mass production as a key characteristic of the mature industrial age. For some authors (Bell, 2000; Kornhauser, 1960; Shils, 1960) the mass society, which is a consequence of a mature mass production, was the most important cause for the transition from the industrial age to the information age. For these authors, the economic growth provided by the mass production increased the welfare of people, which became more participative and aware of their role in society. In the sixties of the last century it was believed that mass production and mass consumption would lead the world to an unprecedented development. However, this did not happen, and, on the

contrary, a severe crisis hit the world in 1973 due to the oil shock. Despite this, the crisis had already become visible in western countries since the sixties due to strong social movements that were the sign that a new era was coming (the hippie movement is just an example). In addition to the social revolution underway, the influence of electronics and computer technology was also starting to be perceptible. The dissemination of electronic equipment among consumers was already a reality while computers were starting to show up, not really to the normal consumer, but in business circles. Although there is no consensus about why the industrial age came to a crisis, there is some consensus that the dissemination of electronics and computers are the basis for the change from the industrial age to the information age, mainly because computers and electronic equipment became so disseminated that new forms of communication and organisation became possible. The disruptive factor that led to this age is thus the computer and electronic technology.

❑ Focus on measurable outcomes. Physical products ❑ Highly specialised knowledge base resulting in single-skilling ❑ Individual accountability ❑ Life long employability ❑ Clearly differentiated and segmented organisational positions, roles, and responsibilities ❑ Linear input-output thinking about programmes ❑ Reactive in solving problems ❑ Local perspective informs programming ❑ Economies of scale	❑ Centralised command ❑ Hierarchical linear information flows and bureaucracy ❑ Internal control ❑ Attention to quantitative differences ❑ The assets are capital, labour, and resources ❑ Achieving effectiveness through methods ❑ Initiatives for improvement emanates from a management elite – blue collar workers ❑ Doing what is known now (present oriented) ❑ Standardisation

Table 2-2 – Industrial age aspects according to a management view

The **information age** started to be felt after the dissemination of communication means (based on electronics and computers) and computers among the masses of people, which is considered to have happened in the seventies of the last century. However, this is not consensual since economists, sociologists, and researchers of politics consider that the information age started with the already mentioned crisis of the industrial age (late fifties of the last century), which was before the dissemination of computers. According to them, the economic growth and social welfare changed the conditions of society in such a way that a new age is needed to characterise the new form. Nevertheless, for technologists (Builder, 1993; Dertouzos, 1997; Gates, Myhrvold, & Rinearson, 1996; Negroponte, 1996; Strackbein, 2002) the information age starts really with the dissemination of computers and computer networks. Computer technology can therefore be seen as the principal catalyst of a social and economic movement.

The information age can also be known by other names depending on the research area. A well known name is third wave, which was coined by Alvin Tofler (Toffler, 1981), who first discussed the impact that the convergence of telecommunications and computer technology would have on the society. Names such as post-industrialism (Beck & Ritter, 1992; Bell, 1976; De Masi, 2000), post-modernism (Lyotard, 1984) are also used as well as second industrial divide (Piore & Sabel, 1984).

Daniel Bell states that the industrial age naturally evolves into post-industrial (service) economies, and sees the information sector as part of the infrastructure for a post-industrial society (Bell, 1981). However, Manuel Castells has a view in which communication technologies and networks have a much stronger influence on society. According to him, after the industrial age came the information age, which is based on the pervasive implementation of technological innovation since the 1970s, clustering around the convergence of computing and telecommunication (Castells, 1996). Manders and Brenner, who state that the introduction of computers in the 1970s into the realm of machine control provided greater flexibility into the organisation of the production, support the same idea (Manders & Brenner, 1999). Peter Drucker, one of the most outspoken management theorists of the twentieth century, has also examined the implications of post-industrialism for management practice (Drucker, 1969, 1992). Another well-known management theorist that has written about this turbulent time is Charles Handy, who says in (Handy, 1991, 1994), *"Today the language is not that of engineering but of politics, with talk of cultures and networks, of teams and coalitions, of influence or power rather than control, of leadership not management"*.

Compared to the information age the industrial age was "perfect and predictable", with "perfect" and uniform factories and machines, and the people that worked there were also alike. In a way, dehumanisation (because human beings are far from being uniform and perfect) is the ultimate consequence of the Industrial Age thinking. This age of uniformity, standardisation, and "perfection" gave way to the age of variety and turbulence as stated in (Strackbein, 1998). Words like *excellence*, *leading edge*, *high performing*, or *innovative* are widely used, and characterise a turbulent world where innovation is mandatory. Another key characteristic is more differentiated products (customised for the customer), shorter product life cycles, and services associated with the products. Customers, on the other hand, are becoming more and more demanding. Manuel Castells considers that these characteristics have occurred in the shift from "industrialism" to "informationalism" (Castells, 1997a).

The main important aspects of the information age, adapted from (Hall, 1995; Hames, 1999; Morrison & Schmid, 1997; Naisbitt, 1982), are presented in Table 2-3.

Many of the aspects mentioned in Table 2-3 are also a characteristic of the **post-information age**, which is also indicated in Figure 2-2. It was not mandatory to consider such an age because the transition from the information age is quite smooth and the step was not as big as the one that happened for the transition from the industrial age into the information age. However, since the emergence of computer networks and the World Wide Web in the nineties, and their strong impact on the way people and companies establish cooperation and do business, it was decided to consider this

age because there was some disruption from the previous time. The disruptive factor is computer networks. Nicholas Negroponte in his book "Being Digital" (Negroponte, 1996) refers:

> *"The transition from an industrial age to a post-industrial or information age has been discussed so much and for so long that we may not have noticed that we are passing into a post-information age. The industrial age, very much an age of atoms gave us the concept of mass production, with the economies that come from manufacturing with uniform and repetitious methods in any one given space and time. The information age, the age of computers, showed us the same economies of scale, but with less regard for space and time. (...) In the information age, mass media got bigger and smaller at the same time. New forms of broadcast like CNN and USA Today reached larger audiences and made broadcast broader. (...) In the post-information age, we often have an audience the size of one. Everything is made to order, and information is extremely personalized. A widely held assumption is that individualization is the extrapolation of narrowcasting--you go from a large to a small to a smaller group, ultimately to the individual."*

Concepts such as *information highways* (Berquist & Berquist, 1996; Chung, 1996; Lebow, 1995), *electronic commerce*, *information society* (Bell, 1981; Brown & Duguid, 2000; Castells, 1997b; Dertouzos, 1997; Minsky, 1986), *knowledge management* (Pfeffer & Sutton, 2000), and *virtual organisations* (Camarinha-Matos, 2002a; Camarinha-Matos & Afsarmanesh, 1999a; Camarinha-Matos, Afsarmanesh, & Rabelo, 2001), among others are typical from this age.

❑ High technology	❑ Electronic commerce
❑ Decentralisation	❑ Organisations are made of individuals and networks. Work organised in teamwork. Team accountability
❑ Networking	❑ Short life employability
❑ Multiple options	❑ Integrated marketing with partnership based relationships with suppliers and customers. Integrated view of work and enterprise
❑ Assets are knowledge and people	❑ Lifelong training
❑ People that do the things (doers) are the same as the ones that think	❑ Holistic approach to organisations
❑ Workers with interdisciplinary skills resulting in multi-skilling	❑ More workers in the services than in the industrial sectors
❑ Lean and Agile production	
❑ Market is focused on the consumer – customer satisfaction is first priority. Mass customisation	

Table 2-3 – Main aspects of the information age

2.1.1.1 Craft Production

Craft production is the typical manufacturing paradigm of the end of the nineteenth century, which was mainly mastered and dominated by Europe. Products were made exactly to suit customer needs (one item at a time) using highly skilled workers able to participate in the different phases of the

production process, and simple but flexible tools (Piore & Sabel, 1984; Womack, Jones, & Roos, 1990).

The components were not standardised, which turned the production process into a 'one of a kind' process, where unique products are produced. The lack of standardisation was a drawback because, without it, it is not possible to create interchangeable parts that are the basis to build identical products. No two items were exactly identical and, in many cases, each item was intentionally made different from others. In many craft industries the workers were their own bosses, often serving as independent contractors or as independent machine-shop owners with whom major companies contracted for specific parts or components. Workers applied their own personal know-how, or "tacit knowledge", which is knowledge stored in the minds of individual workers, to create value (Victor & Boynton, 1998).

Despite the lack of standardisation in the European craft industry, there were already in the United States some experiences in terms of standardisation that became the ground basis for mass production. The most well known examples are Colt and Springfield, which used interchangeable parts for producing their products in quantities. Please note that their techniques were closer to mass production than craft. Since it was not pure mass production this system is called the *American System* (Cammarano, 1997; Davidow & Malone, 1993).

Craft companies were decentralised even if concentrated in a single city. These companies, for instance, coordinated in direct contact a set of small machine shops. Suppliers and customers were in direct contact with the company. The world of craft production was a world of small producers, each specialised in one line of work and dependent on the others.

Although the products were expensive and, consequently, only a few customers could afford them, craft companies could respond appropriately and flexibly to unique, unpredictable, and changing markets by relying on workers with tacit knowledge. Although craft companies could produce a great variety of products (product flexibility), this was done at a very slow pace. Changing or adapting the shop floor to new requirements in terms of product changes or quantities produced was very difficult. Therefore, craft industries, although flexible, were not agile.

2.1.1.2 Mass Production

The following conditions, adapted from (Piore & Sabel, 1984; Womack et al., 1990), were the basis for the emergence of mass production in America:

- High costs of the products from the craft industry.
- The low reliability of the craft products, which were prototypes, due to a lack of standardised components.
- The inability of the small shops that characterise the craft industry to produce the innovation required in the products of that time (automobiles, for instance).

- The existence in the USA of a market of immigrants (customers) connected by an effective railroad network and willing and able to purchase crude standardised products. It must be taken into account that their movement to America had erased the diversity of tastes that characterise these immigrants in Europe.
- The fact that these immigrants constituted a short skilled labour force and speaking various languages, which would make their interaction difficult.
- The existence of enough raw materials to fuel inefficient machines.
- The theoretical work about division of labour already developed by Adam Smith (Smith, 2000) and Karl Marx (Marx & Engels, 1990).

Its most important characteristics are: division of labour, interchangeable parts, and mechanisation.

The most remarkable sign of this emergence is the construction of the Ford model T automobile plant at Highland Park in Detroit in 1913. The impact of this plant in the way manufacturing was organised is the reason why Henry Ford (Ford & Crowther, 1988; Gross, 1997) is widely recognised as the fundamental industrialist of the mass production paradigm. Ford coined the name mass production in an article for the Encyclopedia Britanica, in 1926, and is so important that industrial sociologists call this paradigm *Fordism*. Ford's new recast of techniques and concepts such as the continuous-flow assembly line reduced costs while increased product quality and productivity. Before the assembly line, workers had to move from one assembly shop to another rather than the car. By improving the assembly line so that the Model T could be produced ever more inexpensively, Ford placed the car within reach of the average citizen.

Ford also adopted the techniques of Frederick Taylor (Taylor, 1998) by combining them with his own synchronized methods of mass production. Taylor developed time and motion studies, and scientific management methods, which are called *Taylorism*, and are an important philosophy of the mass production paradigm. *Taylorism* increased production without increasing the individual workload, and the work became easier to manage. By incorporating *Taylorism*, *Fordism* resulted in a greater division of labour and the de-skilling of manual labour that implied workers with no freedom to be creative in terms of how their jobs should be completed, which became a downside of mass production.

Another important character definitely associated with this paradigm is Alfred Sloan (Sloan, 1986), who was a contemporary of Henry Ford and the president of General Motors. Sloan's unique contribution to organisational development was the decentralised divisional structure and the use of financial controls. In addition, he also introduced the concept of marketing as an important tool to create a mass market. Such management helped the spread of the mass production paradigm. These concepts, together with what Ford had pioneered (factory practices), completed what is known today as the ***mass production*** paradigm.

The main goal of mass production is producing big quantities of non-diversified products at a reduced price based on the production line and a complex division of labour. It is the customer that

must adapt to the product and not vice-versa. The famous statement from Henry Ford that the customer could have a car of any colour they wanted as long as it was black is a good example of the low variety of products that characterised this paradigm. The paradigm was mastered and dominated by the USA and, later on, was spread through Europe and even to Japan. In the beginning its dissemination through Europe was not easy because the way the industry and the customers were organised were totally different from America. European customers were used to more customised products. Nevertheless, in the end, the paradigm became predominant at the cost of destructing a very important European concept: the industrial district or cluster (Bartezzaghi, 1999; Piore & Sabel, 1984).

The most important aspects of the mass production paradigm are summarised in Table 2-4.

❑ Fixed costs are high ❑ Marginal costs are low ❑ Big quantities ❑ Highly specialised tools ❑ Vertical organisation ❑ Products are not flexible ❑ Great division of labour ❑ The factory is the "centre of the world"	❑ Relations between suppliers and customers are weak ❑ The factory pushes the product to the customer ❑ Quality control is done only after the product is produced ❑ Great number of rework hours ❑ Workers have no autonomy ❑ Market is predictable

Table 2-4 – Mass production characteristics

The mass production paradigm, however, did not stay fixed. In effect, the world was changing and manufacturing companies had to adapt themselves to the evolution that was taking place in society, business, and technology. Customers, for instance, were becoming more sophisticated, and society in general more aware of working conditions. Customers were requiring more and more varied and better quality products. The mechanisation that was characteristic of the beginning of mass production was replaced in the 1960s and 1970s by automation, which resulted in some jobs becoming superfluous (Manders & Brenner, 1999). The advent of automation in manufacturing companies resulted from technological evolution and economic reasons. Cheap and more reliable equipment that could work 24 hours a day (electrical motors, computers, …) became available. On the other hand, the pressure to reduce the price per unit in the production site imposed the need for an increased pace in production that could only be achieved by automating some of the process tasks. The progress of automation created then a labour divide that contributed to a crisis in society. The labour market was divided into highly employed skilled workers, and unemployed unskilled workers. To worsen the situation these last workers swiftly became long term unemployed people, which increased the expenses of the countries, already struggling with an economic downturn. In addition, more and more people considered mass production systems as being boring and a place where workers had very poor working conditions.

Between 1965 and 1973 there was also an economic downturn, which was intensified after the oil shock (1973).

For some time, between the 1950s and the 1970s, the paradigm was able to cope with this changing world. However, the paradigm was becoming less and less able in dealing with an increasingly turbulent world that was moving from the industrial to the information age. To answer these problems, Japanese companies followed one approach while European and American companies followed a different one.

Japanese companies, in effect were already using a model that had been implemented since the early 1960s by Taiichi Ohno (Ohno, 1988). He developed a model that would later become known as Lean Manufacturing. Although less known, the words *Toyotism* and Toyota Production System are also used.

The main characteristic of the shop floors that exist during the mass production paradigm was rigidity. As everything was more or less predictable (no product variability) and no big changes in terms of capacity, shop floors were designed to be optimised in terms of producing as much as possible units at low cost. Therefore, flexibility and agility were not requirements, and consequently, control/supervision architectures were designed to privilege efficiency rather than functionalities to make them changeable. This was the time of specialised machines that, whenever changes were required, they were scrapped and the system was entirely designed.

2.1.1.3 CIM / Flexible Manufacturing Systems / Flexible Assembly Systems

European and American manufacturing companies, in order to find the best path to overcome the crisis, first concentrated on massive investments in technology, namely on automation and software to manage the production. The use and dissemination of heterogeneous computer hardware and software in manufacturing companies, without a global strategy, created a big problem because they could only be effective if all systems could be effectively integrated. Consequently, the integration of different computer systems became a very important research theme in the 1980s and early 1990s – systems integration (Barata & Camarinha-Matos, 1995; Barata, Vieira, & Camarinha-Matos, 1996; Bernhardt, 1992; Camarinha-Matos, Seabra Lopes, & Barata, 1996; Grady, 1994; Karagiannis, 1991; Miller & Walker, 1990; Olsson & Piani, 1992; Stark, 1989).

Another aspect was how to create a global architecture that could model and accommodate the different tasks in a company and, simultaneously, provide an integrated view. The Computer Integrated Manufacturing (CIM) paradigm (Browne, Harhen, & Shivnan, 1988; Camarinha-Matos, Pinheiro-Pita, Rabelo, & Barata, 1995; Hirsch & Actis-Dato, 1987; Miller & Walker, 1990; Mitchell, 1991; Ranky, 1990; Rembold, Blume, & Dillmann, 1985; Scheer, 1991; Waldner, 1992) became an important effort towards the goal of increasing the competitiveness of manufacturing companies through the introduction of automation and wider use of computers.

Companies also realised that flexibility was the key to cope with an increased tendency towards more diversified products and with varying demand. Automation, with the help of computers, could help this effort. Huge investments have, and are still being made to develop the most flexible and totally automated shop floor, often known as the "factory of the future". Concepts like Flexible Assembly Systems (FAS) and Flexible Manufacturing Systems (FMS) (Barata & Camarinha-Matos, 1994, 1995; Cheng, Simmons, & Ritchie, 1997; Kaighobadi & Venkatesh, 1994; Maleki, 1991; Miller & Walker, 1990; Onori, 1996; Parrish, 1990; Vos, 2001) were developed in an effort to create systems able to cope with varying products and demand. FMS consist of CNC machines connected by an automated material-handling system. Its flexibility arises from the automatic tool exchanging mechanisms, the programming of the CNC, and the flexibility of the material handling system. FAS on the other hand consist of assembly stations (for instance, robot based) connected by automated material-handling system. A difference between FAS e FMS is the type of operations, machining for the latter and assembly for the former. Fully automated FAS systems later came to be termed Flexible Automatic Assembly (FAA) systems.

An example of the thought that CIM and the like could lead to a completely automated factory is expressed by the following phrase from (Teicholz & Orr, 1987):

> *"In CIM the need for paper is eliminated, and so also are most human jobs (...) Why is CIM desirable? Because it reduces the human component of manufacturing and thereby relieves the process of its most expensive and error-prone ingredient."*

The idea that CIM would be the right panacea to implement the automated factory is strengthened by the vision that CIM is more than an integrating technology as it is the case of Mitchell that provides a very broad definition of CIM in (Mitchell, 1991):

> *"CIM is not a product that can be purchased and installed. CIM is a way of thinking about and solving problems. The emphasis is on understanding how to create effective manufacturing enterprises. The determination to apply this understanding must come from personal commitment."*

High levels of automation, integrated information flow, and the widespread application of computers were thought to be the basis for manufacturing companies to succeed. CIM and the development of flexible shop floors were thought of as the panacea to solve the problems the European and American companies were suffering. However, large investments in technology did not prove to be the solution, since the problem was more complex than just a change in the technology of the shop floor. Many large-scale investments in automation came before companies had understood how to implement automation and what its benefits truly were (Hormozi, 2001). One of the problems has also been that the R&D community only focussed on localised problem issues, such as machine flexibility or machine communication, thus losing the holistic perspective. The automation solutions were therefore good at solving only part of the problem, and at a very high cost (Adlemo, Andréasson, Fabian, Gullander, & Lennartson, 1995; Grondahl & Onori, 2000; Onori, 1996, 2002b; Onori,

Langbeck, & Grondahl, 1997). Only CNC, machining centres and surface-mounted automation became cost-effective, primarily because networks and standardisation communities were established at an early stage. In addition, the control and supervision architectures were not very reliable and became obsolete very rapidly (M.G. Mehrabi, A.G. Ulsoy, & Y. Koren, 2000). In a desperate attempt to create solutions that could perform a wide range of tasks fairly well, they never became excellent performers at any given task. Optimised for a given flexibility, they became sub-optimised at individual process level.

Eventually it was understood/realised that the solution was not only technological but also organisational.

Flexibility is the landmark of this period. At the time, agility or system reengineering was not considered so important because it was imagined that systems would be built that were so generic that changes were not necessary or at least very few would be necessary. Generic machines (programmable) turned out to answer the requirements for flexibility, however, in spite of being flexible, shop floors could not cope well with changes (system reengineering) because the concept of systems' evolution (life cycle support) was missing.

2.1.1.4 Lean Manufacturing

The USA in its effort to solve the mentioned problems, and to understand why the Japanese industry was so successful, created the International Motor Vehicle Program (IMVP) at the Massachusetts Institute of Technology (MIT). This program studied thoroughly the automobile industry around the world and produced a report, which is known by the book name "The Machine that Changed the World" (Womack et al., 1990). This study revealed the Japanese production and organisational techniques to the western world, and coined the term Lean Manufacturing. Before this study, western companies thought the Japanese manufacturing success was mainly a cultural question even if, for instance, some academia members were already pinpointing some of the aspects later associated to lean manufacturing (Cammarano, 1997; Svensson, 2001). It was believed that the way Japanese society is organised greatly contributed to the way production and manufacturing business was organised. Although this is partially true, there are many factors associated with the Toyota Production System that can be successfully applied elsewhere, even if requiring some adaptations to cope with the cultural differences.

A lean manufacturing system is one that meets high throughput or service demands with very little inventory, and with minimal waste. "Lean thinking" has consequences for the way production is organised and managed, the way work is done, and in the relationship between manufacturing companies and their suppliers and customers.

The most important idea behind lean manufacturing is **avoiding waste**, *Muda*, which is the Japanese word for waste. *Muda* is any human activity that absorbs resources but creates no value.

Lean organisations claim they are more efficient because they only spend resources in activities that add value. There is, of course, the problem of identifying the value of an activity. An activity which value is not immediately recognised is at risk of being removed, even if this activity proves, later, to be important. Consider the risk of removing innovation just because its value is not immediately recognised. This is why this aspect of "lean thinking" is also one of the most controversial ones. Many authors argue that the goal of reducing waste can be dangerous because the company can become too slim and not able to react to the unexpected events of a turbulent world (Duguay, Landry, & Pasin, 1997; Katayama & Bennet, 1996). (Lewis, 2000), for instance states *"the more successfully any firm applies lean production principles, the less it will engage in general innovative activity"*.

Another aspect of lean manufacturing is the way the production line (shop floor) is organised. Shop floor **workers are organised into teams** with a team leader rather than a foreman, as it occurred in mass production. The workers are polyvalent and able to execute the various tasks assigned to the team. This permits that a worker could easily be substituted by another whenever necessary. The teams are also assigned with some minor repairs and quality control tasks. This increases the final quality of the product (less rework) because they are inspected along its production life cycle, instead of being inspected only at the end of the line.

In clear contrast to the mass production assembly line, teams have the right to stop the assembly line, whenever they think it is necessary, as when repairing it. Furthermore, workers are stimulated to participate with suggestions to improve the process. This continuous improvement (*kaizen* in Japanese[1]) (Imai, 1988) took place in collaboration with industrial engineers who are much lesser than in the mass production case. The continuous improvement strategy can be effective because workers, if properly motivated, can contribute substantially since they are the ones that truly master the processes being taken care of.

The other aspect of lean thinking is **the reduction of inventory**. In lean thinking buffers and warehouses are avoided because they are a kind of *muda*, which is costly. The idea behind this is that an item must only be produced when it is needed (there is an order). This requires a highly synchronised system involving the plant and its suppliers and customers. The philosophy used to coordinate the flow of parts within the supply chain is called Just In Time (JIT) (Chase & Aquilano, 1989; Silver, Pyke, & Peterson, 1998) or *kanban* in Japanese (PPDT, 2002). JIT is a method of production supply and inventory control to guarantee a synchronised flow to make only what is needed with no waste and low inventories through the whole supply chain, i.e., in the marketing channel and in the processes of manufacturers.

JIT is often confused with lean production. However, JIT is only a philosophy, with roots as far back as the 1880s, while lean production is a combination of *Kanban*, *Kaizen* and *Muda*. A visible effect of well-applied JIT is the elimination of warehouses inside the final producer to store product components, which only arrive at the company whenever they are required. Some critics say that JIT

[1] Kaizen is a Japanese word meaning gradual and orderly, continuous improvement. The KAIZEN business strategy involves everyone in an organization working together to make improvements without large capital investments.

implementations are just a clever way of transferring the inventories from the final producer into its suppliers.

At the core of the JIT procedure is *Kanban* control, which uses the levels of buffer inventories in the system to regulate production. When a buffer reaches its preset maximum level, the upstream machine is told to stop producing that part type. This is often implemented by circulating cards, the *kanbans*, between a machine and the downstream buffer. The machine must have a card before it can start an operation. It can then pick raw materials out of its upstream (or input) buffer, perform the operation, attach the card to the finished part, and put it in the downstream (or output) buffer. *Kanban*, therefore, operates in a pull mode and ensures that parts are not made except in response to a demand.

Lean production is a build-to-order system in which the dealer (distributor) is the first step in the JIT system. He starts the process by sending orders to the producer for presold products for delivery to specific customers. Products are pulled by customers out of the factory instead of being pushed by the factory to the customers.

The final important aspect of lean production is the **establishment of a new type of relationship between the company and its suppliers and dealers**. The involvement of suppliers and dealers in the conception of the product, for instance, leads to products better suited for customers, with improved quality, and better suited for manufacturing. Since suppliers and dealers together with the company become used to working together, they create a stronger and more effective structure, in other words a supply chain. This structure becomes more and more a horizontal organisation rather than vertical, as it was the case in mass production. The most important aspects of lean production are summarised in Table 2-5.

❑ Team work ❑ Communication ❑ Elimination of waste - *muda* ❑ Continuous improvement - *kaizen* ❑ Cooperation with customers and suppliers ❑ Just in time (JIT) and kanban	❑ Customers pull the products from the factory ❑ Supply-chain ❑ Customers and suppliers participate in the development of the product ❑ Workers have no autonomy ❑ Market is predictable

Table 2-5 – Lean manufacturing characteristics

Although lean manufacturing principles have been successfully implemented in many American and European companies (Karlsson, 1996; Kochan, 1998), its acceptance is not universal. Many critics refuse to accept that its principles were the right tool to fight the troubles faced by mass production (Berggren, 1992; Spithoven, 2001). However, a large number of people agree that the turbulence experienced by manufacturing companies near the end of the last century can be dealt with by lean manufacturing. Although lean supporters claim that lean production system is universally applicable, most of the successful implementations of lean production have been in the automotive and electronic

sectors, which are high volume with low variety industries. Applications of lean in sectors characterised by highly differentiated products and low volume are rare (James-Moore & Gibbons, 1997). The recent economic difficulties of Nissan and Mazda, which were forced respectively to merge with Renault and Ford is also a sign that the paradigm may not be well suited to all market situations. The main disadvantages of lean production can be found in Table 2-6.

❑ It can be even worse for workers than mass production. Pressures on the managers to reduce waste force downsizing (fewer people, fewer resources, fewer everything) and increase the level of anxiety over the workers for better efficiency (Allen, 1997; Biazzo & Panissolo, 2000; Freyssenet, 1998; Kinnie, Hutchinson, & Purcell, 1997; Kovács & Castillo, 1998; Spithoven, 2001) ❑ Too much slimming down can make the company vulnerable to market turbulence. Innovation can also be slowed down because of a lack of resources (Lewis, 2000)	❑ Difficulties in finding where and when "value" is realised in the manufacturing chain ❑ The efficiency of lean is based on the assumption that Japanese forms of long-term contracting will become an industry practice, which in effect cannot be assumed ❑ The influence of social and political institutions is largely ignored by lean supporters (Lewis, 2000) ❑ Lean production deals badly with variations or reductions in the demand for finished products (Katayama & Bennet, 1996)

Table 2-6 – Lean disadvantages

Since the capacity to produce different products at different production rates is also a landmark of the lean manufacturing paradigm, the control and supervision architecture designed for these shop floors needed to be flexible. In addition, there were also requirements for changes and adaptations because of the continuous improvement philosophy. In spite of the requirements for easy adaptation/changes of the shop floor (reengineering) the control/supervision architecture was designed mainly to be flexible rather than agile. Shop floor reengineering was slow and difficult to perform.

2.1.1.5 Anthropocentric and sociotechnical production

The anthropocentric approach can be considered the European answer to the troubles European companies were facing in the 1970s, and eventually an answer to the spread of lean manufacturing (Berggren, 1992; Higgins, 1995; Kovács & Castillo, 1998; Wobbe & Communities, 1992). This concept was developed in the framework of the FAST – Forecasting and Assessment in Science and Technology, European Union research project in the early 1990s.

The anthropocentric approach intends to move the human from being an object in the production processes to a better job qualification and satisfaction (improved autonomy) by creating organisational and technological structures that support this human integration in contraposition to the previous focus in technology. It is also an answer to the already referred to massive investments in technology of the 1980s, e.g. an alternative to the highly automated factory of the 1980s (Wobbe, 1995). Researchers

from this area often state that Computer Integrated Manufacturing (CIM) concepts, CNC machines and robots, and FMS/FAS systems were introduced using the old *tayloristic* methods of organisation, which contributed to its failure. However, these researchers did not consider other reasons such as market pressures for fast solutions as well as that the promises of high-tech were not realised because basic technologies were not mature, the requirements were not well detailed, and the customer-producer relationship was changing. The reason for missing non-human related reasons is likely due to their sociology-based background that tends to emphasise human related aspects. It can be noticed here the old European tradition of craft industry, which concedes the human as being of great importance in the productive process. This is why some researchers call this approach neocraftism (Womack et al., 1990).

Since its basic principles are based in the **Sociotechnical** approach that started to be developed in the 1950s[2] (Trist, 1963), it can be considered an evolution of that approach. However, the sociotechnical approach was only incremented after 1962 due to the Industrial Democracy project in Norway (Teixeira, 1996). Even though the dissemination of these ideas was not very important among Norwegian or worldwide industries, the same did not happen in Sweden where they had much more acceptance, especially at Volvo, the biggest Swedish car producer, which, in 1974, installed a plant in Kalmar following this approach. Volvo followed this approach because at that time it was difficult to get people to work in the assembly line (Karlsson, 1996).

However, the most remarkable and studied case is the Volvo Uddevalla plant. The basic concept behind this plant was to let the workers build complete cars. This was completely against the *tayloristic/fordistic* approach of making the job as easy as possible. Volvo, to attract workers, announced they wanted car builders instead of assembly workers. Work was organised into result-oriented teams, which implied that the workers that were actually producing the car were in contact with the customer. This had a psychological effect on the customer and the worker. The customer thought his car was being carefully assembled while the worker felt he was making something useful for some identified entity[3]. The industrial culture of Volvo was thus very much oriented towards the humanization of work posts[4] (Muffatto, 1999).

This Swedish model is better known as ***Volvoism*** or **Reflective Production** and has been deeply studied by (Berggren, 1993; Ellegard et al., 1992; Engstrom, 1994, 1998; Engstrom, Jonsson, & Medbo, 1996; Niepce & Molleman, 1996).

Despite the apparent success of *Volvoism* in Sweden it has not spread over to other countries. To make things even worse, the Kalmar and Uddevalla plants were closed in 1994 and 1993 respectively. However, Uddevalla was reopened in 1999 as a joint venture between Volvo and TRW. According to its supporters, *Volvoism* was just stopped when the evolutionary process would have taken it to a new peak of development (Muffatto, 1999), while for the critics it was just that the model could not

[2] The sociotechnical first ideas were developed in the Tavistock Institute of human relations in England.
[3] The influence of craft is also evident here
[4] This is changing as Volvo is now moving over to Ford

produce cars effectively and economically. The picture, however, is not so simple. If big and complex products are being produced in small volumes, it can be a great concept. If, on the other hand, big and complex products are being produced in big volumes, it can be very poor.

The diffusion of *Volvoism* and the Anthropocentric approaches in Europe were restrained by the importance of lean manufacturing or, in other words, by the strong case which had been made by lean supporters. "Lean thinking" was quite popular among managers and entrepreneurs, which avoided the emergence of the more human oriented approaches.

Table 2-7 indicates the most important aspects of the anthropocentric approach, adapted from (Kovács & Castillo, 1998).

The concept of Balanced Automation Systems (BAS) (Camarinha-Matos & Afsarmanesh, 1996; Camarinha-Matos, Afsarmanesh, & Erbe, 2000; Camarinha-Matos, 1995; Camarinha-Matos, Afsarmanesh, & Marik, 1998) was introduced in the mid 1990s by researchers from the technology domain as a direct result of the CIMIS.net and FlexSys collaborative projects between the European Union and Latin America, launched by an exploratory ESPRIT basic research activity in 1993. BAS addresses an industrial environment with appropriate level of automation. In contradiction to both pure human based and the total automation environments, its emphasis is on the combination of Anthropocentric, Technocentric, and Economic approaches (Camarinha-Matos & Afsarmanesh, 1996). The goal is the achievement of a right balance between automated and manual components and the hybrid solutions. It was already considered that BAS should cover the reengineering aspects of shop floors as well as life cycle support.

❑ Increase the quality of products by continuous improvement and flexibility of the production system ❑ Increase the working conditions of workers using new organisational forms and technologies ❑ Human resources are valorised by qualifications, training, and involvement with the production	❑ Technology is specifically adapted to human needs and not vice versa ❑ Workers are organised in teams using the sociotechnical principles ❑ Decentralisation and fewer hierarchies. Responsibilities are spread through the teams ❑ Polivalence of competencies

Table 2-7 – Characteristics of the anthropocentric approach

In human centred approaches the requirements for control and supervision architectures with support for shop floor changes are more important than the requirements for flexibility. This happens because humans are inherently flexible and autonomous, which makes the control architecture less complex because humans can make many of the decisions in a flexible shop floor. On the other hand, adapting humans to technology (human centred approach) implies more changes, which makes the architecture more demanding in terms of shop floor reengineering. Another interesting work about the

development of technical systems adapted to workers is (Soares, 1998), which deals with the conceptual modelling for technical, social, and organisational development of integrated manufacturing systems.

To conclude, it became apparent at the beginning of the 1990s that there was no general solution to different production scenarios. Small complex products in very high volumes, with many variants but low individual volumes, big products in big volumes, coexist, and therefore increased production systems complexity. One reason why everyone was doing things wrong in Europe is because companies were still focussed on finding a single panacea to diversified problems, and very little attention was being placed on what was happening to the products themselves. SONY, for instance, on the other hand, was further ahead than Toyota in some respects, because they had already coupled the product design to assembly system specification (Erixon, 1998). All of this changed when the manufacturing community began to talk about mass customisation, changing the focus from production system to produced object.

2.1.1.6 Mass Customisation

Besides the existence of lean manufacturing and human centred approaches, manufacturing companies were not coping well with the market and social situation of the 1990s, and the omnipresent globalisation of the markets that was starting to emerge. Consumers wanted, and still want, more and more exclusive products at lower prices while keeping or increasing the quality. They are, therefore, no longer lumped in a huge homogeneous market, but are individuals. Companies, on the other hand, realise that if they reduce the product life cycles and fragment the demand of their products they can obtain advantages over their competitors that are forced to follow them[5]. This trend is not confined to a specific area, and can be observed in the automotive, electronics and telecommunications, beverages, information technology, fast food, personnel care sectors, apparel, and garment and in many areas of the services sector (Pine II, 1993).

The concept of providing personalised products at a reasonable price is called **mass customisation**, which was popularised by Joseph Pine II at MIT. To Pine II mass customisation is a new way of doing business and at its core is the fast increase in variety and customisation of products, without increasing costs, while at its limit it is the mass production of individually customised goods and services. The identification and fulfilment of the "wants and needs" of individual customers without sacrificing efficiency, effectiveness, and low costs are thus basic requirements and challenges. Mass customisation helps to understand why product life cycles are decreasing, why product development and the development of the production system must also decrease their life cycles, why companies need to reengineer their processes to become more agile, and why networked organisations are

[5] This is what happens with Zara, the apparel seller. They take advantage of their capability to have high rotation of products (short life cycle) and highly diversified products. Their competitors are always forced to react to their products, and when they react Zara is already introducing a new set of products.

emerging. The opposite requirements that characterise this paradigm (mass and craft) are an important source of the turbulence manufacturing companies are facing. Other sources relate to social and political aspects.

To be able to cope with the new paradigm companies need to change their focus from producing standardised products for an homogeneous market to the unstable and unpredictable world of heterogeneous and fragmented markets. They need to produce, market, and deliver affordable goods with enough variety and customisation that nearly everyone finds exactly what they want. When more and more customers start to have customised products the markets become more and more fragmented. Fragmented markets on the other hand will imply a heterogeneous market with even more demanding customers asking for more specialised products. In turn, companies have to research for innovative products and then they need to research and find innovative process technology to be able to respond to product variety and customisation. With the products in the market the loop is closed and the process starts again.

A vital aspect in this loop is thus **innovation in product and process technology**, i.e., the way the products are produced (Pine II, 1993). The need for flexible shop floors, able to cope with variety in products and in demand, as well as with unpredictable situations is thus a mandatory requirement to support mass customisation. The process should be as decoupled as possible from the ever-changing flow of products.

Another aspect is the low cost, which must still be a main goal. It can be achieved either through economies of scale, i.e., cost is reduced by increasing the number of units sold, or by economies of scope. In this case costs are lowered by using a single process to produce a greater variety of products. Companies often achieve both by applying economies of scale on standard products that can then be combined in different ways to produce a great variety of final products. This is called product modularisation and is one of the main factors for succeeding with mass customisation (Erixon, 1998). If properly designed, it is possible to create a huge variety of products that are based on standardised components that can be mass-produced. The final customised product is thus assembled from a set of mass produced components.

One important alteration under the mass customisation *world* is the evolution of the supply chain from a very static entity to a very dynamic and highly interactive one. Due to the market fragmentation and the rapid change in technology, more actors are now involved and a deeper cooperation is needed within and between them to create and sell the customised goods. No company can have all the required skills and knowledge. On the other hand, the proliferation of products may require an increased number of suppliers (producers of components that will participate in the final customised product). This aspect is highlighted by the need to seek external partners to be able to provide a more completed product (services, information, and logistics, for instance). To cope with this situation the supply chain must now be dynamic to accommodate the easy integration and departure of members.

The **fission** (fragmentation of companies) versus **fusion** (fewer players in some businesses) problem, which is seen nowadays, is related to this situation. Fission occurs because of market

fragmentation and the need for companies to focus on their core competencies. Fusion, a contradictory trend, happens because with the globalisation and the demanding business requirements of mass customisation, only powerful companies operating at a global level can have the financial resources to be successful.

A final aspect of the mass customisation trend is that products are including more and more services. Products need to be customised to the desires of special markets segments. This association of services to the products is becoming more and more important.

2.1.1.7 Agile manufacturing

Changing and uncertainty of markets, society, or technology are so omnipresent that a paradigm that can handle turbulence is demanded. This manufacturing paradigm is called agile manufacturing and the term was coined by Nagel and Dove in the famous report "21st Century Manufacturing Enterprise Strategy", they developed at the Iacocca institute (Nagel & Dove, 1992)[6].

Agile manufacturing is a step forward with respect to Anthropocentric and Lean Manufacturing because those paradigms can only satisfy in a controlled environment, while agile manufacturing deals better with things that cannot be controlled (Maskell, 2001). Agility is the ability to thrive and prosper in an environment of constant and unpredictable change (Goldman, Nagel, & Preiss, 1995), and is required not only to accommodate change but also uncertainty. Agility is more than being flexible. The flexible manufacturing systems already developed in the 1980s cannot cope with uncertainty. Even if they can produce a variety of products and accommodate some changes on demand, they can only do it if these variations are predictable. The goal of manufacturing systems is now to provide high quality products instantaneously in response to demand (Davidow & Malone, 1993).

Agility covers different areas of manufacturing, from management to shop floor. It is a top down enterprise wide effort. The agile manufacturing company needs to integrate design, engineering, and manufacturing with marketing and sales, which can only be achieved with information and communication technology (ICT). If the requirements imposed for agile manufacturing had been needed before the advent and dissemination of ICT, it would not have been possible to implement it effectively, because ICT is one key technology enabler. The successful implementation of agile manufacturing requires the following points, which were adapted from (Barata & Camarinha-Matos, 2000; Barata et al., 2001; Camarinha-Matos & Barata, 2001; Hormozi, 2001; Leitão, Barata, Camarinha-Matos, & Boissier, 2001; Maskell, 2001; Onori, 2002b; Vernadat, 1999):

1. **Political decisions** – It is important to create, for instance, regulations to help cooperation and innovation.

[6] This report was prepared in response to a request from the USA Congress to identify the requirements for the USA industry to return to global manufacturing competitiveness.

2. **Business cooperation** – With the increase of short term relationships between suppliers and customers, it is natural that companies should be able to diversify cooperative relationships as much as possible. The need for agile business structures to facilitate the creation of dynamic networks of enterprises is mandatory. The ability to create virtual organisations, which are opportunistic alliances of core competencies across several firms to provide focused services and products, is thus the limit in terms of cooperation.
3. **Customer focus** – Although this is related to the previous point it was decided to separate it to stress how important it is to create a philosophy to focus the company on the customer. Solutions, which involve product and services, must be sold to customers to increase the product value and the awareness of the customer to pay more for that product. The addition of value to the product may even be obtained by cooperation with other companies, which is achieved by the previous point.
4. **Information technology** – The requirements needed to keep a close relationship with customers and suppliers can only be effective with good computational support. Production systems need also computational support to create flexible and agile shop floors. Shop floor equipment such as robots, CNC, and transporters, all require computers. Finally ICT is also fundamental to support the fast design of products. Concurrent systems to involve the customer, suppliers, and the manufacturing people in the product life cycle can only be achieved using computers.
5. **Processes reengineering** of the company – This involves not only identifying what processes should exist but also redesigning them. Reorganisation must be a routine. In addition, more than one organisational structure might be needed at the same time.
6. **New work organisation** – The tayloristic work organisation of division of labour has to be replaced by an organisation based on teams and cooperation. On the other hand workers need to be more skilled because the company is more complex and they can intervene more and have more autonomy. The workers need to be highly educated and trained. The anthropocentric work organisation approach seems to be well indicated for agile manufacturing.
7. **New agile shop floor paradigms** – The current approaches used in the shop floor like traditional FMS/FAS are not well indicated to deal with turbulence. One important limitation of those strategies is their inability to deal with the evolution of the production system life cycle. A methodology that could integrate the various aspects related to the life cycle of the production systems is thus very important for an agile shop floor. This aspect is much wider than the problem of flexibility and the addition of computer equipment. It involves a careful study of the actors, the processes, and the areas that are involved in the production system and then the development of a methodology that helps the integration of these areas and supports their life cycle evolution.

8. **Willingness to change** – This aspect must be present in all the previous points. The agile manufacturing can only be successful if all its actors are constantly monitoring the situation around them and willing and prepared to react whenever it is necessary. This might be achieved by a system of incentives. The company needs to have a clearly defined vision of where the company is going to, and how these objectives will be met. Being responsive is the key word for success. This applies to supportive R&D and academia as well.

2.1.1.8 Business Process Reengineering – BPR

Although the business process reengineering (BPR) is not shown in Figure 2-2, because it may be deemed not so relevant as the other paradigms indicated there. Nevertheless, it was decided to include it here because it is a quite relevant management philosophy, which started in 1993, as an answer to the manufacturing companies troubles of the 1980s. BPR is an answer at management level to the problems already identified when mass customisation and agile manufacturing were discussed. The book "Reengineering the Corporation" (Hammer & Champy, 1995) is probably the most well known reference. BPR is mainly an organisational philosophy, thought of for the higher managements aspects of the companies. It was not thought of for the shop floor (lower level). In fact, although its supporters claim that it should be applied to all company levels, this was not the case and most of the reengineering aspects were applied to the higher-level organisational aspects only. BPR specifies "*how and when to redesign old processes to eliminate waste, ineffectiveness, and lack of added value*"[7] (Victor & Boynton, 1998). However, the official definition by Michael Hammer in (Hammer & Stanton, 1995) is "*the fundamental rethinking and radical redesign of business processes to bring about dramatic improvements in performance*".

The goal of BPR is to move from an old type of organisation, which is based on a functional structure, to an organisation based on business processes that are able to add value to the product. This implies a tremendous effort of changing the way people are used to thinking in organisations, and this change is so radical that in many companies reengineering was more unsuccessful than successful. Even though the concept is easily understood, its implementation is much more difficult because, among other reasons, it is not always easy to find precisely which processes create/add value. This was already a problem in lean manufacturing. By only focusing on reducing the non added value processes, companies could fall into a disadvantageous situation in the medium to long term. In addition to reengineering it is important that the full process should be completely understood in order for it to be analysed and optimised. However, in many situations the processes can be so complex that became almost impossible to understand the full picture, which conducts to the failure of reengineering (Victor & Boynton, 1998).

In manufacturing, company business processes may include, for instance, the product design, the assembly of the product, or even the process of getting subcomponents from suppliers. A successful

[7] Notice the similarity with Lean Manufacturing in what concerns the aspects of waste.

BPR implementation of a manufacturing component thus requires a profound understanding of how the product is designed, how it is created, and how it is followed after it left the company. These activities are no more than product life cycle activities, which if properly understood can be very useful when organised as business processes.

2.1.1.9 Virtual Organisations

As it was mentioned under agile manufacturing, one form of creating agile business cooperation is by implementing opportunistic alliances of cooperating companies. This structure is known as **virtual organisations (VO)**. There are many definitions, some of them even contradictory to VO. For Davidow and Malone it is a structure that "*requires taking a sophisticated information network that gathers data on markets and customer needs, combining it with the newest design methods and computer-integrated production processes, and then operating this system with an integrated network that includes not only highly skilled employees of the company but also suppliers, distributors, retailers, and even consumers*" (Davidow & Malone, 1993). In the book "Agile Competitors and Virtual Organisations", Goldman, Nagel and Preiss define it as "*an opportunistic alliance of core competencies distributed among a number of distinct operating entities within a single large company or among a group of independent companies*" (Goldman et al., 1995). To Goranson VOs "*are opportunistic aggregations of smaller units that come together and act as though they were a larger, long lived enterprise. The virtual here is meant to convey that many of the advantages of a large enterprise are synthesised by its members*" (Goranson, 1999). To John Byrne "*Virtual Corporation is a temporary network of independent companies - suppliers, customers, even rivals - linked by information technology (IT) to share skills, costs and access to one another's markets. It will have neither central office nor organisation chart. It will have no hierarchy, no vertical integration*" (Byrne, Brandt, & Port, 1993). However, the definition by Camarinha-Matos and Afsarmanesh combines some of the ideas presented in these previous definitions and is a suitable synthesis (Camarinha-Matos & Afsarmanesh, 1999b):

> *"A virtual enterprise (VE) is a temporary alliance of enterprises that come together to share skills or core competencies and resources in order to better respond to business opportunities, and whose cooperation is supported by computer networks".*

The keywords are networking and cooperation. According to Camarinha-Matos and Afarmanesh in (Camarinha-Matos & Afsarmanesh, 1999b) *"there is a clear trend for the manufacturing process not to be carried on by a single enterprise any more, rather every enterprise is just one node that adds some value (a step in the manufacturing chain) to the entire production cycle".*

It is interesting to compare virtual organisations with supply chain. Both concepts have in common the idea of supporting the product from raw materials to a finalised customised product and associated services in the final customer's hand. The supply chain is a stable structure and focused on logistics

and related business information[8], while virtual organisations are unstable (temporarily), cover all the areas required to produce customised products, and are focused on collaboration.

Even if the idea of cooperating in business is not new it has been restrained because of technological reasons. In effect cooperation can only be achieved by fast, reliable, and updated information exchange. Only with the dissemination of information and communication technology (ICT) is it now possible to extend and further develop business cooperation to new levels, which is the objective of virtual enterprises. Only within a networked world composed of pervasive powerful computers and programs is it possible to implement this paradigm. This is why it is starting to be a reality only now. To implement this paradigm, however, the advances in ICT are not sufficient yet. Further research is required.

As would be expected, this new way of doing business challenges the way production systems are organised. Much research has started and much more is required (Camarinha-Matos, 2002a; Camarinha-Matos & Afsarmanesh, 1999a; Camarinha-Matos et al., 2001). Virtual organisations should not be considered an answer to agile manufacturing but rather an answer to the problem imposed by mass customisation. As they share many of the concerns of agile manufacturing, which in effect were imposed by mass customisation, they are an implementation of agile manufacturing in terms of business organisation and cooperation.

The area of VEs/VOs is particularly active in Europe, not only in terms of research and development, but also in terms of the emergence of various forms of enterprise networking and advanced clustering at regional level. This "movement" is not only consistent with the process of European integration, which represents a push towards a "culture of cooperation", but also with the very nature of the European business landscape that is mostly based on small and medium size enterprises (SME) that have to join efforts in order to be competitive in open and turbulent market scenarios. Note that important activities can also be identified in other regions such as Australia, Brazil, Mexico, and Canada, in addition to the USA.

Much work is currently being done in the organisational aspects of Virtual Enterprises, namely on how to create IT infrastructures to support the VE life cycle. However, much less work is being done on how to connect companies with production systems to VEs/VOs. This mainly happens because traditional companies with production systems are not aware of this trend and the impact it will have on them. Nevertheless, only when these more traditional producing companies are successfully integrated into networks of enterprises (VEs/VOs) will it be possible to speak about successful VEs/VOs

The problems associated with integrating a production system into the VE/VO world are significantly different from integrating it in traditional companies because of the following reasons (Camarinha-Matos, 2002b):

[8] In most situations there is a company (the final producer) which is the main node and it is it that effectively controls the flow of information and dictates the rules of the game. This is very common in the automotive industry.

- The production system must be agile in order for it to be easily adapted whenever the company moves from one consortium to another (business opportunities).
- Supervision mechanisms are required to support a controlled "intrusion" of another company partner.
- Simulation models and tools are required to verify if the consortium being created answers the requirements imposed by the business opportunity that generated it.
- It is necessary to structure the actual processes behind the VE/VO business opportunity.
- Legacy systems must be effectively integrated.
- The integration of different enterprise cultures.

2.1.2 External Factors of the Manufacturing Business Paradigms

In this section some enabler and barrier factors of the manufacturing business paradigms evolution are discussed. The factors are shown in a time line (as in the case of the business paradigms), and compared to the evolution of the business paradigms.

2.1.2.1 Business environment

Figure 2-3 shows some factors associated with business conditions. All these factors have been already discussed in the previous section, when the various business paradigms were discussed. However, in this figure, a global view of the evolution along the period is possible. Four relevant aspects can be highlighted: 1) an evolution towards market fragmentation (barrier), 2) more ICT technology (enabler), 3) a business environment characterised by turbulence (barrier), and 4) trend for customer focused business (barrier).

In the period of craft industry the market was constituted by a fixed group of people with enough money to go to craft producers and ask them to produce what they wanted. In the beginning of mass production there was no distinction between customers and so the market was massified. With the increased sophistication of customers, manufacturing companies reacted by creating segments of customers to be addressed separately, which led to niche markets. This is what happens in mass customisation where a company addresses the individual customer. The tendency shall move to a situation where an individual is a source for various markets, which is even more demanding for companies. However, this claim should be addressed very carefully. In fact, in a world of variety, the Nokia 3310 sold in 10 million copies, and the Nokia 3330 sold 100 million, the Land Rover still sells very well, and the SONY Freestyle is a huge seller. If very high quality products can be provided with a very distinct functionality, they will sell without variants. However, what will customers want in 10 years' time? Will they want 10 millions of one product today and 20,000 of 50 variants tomorrow? Hopefully if the production system can adapt to changing markets, there will not be any problem, whatever the future customers trends might be.

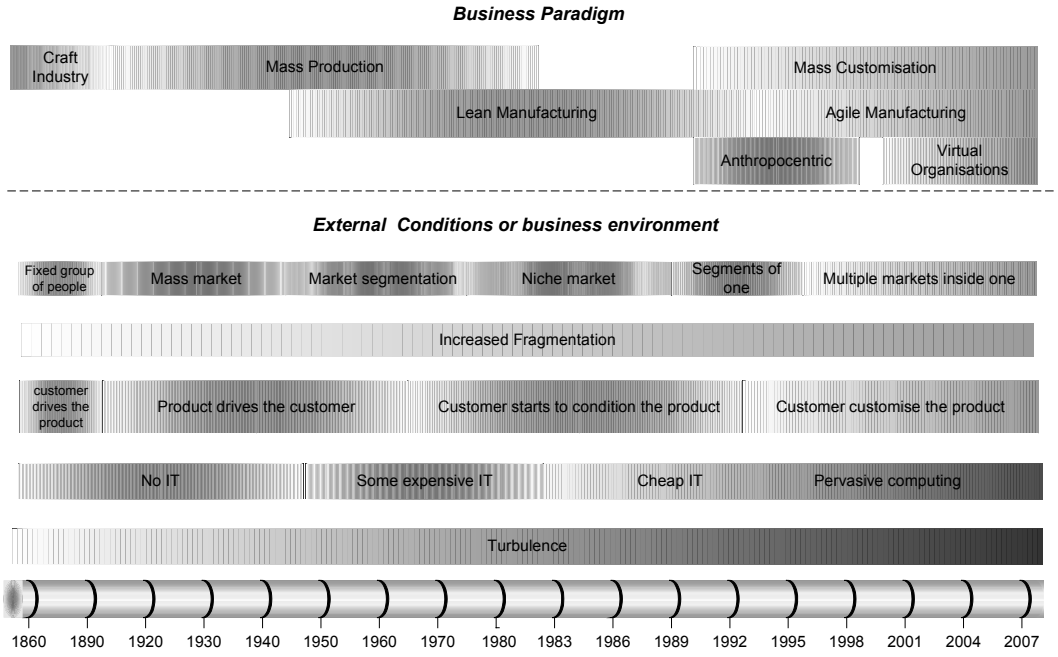

Figure 2-3 – External conditions or business environment

In the craft industry period the customer decided how it would like his/her product, participating in this way in the product design. With the advent of mass production this situation was changed by suppressing the intervention of customers in the process of design and producing the product. In lean production this situation was changed again since customers could influence, even if indirectly, product design and, therefore more adapted products to the user needs were possible. At the end of mass customisation there is a return to craft industry principles in terms of customer participation in the design of its product. Figure 2-3 also indicates the increasing availability of ICT technology and market turbulence.

2.1.2.2 Product

Figure 2-4 shows how the factors associated with the product have evolved along the period covering the end of the nineteenth century to the beginning of the twentieth first century. The most important considered factors are: 1) customisation, 2) variety, 3) complexity, 4) quality, and 5) product life span.

The "degree of customisation" is also related to the inclusion of services in the product. The more the degree of customisation, the bigger the number of services included. During the craft period it was possible to configure the product without any ICT infrastructure because the customer participated personally in the product design. With the advent of mass production and the customer away from the product creation process, products became unconfigured. On the other hand, with the increasing use of ICT technology and the requirements for more customer intervention in the product life cycle, a tendency towards customising the products became noticeable, and will be more and more a fact in the period of agile manufacturing and virtual organisations.

The "variety" measures the number of available models within a family of products and not if they are customised. During the craft and mass production periods the variety of products was low, even though for different reasons. In the craft period the variety was low because they were 'one of a kind' products, individual products without variants. In the mass production period the variety was kept low to optimise costs. Today, with the increased sophistication of customers and market segmentation, manufacturing companies need to start thinking about increasing their products' variety. Already in the 1920s General Motors started to offer various car models in opposition to the strategy followed by Ford. The need for more diversified products is growing, and the limit of this tendency could be a return to the craft era and its 'one of a kind' products. Although the 'a product for each customer' is becoming true, these products are configurations of base components instead of being completely new products – product modularisation.

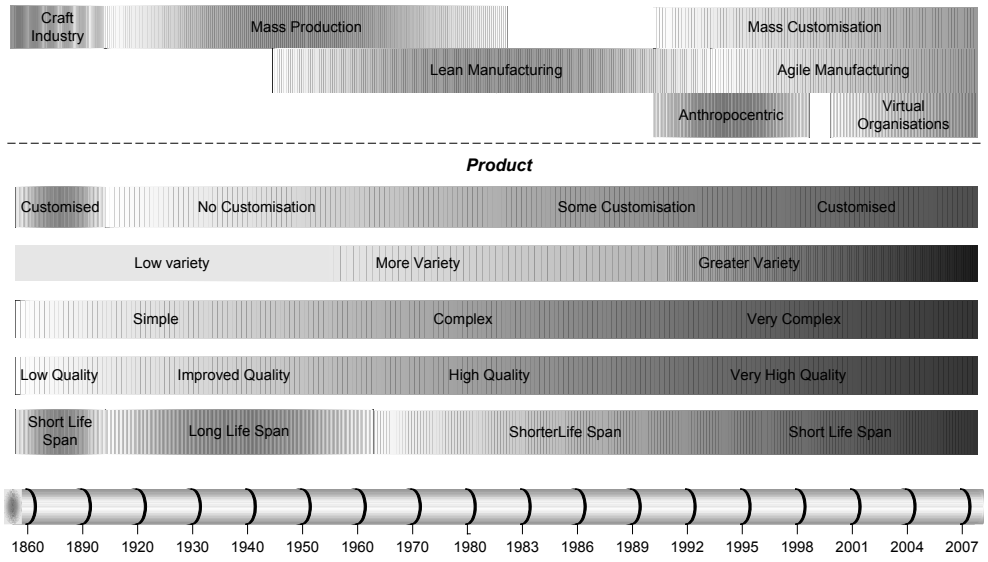

Figure 2-4 – Product conditions

The "complexity" aspect indicates the evolution of product complexity. With the advances in technology products are becoming more and more complex and integrate, in the same product, different types of technologies, which make the design of their production systems more and more challenging. Already in 1998 Onori in (Onori, Arnstrom, & Erixon, 1998) stated that "by combining a modular design with product planning, the product development process and the planning of the corresponding production system changes is greatly simplified. Basically, it allows for the production system to be developed in a stepwise manner. Furthermore, the entire assortment of products may be divided into independent modules that are manufactured in equally independent *assembly module workshops*. The traditional assembly line, in which all parts are assembled along a line, no longer exists". Another source of complexity is the integration of services in the product.

The "quality" aspect measures the quality of the products and there is a linear tendency from the craft industry to the agile era for better quality products. This happens mainly for two reasons. The

first is technological and the second is more social. With advances in technology it is becoming possible to create improved processes and philosophies to produce better products. On the other hand, more sophisticated customers impose on manufacturing companies new requirements in quality. The introduction of the lean manufacturing was an important mark in terms of improved quality.

Finally, the "product life span" indicates for how long products are produced. In craft times, products had a very short life span because they were unique. Mass customisation, on the other hand, changed this because of costs and optimisation. The cost per unit could only be kept low if many products were produced (economy of scale), which is only achieved if they were produced for a long time. However, requirements for more diversified products (product exclusivity) and increased competition imposed a tendency towards shorter presence in the market and consequently less time being produced. This becomes possible through the possibility of creating new flexible/agile production systems, which are cost effective without needing to produce big series of the same product. The limit is life cycles as short as the ones in craft production.

2.1.2.3 Customers

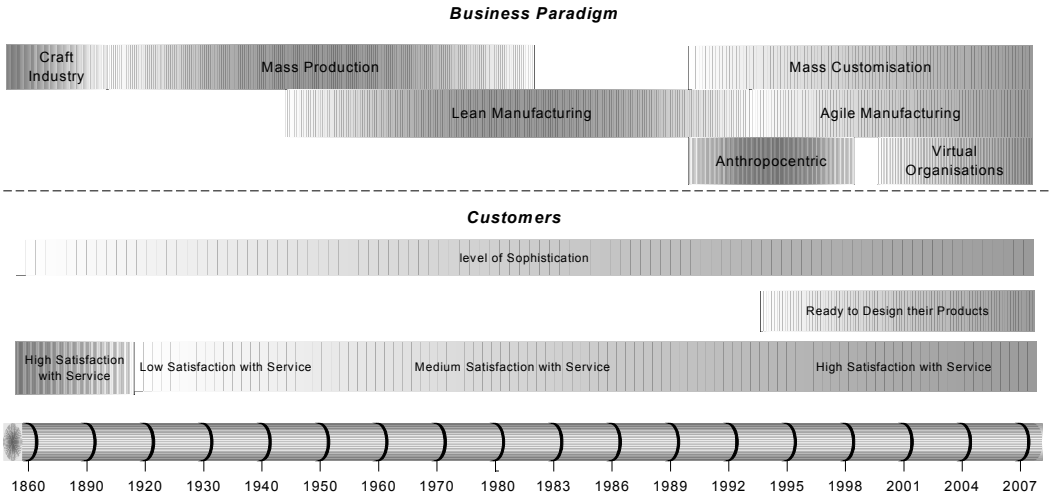

Figure 2-5 – Customers

Figure 2-5 shows the evolution of some aspects related to the customers, which are the reason why manufacturing companies exist. Since societies are becoming more educated and wealthier, at least in some parts of the world, customers are increasingly becoming more sophisticated. The level of education, in addition to ICT infrastructures, also contributes to their willingness to participate in the design of their unique products. Another important remark about customers is their satisfaction level with the service they receive from products. In fact this satisfaction level was high in the craft period because customers received exclusive attention or personalised service. With the advent of mass production this was completely altered and no personalised service in the great days of mass

production was provided to the customers. Then, gradually, as companies began to provide more and more services associated with the product, they became more satisfied (Hormozi, 2001).

The reason why customers are becoming more satisfied can also be explained by psychology. There are several motivating factors for humans (Maslow, 1943). Survival, for instance, is the highest factor. As societies become more comfortable and there is no fear of survival, other motivators take over, namely the awareness for better products.

2.1.2.4 Workers

Figure 2-6 shows the evolution of the factors related to workers. The most important factors are: 1) skills, 2) work satisfaction, and 3) work organisation.

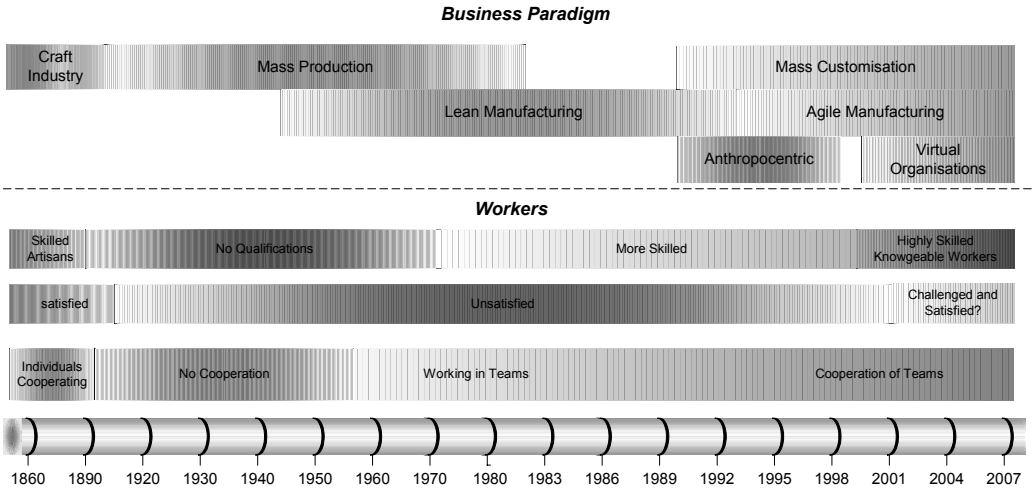

Figure 2-6 – Workers

The skills required by the workers at manufacturing companies have evolved differently along the considered period. During the craft period the workers needed to be highly skilled artisans with broad coverage of specialities. They were polyvalent because they could cover different domains[9].

With the advent of mass production and the *tayloristic* approach of work division, the workers did not need to be skilled anymore. The assembly was divided into very simple basic operations, which were then mechanically executed by the worker. While the craft worker put some creativity in his work, the mass producer worker was just an executor. Charles Chaplin in his Charlot character demonstrated the repetitive aspect of mass production in the famous movie "Modern Times". A consequence of this work division was the proliferation of two types of workers: the ones executing the repetitive operations and the others that were thinking of how the process should be organised and optimised. The latter type of workers is known as "blue collar workers". This division, in time led to a

[9] In the automotive sector, for instance, an artisan was able to participate in most of the activities required to produce an automobile.

stricter separation between the manufacturing and engineering activities, which affected the way the process could be improved. Another consequence was bigger hierarchy levels.

The lean manufacturing paradigm tried to change this situation because it believes that the process can only be continuously improved by the involvement of shop floor workers. Furthermore, as the success of lean was also dependent on an effective quality control spread along the production line, only with better-qualified workers was it possible to reach those objectives.

With the advent of more complex products, more complex production processes, the gradual introduction of more and more services into products, the dissemination of ICT, and new ways of organising the production, new kinds of workers are needed. Workers doing repetitive and boring tasks are gradually disappearing with the introduction of automation. On the other hand, the number of workers involved in the product support services are gradually increasing. The consequence of this fact is the gradual transition to highly skilled workers in ICT technologies. Even those tasks that are still needed and do not require college education, necessitate highly skilled people and with a polyvalent attitude. It is not possible to cope with a distributed manufacturing environment with narrowly trained people.

During craft production the artisans cooperated in a loose manner. They were individuals that could do their job without tight cooperation, which was only required in some specific point of the production process. This mainly happened because they were able to execute most of the operations.

Mass production completely changed this because the work division, which is a characteristic of this paradigm, did not require any cooperation among the workers. Each operator knew exactly what he/she had to do.

This situation was kept until the ideas of organising the workers in teams arose, which is a characteristic of the sociotechnical organisation and lean manufacturing. It was realised that workers can be much more effective if they are organised in teams and that the members of the team cooperate.

With the advent of the concepts of agile manufacturing and distributed manufacturing (virtual organisations) an adaptation is required to put the teams of the different organisations in close cooperation. This requirement is far more demanding because the necessity to cooperate among entities that can have a completely different corporate culture becomes real, which was not the case in lean manufacturing. If the idea of virtual enterprise is taken to the limit where a company is a virtual organisation composed of individuals, then the manufacturing workers are a team of individuals tightly connected (cooperating).

The final factor is about satisfaction. Repetitive operations are boring and produce unmotivated and dissatisfied workers, which is why mass production workers were unsatisfied. Once again this is also related to the Maslow theory (Maslow, 1943). Workers were not dissatisfied as long as this was their means of survival. Only when more and more people were living at a higher standard did they start to object.

The satisfaction level of lean manufacturing workers did not increase as it could be expected by an organisational structure where they could participate more. The reason was that although they were

participating more, the pressure put on them by managers to reduce waste was so high that in the end they were even more dissatisfied than mass production workers. In the new paradigms it can be argued that workers are more satisfied because they have less repetitive operations and they are more skilled. Although this is true, there is still a pressure due to the increasing competition between companies and the globalisation. However, the fundamentalist dogmas of waste reduction from the lean manufacturing are being overtaken and it can be hoped that in the future there will be a tendency towards more satisfied workers. This is of course an optimistic scenario.

2.1.2.5 Supply Chain

Figure 2-7 shows how manufacturing companies have dealt with their suppliers and customers. The typical mass production company in the initial phase of this paradigm was characterised by a vertical organisation. In this type of company the product and its components are produced within the company. The company controls almost every phase of the production process, from raw materials down to customers. The Ford Company, for instance, produced almost everything that was required for the Ford T model production in its headquarters (from tires to engines). This situation was even extended to the limit of Ford owning fields of rubber trees in the Amazonia.

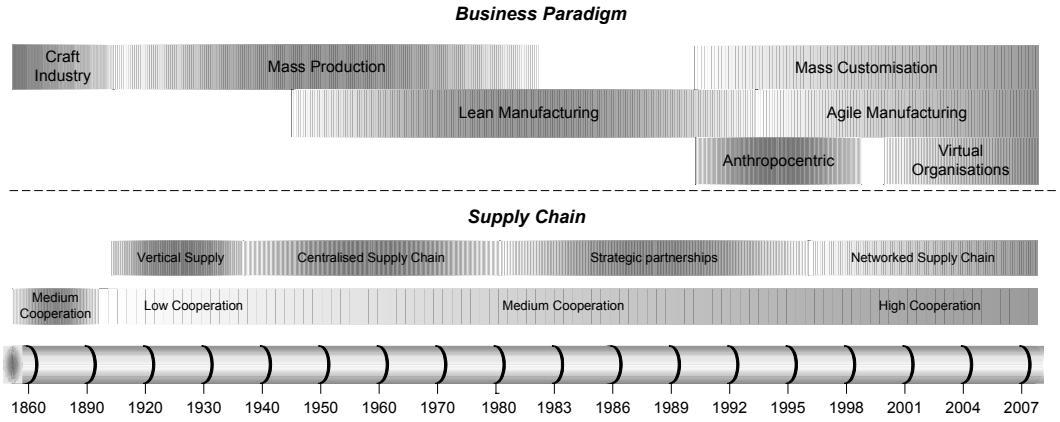

Figure 2-7 – Supply chain

As companies grew, and products became more complex, this structure started to be unmanageable. It was then decided that it was better to decentralise (outsource) some aspects of the product life cycle. In a centralised supply chain the company has a number of suppliers from where it buys the components or raw materials it requires to make the product. However, the cooperation degree is low and the company still commands the relationship. As it was pointed out in (Womack et al., 1990) this type of relationship is a barrier for innovation in products and production process. Companies then start to form strategic partnerships that increase the degree of cooperation between the involved entities. The company that produces the final product is still leading the process but now the suppliers can participate in the product definition and production. However the strategic partnership was too static to cope with the requirements of manufacturing in the 1990s. Companies needed to rethink the

way they cooperated in such a way that active and dynamic cooperation between equals (at least theoretically) became prevalent.

The degree of cooperation in the craft period was medium because companies depended on the others for supplying, creating in this way a network of companies. This is in line with the industrial districts that have been developed in Europe (Piore & Sabel, 1984). The cooperation was not bigger for technological reasons, namely the unavailability of ICT technologies to support the cooperation process. Mass production was disruptive in terms of cooperation and, from then on, there is an increasing tendency for more and more cooperation. The external business conditions and the availability of ICT technologies have greatly contributed to this.

2.1.2.6 Manufacturing software

Figure 2-8 shows the major forces and trends related to manufacturing software. The intention is to present how the various manufacturing areas have been supported by computers.

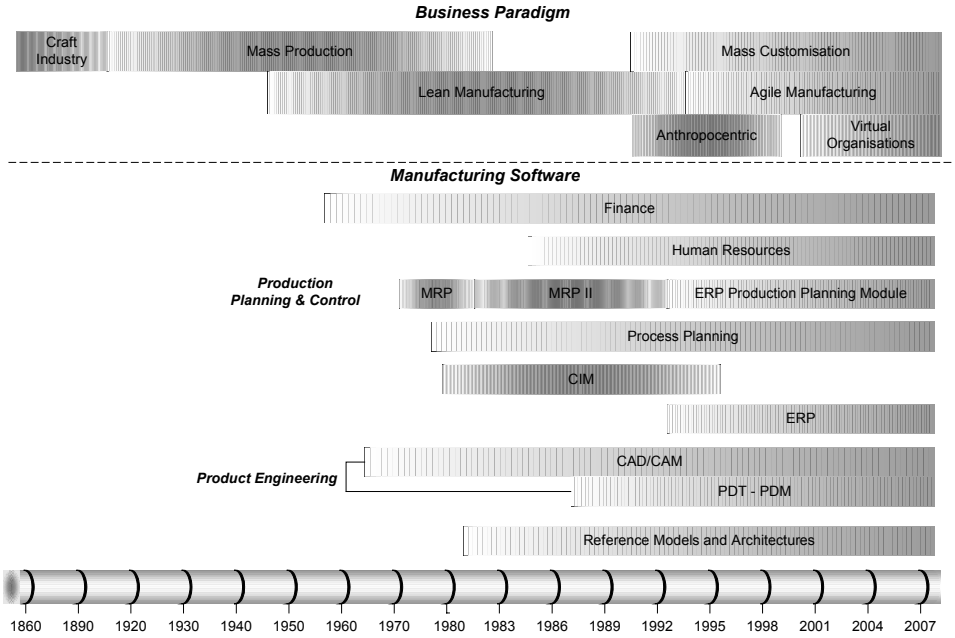

Figure 2-8 – Manufacturing software

In the early days, most computer-based tools were helping the financial area: accounting and payroll applications. The situation has changed quite a lot and it is now possible to find computer-based tools in all areas, from finance to marketing, including production and product design.

The financial area was the first to receive computer support. Accounting information systems recorded transactions and reported on the financial condition of a business and the results of its operations (financial accounting). They also still produce projections, budgets, and cost analyses (management accounting). Specific support is provided for order processing, inventory control,

accounts receivable, accounts payable, and payroll. Accounting programs have existed in manufacturing companies since the 1960s. However, at that time, only big companies could afford to buy the existing big and expensive mainframes. The use of computers to support these functions has been increasingly keeping up with the pace of technological evolution.

The use of programs to manage human resources is much more recent. Human resource information systems support the recruitment, placement, evaluation, compensation, and development of the workers of an organisation.

An important application of computers has been in the planning and control of production. In the 1960s control was made through the inventory (Inventory Control). With the dissemination of computers during the seventies, the situation changed and the focus turned to the planning of the required materials (Kumar & Meade, 2002). It then became possible to automate many of the manual practices employed in manufacturing for acquiring and tracking of materials. The software tools developed to do this were coined Material Requirements Planning (**MRP**) and were designed to translate the master schedule, which was based on sales forecast, into time-phased net requirements for the sub-assemblies, components, and raw materials. The material was planned for according to the forecast timing, without considering the actual capacity (infinite capacity).

In the eighties the focus was changed from materials to resources. The developed software tools to handle the new concept were coined **MRP II** - Manufacturing Resources Planning. The production schedule now considers the capacity, and the timing of material acquisition activities. Since the MRP concepts are based on a stable world, namely the capacity to precisely predict sales, when market conditions changed and it became impossible to assure stable product demands, the MRP started to fail. Furthermore, the increasing complexity of products and the need for immediate delivery of products put even bigger challenges on the model.

The **ERP** – Enterprise Resources Planning was the answer to overcome the MRP II limitations in the more unstable and turbulent world. Besides providing new production planning & control solutions ERP extended MRP II to cover enterprise areas that were needing attention like Engineering, Finance, Human Resources, Projects Management, etc. This is why the figure shows an ERP production planning & control module strip and a separated ERP strip. The latter represents the full ERP application that includes all areas, while the former shows only the ERP module for production planning & control. ERPs try to overcome one limitation of traditional enterprises in which different functions of an enterprise have been modelled with a focus on the performance evaluation of the individual departments rather than the effectiveness of the entire organisation (Kateel, Kamath, & Pratt, 1996). ERPs became a comprehensive and integrated set of functionalities and the organization's information key-bone integrating a wide variety of functions including sales and distribution, planning, purchasing, production, cost accounting, and finance. Some systems also include engineering management, finite scheduling, production control, and enterprise information system capabilities. In practice, however, its effective application has not been too successful. Few people can manage them and they are seldom relied upon blindly. Although ERP covers some aspects

related to production, its focus is mainly concentrated in the high level part of the enterprise. To be more competitive, companies should not only be capable of adapting and efficiently managing their high level business processes, but also their lower level processes, which are closely related to production resources. This means that a new type of software tool is needed to cope with the lower requirements imposed by production execution. These software tools and the underlying integrating infrastructure is called **MES** - Manufacturing Execution System (MESA, 2002). The focus of MES is operations, especially production operations.

Process planning (Chang, Wysk, & Wang, 1998; Teicholz & Orr, 1987), which is also known as manufacturing planning or process engineering is another area where software has been used to support its activities. (Chang et al., 1998) define process planning as "*the function within a manufacturing facility that establishes which processes and parameters are to be used (as well as those machines capable of performing these processes) to convert a part from its initial form to a final form predetermined (usually by a design engineer) in an engineering drawing*". This can be summarised as preparing detailed work instructions to produce a part.

Before the advent of computers this process was manual. In the early 1970s computers started to be used to automate report generation, storage, and retrieval of plans. In the 1980s an intense activity was done to automate process planning using artificial intelligence techniques. The application of computers to solve process planning is known as computer-aided process planning (CAPP). With technological evolution, process planners became more and more advanced, for instance by applying process knowledge, and requiring more integration. Although an ideal process planner should be able to generate a process plan for every completely new part in a completely automatic way, this situation has not been reached yet. In fact many process planners are based on the variant technique, which helps the plan generation only for parts that belong to the family of parts for which a plan has been already defined.

As computer based tools started to proliferate within the manufacturing environment during the 1980s it became necessary to integrate them. First it was required to provide suitable techniques that could accommodate the heterogeneous computer hardware platforms. Then it was necessary to develop an infrastructure to accommodate the information exchanging and sharing of data from the different applications. Individual processes had been automated without concern for compatibility with one another, which resulted in "islands of automation". Finally computer-based applications used in the manufacturing environment should be regarded as a global entity that supports the production as a whole. This concept of integrating the different computer based activities related to the planning and control of manufacturing systems is called Computer Integrated Manufacturing (CIM), a terms that was coined by Joseph Harrington in his book (Harrington, 1973). It covers, for instance, the production planning and control, CAD, CAP, CAM, and Computer Aided Quality (CAQ) applications. Ranky, on the other hand, explicitly includes shop floor control systems (SFCS) as one of the areas covered by CIM (Ranky, 1986); for him the other areas are business information systems (BIS), CAD, CAM, and CAPP. However, despite the potential behind the CIM concept it was never really a

generalised reality, and nowadays is a word that is heard very seldom, even if some of the concepts behind it are still valid. Nobody can deny, for instance, the importance of integration even if the requirements of today's manufacturing are completely different from the 1980s. In fact most of the technological infrastructure required by CIM is still needed, although improved, as it is the case of enterprise modelling, computer networks, interoperability, and exchanging mechanisms. Most of these concepts are today supported by the ERPs.

A fundamental concept in manufacturing integration is reference architectures and models for enterprise modelling. The architectures CIMOSA (Williams, Bernus, & Nemes, 1996), GRAI-GIM (Williams et al., 1996), and PERA (Williams, 1994) are examples of reference models developed to support the integration of activities under a CIM environment. Reference models are useful because with them it is possible to model the whole enterprise, achieving in this way a model driven enterprise design, analysis, and operation (Fox & Gruninger, 1998).

Computer based applications for product-engineering support have been around since the late 1960s. The first applications were conceived to help the creation, modification, and documentation of an engineering design. These are known as Computer Aided Design (CAD) applications. The proliferation of CAD systems since its first introduction led to the problem of how the designs could be exchanged between different entities (different protocols). In the beginning this was not a problem because the designs were exchanged within the same company, which, in principle, was using the same CAD application. However, when external actors started to participate in the product life cycle, the problem of exchanging different CAD representations between different CAD programmes became a reality. It was necessary to develop mechanisms that could permit the exchange of drawings and product data. An important stream of research work to solve the problem of data exchange took place, of which the most relevant results are the EXPRESS language, and the International Standard for the Exchange of Product Model Data (STEP) (Owen, 1993). STEP provides a basis for communicating product information at all stages of the product life cycle, covering all aspects of product description and manufacturing specifications. The fundamental components of the STEP are product information models and standards for sharing information corresponding to such models. EXPRESS was originally developed under the auspices of ISO TC184/SC4 to provide a formal means of defining the data necessary to describe a product (anything from a microchip to a battleship) throughout its lifecycle, from the time of conception through its manufacture to its time of disposal. STEP uses EXPRESS as the formal specification of the required data and its relationships. It must be pointed out that there are other formats for CAD exchange like IGES – Initial Graphics Exchange Standard, DXF, and CADL (neutral formats).

Computer Aided Manufacturing (CAM) aids in the planning, management, and control of manufacturing processes. Please note that in fact CAM is more about machining than about manufacturing. Based on a geometric model, the application is able to generate numerical control (NC) programs that can be downloaded directly to the machines. It is easy to see that CAD and CAM are closely connected. One produces the geometric models while the other generates the programs.

This tool was a big improvement when it was first introduced since the time required from designing the product to producing it was shortened and the task became less error prone. NC and CNC machines have reached some form of conformity for the downloading of programs based on product data features. Unfortunately the same kind of success has not been achieved in other shop-floor machines, such as industrial robots (Onori, 1996). Nevertheless, CAD/CAM is becoming more and more advanced, with more features to guide the designer in his task, and more integrated in the manufacturing environment.

The new tendency for more complex products, more complex supply chains, and new government regulations imposed new challenges to manufacturing companies in terms of product traceability. This signifies that the full life cycle phases of the product must be supported in order to be possible to identify each product's position within its life cycle phase. The application of information technology is thus the only solution to achieve this goal. All aspects related to information technology required to support the full product life cycle is called Product Data Technology (PDT), which was coined in the context of European programmes[10]. Nowacki states that PDT "*refers to a comprehensive subject area that embraces all innovative aspects of the modern production and information technologies from an integrative point of view*" (Nowacki, 1998). PDT when compared to CAD/CAM, is a step further because in addition to covering the product design phase (CAD/CAM) it also covers the other phases. Several techniques and methods are required to ensure the interoperability between the different information systems involved in the whole product life cycle. STEP is widely used to support product data exchange. So far PDT ability to cover all its initial dreams has been limited, except for the exchange of product data and data sharing. Better interoperability is mandatory for successful PDT implementation as well as better knowledge process support. In fact, PDT is now restricted to the aspects related to data sharing and exchange. The more complex aspects related to full coverage of product life cycle is covered by what is now known as Product Data Management (PDM).

2.1.2.7 Shop floor control

Figure 2-9 shows the evolution of shop floor control as well as the evolution of some computer-based technologies that were important for the development of shop floor control solutions.

Initially the automation of production lines was achieved using only purely mechanical solutions[11]. As would be expected, mechanical solutions have some limitations in terms of being able to control complex situations, and even in the cases where this is possible, such solutions require much maintenance, are complex to build, subject to wear, and any required change implies complete rebuilding of the system. Mechanical solutions are very inflexible whenever changes are required. Pneumatic control, which is still very popular for certain applications, uses compressed air, valves and switches to construct simple control logic. Despite exhibiting slow response times, it is easier to build

[10] One of the most recognised projects in this area is the ESPRIT project PDTAG-AM (ESPRIT 9049)
[11] Cams and governors are examples of mechanical controls (Chang et al., 1998).

than mechanical control because it is possible to construct logic functions using standardised components. However, reprogramming is difficult and time-consuming since it requires rewiring air ducts.

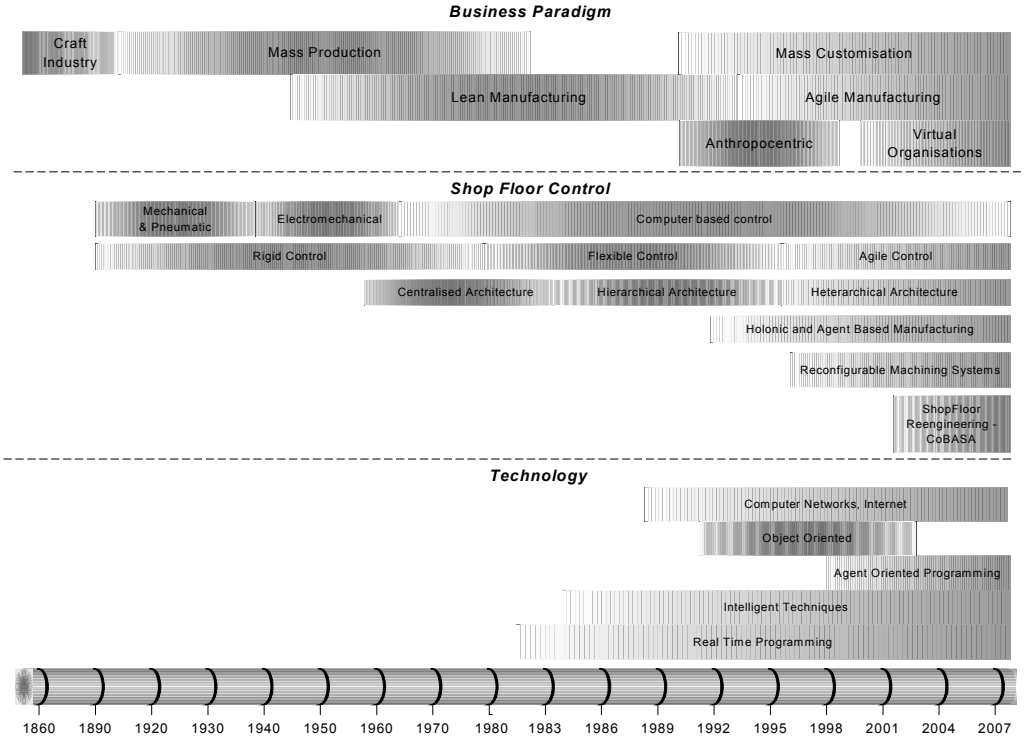

Figure 2-9 – Shop floor control

Before the advent of computers the creation of electromechanical control solutions became possible. An electromechanical control uses switches, relays, timers, counters, etc, to build control logic. Since electrical signals move faster than air, this type of control was faster than pneumatic control. Although far from the flexibility of today's computerised solutions, it is much more flexible than mechanical and pneumatic control because it is possible to build more complex logic and it is also easier to rewire electrical cables than air ducts.

Even if mechanical and electromechanical control solutions could only achieve poor flexibility, and required fairly high skills, this was not very problematic in the era of mass production because the goal was to rapidly produce as many pieces as possible, without changing the product. This signified that few control alterations were needed. However, this situation changed when the demand for more product variety started, and then these rigid (fixed) control solutions were no longer effective. Despite this need, it was only possible to create more flexible control solutions with the advent of computers because they are inherently flexible. With them it is possible to create a control sequence and, if not appropriate, that sequence may be completely altered just by software (programming). This was a revolution in terms of production systems design, and it was the most important enabler to build production systems able to cope with diverse products, i.e. flexible production systems.

In the 1980s it started to be felt that only with flexible control and supervision architectures, either for manufacturing or assembly, would it be possible to overcome the difficulties of manufacturing companies. Flexibility of a system is defined in (Gupta & Buzacott, 1991) as the "*ability to cope with changes of its environment*". However, different types of flexibility exist. The most important division is between capability and capacity flexibilities (Tichem, 2000). Another type of flexibility is the ability to keep up with errors, which was identified in (Brynjolfsson & Arnström, 1990; Camarinha-Matos et al., 1996; Donald, 1989; Heilala & Voho, 2001; Redford, 1989). While the term capability is related to the ability of the system to react to product variants that are asked for[12], capacity is related to the ability to cope with varying demands (quantities)[13].

Flexibility is mainly constrained by the physical equipment and the control/supervision architecture (software and programming). Feeders and grippers, for instance, constrain the kind of components that can be handled (capability constraint) (Onori, 1996). It is important, nevertheless, to emphasise that flexibility and adaptability cannot be achieved through mechanical solutions alone. The control architecture can limit the capability, by limiting, for instance, the flow of materials, the change of tasks, the ability to add and remove equipment, and also the capacity since the range of operations that can be executed at any one time is limited[14]. The ability to change equipment and handle complex conditions, i.e. most of the controllers still run on state-diagrams and IF_THEN_ELSE variable/condition handling, is one of the most important drawbacks of current control architectures. Changes in the manufacturing equipment are very important to ensure flexibility and should be done quickly (reconfigured). However, the control architectures used for FMS/FAS are poorly configurable, and changes in the shop floor equipment require big programming efforts, which are costly and error prone.

The main problems with the control and supervision architecture is how to integrate the various heterogeneous controllers that can be found on the shop floor, how to supervise and synchronise the various tasks, and how to develop a strategy to facilitate the plug and unplug of equipment. Much research was done in the 1980s and early 1990s but none resulted in commercially adaptable solutions. The problem is not with the flexibility of the individual machines, namely CNCs and robots that are reprogrammable, but with the system considered as a whole (integration). The complexity of the shop floor and the diversity of applications and operational requirements just increase the difficulty of integrating them because the system becomes difficult to understand, generalise, and standardise. Being able to develop a control and supervision architecture whose programs could be independent of the process has been the dream of shop floor engineers ever since. In fact, this dependence is a source of inflexibility because whenever a machine (resource) is changed, the product is changed, or the process plan is changed, it is immediately necessary to change the programs of the individual

[12] This is also known as product flexibility (Vos, 2001).
[13] Also known as volume flexibility (Vos, 2001).
[14] The Mark III (Onori et al., 1997) assembled 7 products in 800 variants, the total capacity was of 350,000 products per year. This was possible due to the new control architecture (FACE) that supported the co-existence and handling of a great number of robot, PLC, vision and other programs. Orders could be changed on-line.

components. The CoBASA architecture being introduced in this thesis tries to solve this problem in part.

The lack of ability to plug and unplug equipment is also very important in the context of this thesis, and the architecture being introduced here is a partial answer to this problem. To achieve true flexibility the control architecture should be able to absorb new equipment without big programming efforts. These new requirements for shop floor control and supervision architectures, in which configuration is favoured over reprogramming whenever changes are required are becoming known as agile control/supervision shop floor architectures, to be distinguishable from the traditional rigid and hierarchical control/supervision architectures. The advent of multiagent based architectures seems to be an important enabler for the development of this type of control architectures, which was not easy using traditional programming paradigms.

The following points describe some of the relevant control systems architectures for manufacturing systems (Leitão et al., 2001):

- **Production Activity Control (PAC) -** The ESPRIT project 447, COSIMA (Control Systems for Integrated Manufacturing) (Bauer, Bowden, Browne, Duggan, & Lyons, 1991), developed a functional software architecture for cell and shop floor levels. The proposed PAC concept defines five different modules: Scheduler (which plans the manufacturing resources according to long term tactical plans and resources capacities), Dispatcher (which is the heart of the control system and acts in real time control over the manufacturing environment), Monitor (which collects shop floor data to give a logical view of the actual states in the manufacturing environment), and Producers and Movers (that control the shop floor resources). The coordination between PAC sub-systems is performed by the FC (Factory Co-ordination) module.

- **CHAMP** - The CHAMP (Chalmers Architecture and Methodology for Flexible Production) architecture is a modified hierarchical reference architecture, to model the control software of manufacturing cells, which was developed at Chalmers University of Technology. It derives from the experiences made with implementations of control systems based on the PAC architecture and its extensions, PAC+ and PAC++. The CHAMP architecture includes functions for scheduling, dispatching, resource control, monitoring and error handling (Gullander, 1999). The main features of this architecture are the separation of product and resource information, and the physical separation of generic functions from specific functions of the products and the resources currently in use.

- **RapidCIM** - The RapidCIM was a joint venture project between Texas A&M University, Penn State University and Systems Modelling Corporation. The RapidCIM objective was to facilitate the process of developing fully automated computer controllers for Flexible Manufacturing Systems (FMS). The basic components of the RapidCIM concept are the Shop Floor Architecture (based in a hierarchical architecture), Factory and Process Plan Model,

Formal Models of Execution and associated tools (for development of execution software) and Simulation for Real Time Control.

- **MOSCOT** (Modular Shop Control Toolkit for Flexible Manufacturing) - This architecture is characterised by two main parts: a kernel, which contains the common modules (objects) to all shop floor applications and the shells, which are developed or customised according to the specifications of each applications (Teunis, Leitão, & Madden, 1998). An innovation introduced by this architecture is the SCAPI (Shop Control Application Program Interface), which acts like an operating system for shop control applications developers.
- **CCP´s Manufacturing Cell Controller** - CCP´s Manufacturing Cell Controller Architecture was developed and implemented for the Flexible Manufacturing Cell of the CIM Centre of Porto. The flexible manufacturing cell has two CNC machines and an anthropomorphic robot for the load/unload of the machines. The CCP´s Manufacturing Cell Controller Architecture uses a modified hierarchical architecture approach and is composed by a set of several modules whose main controller is the Manager Module, which is responsible for the control and supervision of the production process of the manufacturing cell and also for the management of cell resources (Leitão & Quintas, 1997). Each physical device has a module, designated by Device Controller, which is customised to each industrial machine, such as production or handling equipment, and it has the responsibility for the local control of the machine and for the execution of the jobs requested by the high level module. The interface between the Cell Controller and each of the industrial machines is implemented using the MMS (Manufacturing Message Specification) communication protocol.
- **Flexible Assembly Control Environment (FACE)** – This architecture was developed at KTH under a PhD work (Onori, 1996). The main functions, which the FACE system is designed to account for, are cell control, process supervision, and robot task program generation. Due to the de-centralised database and networking possibilities, it was decided to develop a separate module for each function (Onori et al., 1997): 1) Control Module, 2) Supervision Module, and 3) Robot Motion Module. Only the Control Module is compulsory for the running of the FAA cell. The FACE system architecture has been implemented mainly according to the revised PAC model (Maglica, 1996). The Control Module runs on a PC and is the engine in the system. It decides which sub-batch to assemble and it gives the orders to all the equipment. It consists of short and long-range schedulers and a dispatcher. During assembly, the dispatcher is fully occupied with the communication. Therefore, in order to provide easy access to the system, an operator interface (Supervision Module) was added. The third FACE-module is the Robot Motion Module (Onori 1997). In this module, which also runs on a separate PC, the robot and track motions may be programmed off-line.

- The **Holonic manufacturing** (Bussmann & McFarlane, 1999; Gou, Luh, & Kyoya, 1998; Tharumarajah, Wells, & Nemes, 1996; Van Brussel, Wyns, Valckenaers, Bongaerts, & Peeters, 1998) started in the early 1990s supported by an IMS – Intelligent Manufacturing System project. The main idea behind holonics is the ability to "*plug & play*" manufacturing components, and the basic IT paradigm is agent oriented programming.
- **Multiagent based architectures** are gradually making their appearance in the manufacturing world because agents can be an adequate paradigm to implement agile shop floor control architectures. The most relevant ones are: MetaMorph (Maturana, Shen, & Norrie, 1999), AARIA (Autonomous Agents at Rock Island Arsenal) (Parunak, Baker, & Clark, 1997), MASCADA (Valckenaers, 1997), HOLOS/MASSYVE (Rabelo & Camarinha-Matos, 1996), DEDEMAS (Decentralised Decision-Making and Scheduling) (DEDEMAS, 1998), B-LEARN (Camarinha-Matos et al., 1996), and TeleCARE (Camarinha-Matos & Vieira, 1999).
- The **Reconfigurable Manufacturing Systems** (RMS) concept was developed in the mid 1990s at the University of Michigan by the Department of Mechanical Engineering and Applied Mechanics. RMS is a production system architecture that answers shop floor challenges imposed by agile manufacturing (Huff & Edwards, 1999; Koren et al., 1999; Mostafa G. Mehrabi, A.Galip Ulsoy, & Yoram Koren, 2000; M.G. Mehrabi et al., 2000). In (M.G. Mehrabi et al., 2000) RMS "*will allow flexibility not only in producing a variety of parts but also in changing the system itself. Such a system will be created using basic process modules – hardware and software – that will be rearranged quickly and reliably*". In terms of flexibility RMS stands between FMS and dedicated transfer lines. Its main goal is not so much flexibility but above all responsiveness. The system aims to be continuously updated and rapidly reconfigured. This achieved by using new programming tendencies (agent oriented programming, open control architectures, …) and modular machines. Although RMS can be used for areas other than machining, it is in this area that it is mainly applied.

The technology evolution shown in Figure 2-9 clearly indicates that suitable technologies to implement the IT tools needed by manufacturing enterprises to cope with the current manufacturing challenges only started to become widely available towards the end of the last century. This is the case of highly available computer networks, intelligent techniques, and agent oriented programming. Although object oriented programming has been very important up to now, it might be replaced in the future by agent oriented programming due to being more promising in terms of *plugability*.

2.2 CURRENT CHALLENGES

In the previous section a brief historical evolution of the manufacturing paradigms was done to illustrate how manufacturing requirements have changed in the considered period and how they have influenced the requirements for shop floor control and supervision architectures. The focus of this section is in finding out what the current challenges faced by manufacturing companies in the age of agile manufacturing and virtual organisations are. Roadmaps, informal interviews with production managers involved in the analysis, design, and implementation of production systems, and other observations are used as supporting sources to find the current challenges faced by manufacturing companies.

2.2.1 Roadmaps

Roadmaps are important sources of evidence about the challenges and trends in manufacturing because they correspond to studies involving a reasonable number of industrialists as well as academic experts.

Visionary Manufacturing Challenges for 2020. Several studies and roadmaps have been produced in the last years to analyse the current and future challenges to be faced by manufacturing companies. One important study is the "Visionary Manufacturing Challenges for 2020" from the USA (NRC, 1998). This study has identified the most important factors that will likely affect the development of manufacturing. The identified factors, related to the problem of changing/adapting the production system, are:

- **Competitive climate**, which will require rapid responses to market forces.
- **Sophisticated customers**, who require increasingly customised products.
- **Development of innovative process technologies**, which change the scope and scale of manufacturing.
- **Environmental protection to minimise use of extinguishable resources and meet legal goals**.
- **Information and knowledge**, covering various enterprise levels including the shop floor, can be used, for instance, for decision making when changing/adapting the system or for maintenance purposes (operating phase of the life cycle).
- **Global distribution of highly competitive production resources**.

Only manufacturing companies able to cope with **flexibility and responsiveness** (agility) will be able to survive in an environment suggested by these factors. The study also identified the major challenges or goals that need to be addressed when moving manufacturing companies from the current status to manufacturing in 2020. Interestingly, the report considers the *rapid reconfiguration of*

manufacturing enterprises rapidly in response to changing needs and opportunities as one of the "grand challenges".

IMTR Roadmap. The IMTR – Integrated Manufacturing Technology Initiative produced another roadmap (IMTI, 2000c), which has identified six "grand challenges" for manufacturing:

1. Lean, Efficient Enterprises,
2. Customer-Responsive Enterprises,
3. Totally Connected Enterprises,
4. Environmental Sustainability,
5. Knowledge Management, and
6. Technology Exploitation.

The visions this roadmap has produced and important for the goal of this section are:

- "Total process control gained with equipment that is self learning, self-diagnosing, self correcting, and plug &play" – related to item 1
- "Intelligent, self-configuring systems will enable instant plug & play of new enterprise functions with zero integration time & cost" – related to item 3
- "Manufacturing systems built on a modular framework using intelligent modules able to learn from experience and reconfigure themselves to dynamic conditions" – related to item 5

The term reconfiguration is again a key word in this roadmap. The concept 'plug & play' can be extended to 'plug & produce', which is an important characteristic of any shop floor. In fact, if the units that compose the shop floor can be easily plugged together, the shop floor becomes more agile. It is important to stress that other roadmaps of the IMTI initiative indicate the same tendency towards a responsive and agile shop floor (IMTI, 1999, 2000a, 2000b).

IMS Roadmap. The roadmap developed by the IMS Europe secretariat (Cahil, Garello, & Jering, 2001) states that the future of manufacturing and production sector in Europe will be determined by how it meets the challenges of "new manufacturing", which is identified as the shift from the dominance of product in many value chains to the dominance of services. The pace at which the "new manufacturing" is implemented is considered by the roadmap as significantly influenced by the development of a wide range of new methods and technologies. The challenges identified are:

1. Concurrency,
2. Integration of Human and Technical Resources,
3. Conversion of Information to Value Creation,
4. Sustainable Management Systems,
5. Reconfigurable and Flexible Enterprises, and
6. Innovative Processes.

The concurrency challenge intends that the conceptualisation, design, and production of services should be as concurrent as possible. At this level the report states, that the level of concurrency be needed will require technological advances in *modular, adaptable design methodologies* and *adaptable manufacturing processes and equipment*. For the challenge 2 the roadmap indicates, among others, that *adaptable, reconfigurable manufacturing processes and systems* enabling technology should be improved. The important aspect mentioned in challenge 5 is the one related to the reconfiguration of a production system. Here the important concepts are *rapidly changing customer needs* and *rapidly changing market opportunities* and the impact they have in the production system. An improvement in the *reconfiguration of manufacturing operations*, which includes an adaptable and reconfigurable shop floor, is considered as a basic requirement to address this challenge.

Assembly-Net Roadmap. The roadmap European Precision Assembly – Roadmap 2012 (Onori, Barata, Lastra, & Tichem, 2002) from the Assembly-Net project (Assembly-Net, 2002) has also identified major challenges to be confronted by the European industry in the forthcoming decades. The globalisation of markets, shorter product lifecycles and miniaturisation of products were identified as major threats. However, rapidly diminishing labour forces, indiscriminate outsourcing, and the advent of radically new technologies (ranging from Virtual Enterprising to Nano products) have also been considered as having a debilitating effect on European industry if no adequate measures are taken immediately. The important vision brought in by the roadmap is the Evolvable Assembly Systems – EAS (Onori, 2002a), which considers an assembly system (production unit) as a living entity that evolves along its life cycle – evolution rather than adaptation. The roadmap also shows a generic product life cycle view that includes production system and product related activities (Figure 2-10).

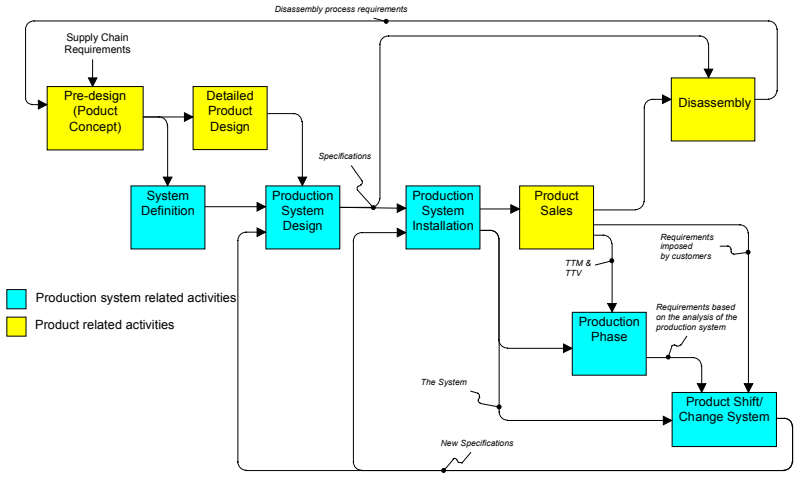

Figure 2-10 – Generic product life cycle view (Onori et al., 2002)

The important aspect for this thesis is that an activity for shop floor reengineering is clearly indicated as important in the life cycle of the production system – *Product Shift/Change System*. Although this phase is considered as important the roadmap also states that it has not been solved yet,

and is still an open problem. Another important conclusion from the roadmap is pinpointing the fact that "plug & produce" solutions are an important research theme. The roadmap also states that process knowledge is vital and lacking. Without process knowledge it is not possible to have stable processes, which in turn imply the lack of models or standards. Without these models it is difficult to create agile shop floors and, consequently, manufacturing VOs.

THINKcreative Green Report. The THINKcreative project (THINKcreative, 2002) produced a green report (Camarinha-Matos & Afsarmanesh, 2002), which introduces a set of interim recommendations regarding the required research in the area of virtual organisations and other emerging collaborative networks. Several driving forces faced by manufacturing companies in particular were identified. The most important ones to be considered, in the context of this thesis, are:

- Globalisation.
- Customer orientation.
- Complexity of products and production systems.
- Complexity of collaborative networks (supply chain).
- Turbulent environment.
- Focus on core competencies.

To be able to cope with these "driving forces" companies must implement new forms of collaborative mechanisms, as it is the case of virtual organisations or enterprise networks. The study then stresses that cooperation ability is even more important to SMEs (small and medium-sized enterprises), which are predominant in Europe and have to cooperate more intensively with other enterprises throughout the product life cycle. Cost reduction, increased flexibility, and focus on core competencies can be achieved in scenarios of virtual organisations, which makes individual organisations more competitive. An important conclusion of the roadmap related to the focus of this thesis is considering the area of shop floor reengineering as an important item that needs focused attention.

Report on Flexible Assembly Automation. This report was produced by Marcel Tichem from the Delft University (Tichem, 2000). In this work the author answers two questions. The first question is about the causes for the low degree of application of flexible assembly automation in industry, while the second one asks what can be done to increase the applicability of flexible assembly automation. The bottlenecks identified to be causes for making the application of flexible assembly automation difficult are:

1. Wrong attitude and strategy towards assembly and assembly automation.
2. Insufficient production volumes.
3. Uncertainties in forecasts on production volumes and product-mix to be handled.
4. Products are not designed for assembly.

5. Unreliable assembly automation techniques.
6. Lack of planning, programming, and control aids.
7. Wrong introduction of automated solution in company.

From these bottlenecks points 5 and 6 are directly related to the work of this thesis. To solve the bottlenecks, several research directions are indicated: 1) support for modular product design, 2) development of assembly system concepts, 3) development of standard assembly technology, 4) technology for precision assembly, 5) **development of programming and control techniques**, and 6) development of disassembly technology.

In the development of assembly system concepts, one of the identified goals is *fast set-up, reconfiguration and dismantling*, which is right in accordance with the target of this thesis. The importance of modularisation and standardisation of the system concept as well as the importance of integrating the control architecture with the development of the production system are also mentioned. The report also states that the cost related to programming Flexible Assembly Systems is high compared to the hardware costs and so new control architectures are needed that privilege configuration over programming.

2.2.2 Informal interviews

Some interviews and working meetings were conducted with production managers, which do not intend to be statistically relevant but their conclusions are included to illustrate that these managers also support the roadmap results.

An informal interview with a production manager from **BLAUPUNKT** car radio assembly plant in Braga – Portugal showed that it is quite common that some of their major customers require changes to the initial requirements during the assembly period of the product. These changes are mostly related to quality control (final assembly); however sometimes other changes involving the product itself are also required. These changes involve, in many cases, modifications that require changes in the layout of the assembly system. Unfortunately for BLAUPUNKT these changes in the layout also imply a great effort of programming, which is very time consuming. Another reason mentioned for requiring frequent shop floor reengineering is the short life cycle of the products.

In order to better understand the problem BLAUPUNKT faces, two scenarios, based on real cases, are considered:

First Scenario – Assembly of Car Radio Front Panels. Originally this assembly line was conceived to produce one type of front panel for car radios. The work stations were distributed along the conveyors. They offered manual assembly operations and manual inspection operations. The inspection of the radio buttons and knobs was completely manual and limited to visual inspections. The line (Figure 2-11) had to be changed (reengineered) because the production department was being

pushed by its customers to increase the quality control and, at the same time, it was necessary that the assembly line could produce more than one type of front panel. One of the major concerns from the customers implied the need for an automatic inspection system that would be based on artificial vision.

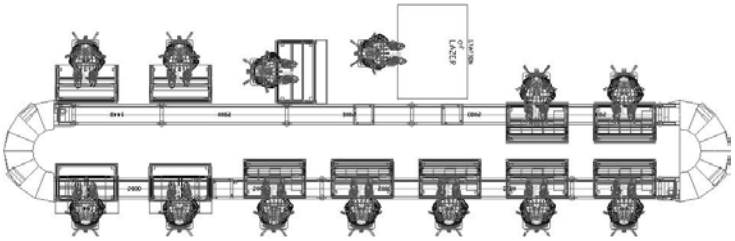

Figure 2-11 - First scenario, before reengineering

To answer these requirements a new layout was proposed (Figure 2-12) which integrated an automatic inspection system based on artificial vision and a robot. A requirement from the management was the need to use, as much as possible, components from the old assembly line (conveyors, work stations, motors, …). These requirements implied a completely new layout as shown in Figure 2-12. The reengineering aspect resulted from the fact that parts of the old line were used in the new one.

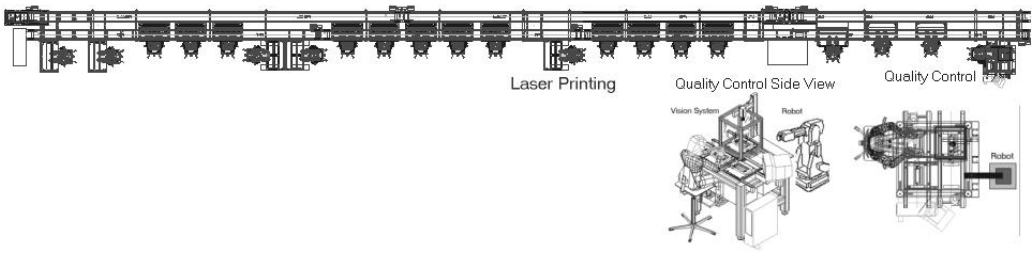

Figure 2-12 - First scenario, after reengineering

To obtain the reengineered assembly line it was necessary to go through the following major activities:

- Requirements specification.
- Design several alternative solutions.
- "What-if" analysis to choose the most adequate solution.
- Dismantle the old line.
- Choose the components to be reused in the new assembly line.
- Mechanical implantation.
- Programming of the PLC, Robot, Vision System.
- Integration of the assembly line controller with the Information System of the factory.
- Tests.
- Skills definition for the operators.
- Documentation and definition of operating procedures.

Although legacy controllers from the old line were used, a big programming effort was still necessary because the old software could not be used. The main reason for this was that the software was too dependent on the hardware. One big difficulty also felt by the development team was the lack of knowledge about some supporting aspects (i.e., technical knowledge about the legacy systems).

Second Scenario – Optimisation of the Throughput of an Assembly Line. In this scenario the assembly line described in Figure 2-13 was having serious throughput problems because of poor design. This line had 3 parallel conveyors and the workstations were mounted along each of the outer conveyors. The left outer conveyor serves the workstations located on the left, while the right conveyor serves those on the right, and the central conveyor is used for the pallet's flow. The parts of the right and left conveyors that are near the workstations are used as buffers to them. The 3 conveyors represented in the bottom part of Figure 2-13, are physically located in the bottom of the assembly line, below the others. They were designed to be a return path for the pallets from the end to the beginning of the assembly line. Pallets carrying unassembled car radios visited the workstations, according to a specific sequence (process plan). In each workstation one or several operations from the process plan were performed, and after that the pallet follows its way to the next workstations. A crossing section to allow the transfer of pallets from one conveyor to another was installed before each workstation. Whenever a pallet reached a crossing section, the control system guided the pallet to the appropriate conveyor. If the next working place of the pallet was located either on the right or on the left conveyors and if there was no available place there, it was transferred to the central conveyor. When a pallet reached the end of the conveyor it was transferred to the bottom conveyor, using an elevator, and moved to the beginning of the line. The amount of pallets travelling resulted from the length of the line. If a pallet did not find a place in the workstations located at the beginning of the assembly line it was forced to travel along the top conveyor and return through the bottom conveyor. The production team concluded that there were a large amount of pallets travelling in the central conveyor.

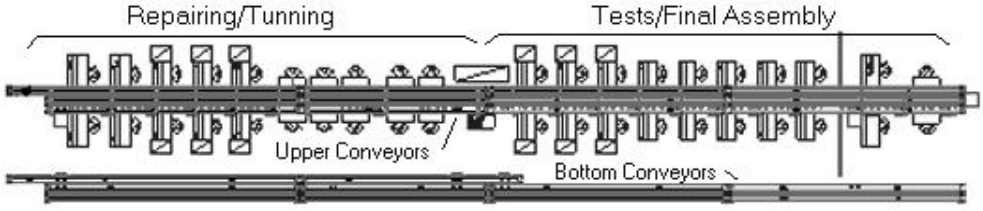

Figure 2-13 – Second scenario, before reengineering

The reengineering of this assembly line started with several simulations of its operation ("what-if" analysis"). It was found that if two elevators were inserted in the middle area that separates the Tests/Final Assembly area from the Repairing/Tuning, a big reduction of pallets' traffic was observed (Figure 2-14).

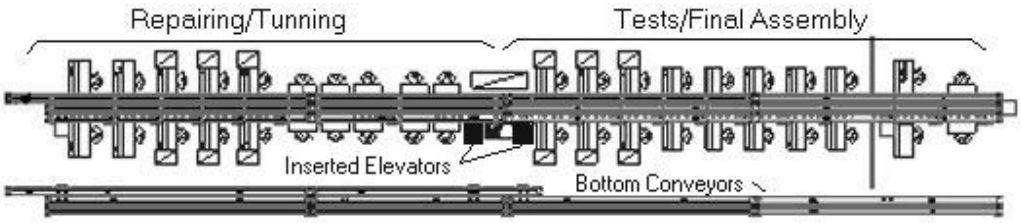

Figure 2-14 – Second scenario, after reengineering

Besides the simulation activity that was necessary during the reengineering of this assembly line, other activities were required. Although the hardware changes were not as profound as in the previous case, the modifications of the control program were significant because it was not very flexible. This meant a lot of effort in software changes, which also implied more time spent during the tests phase. It seems clear from this example that the control architecture should be developed in such a way to make it easier to adapt. For the programmers it would be very helpful if they could configure the controller instead of developing it from scratch.

Another informal interview with a production manager from the **GRUNDIG** assembly plant, also in Braga, was in the same direction as the previous one.

Mr. Friedrich Durand, who is the technical manager from **SMH Automation** in Switzerland[15], made a presentation[16] about the dream of a production manager:

1. *I am the chief of a production site and I have gotten the request to produce 10'000'000 pieces of a known product per year.....*
2. *The planned production time for the actual product is defined with a half year, then an advanced product or a new product will run on the same line*
3. *I call the "Rent-a-Automation" firm "Rentomatik" and I order 20 different standard pick and place stations ...*
4. *I call the "Rent-a-Testequipment" firm "RentTest" and I order 20 standard test stations*
5. *I call the "Integration" firm "IntegralAutomation" for the integration and specification of the line and the training of the production crew*
6. *After 3 weeks I will get the rented standard machines, adoption will be done and after 4 weeks the line will produce the zero series*
7. *After another month the production is changed to series production*

[15] SMH AUTOMATION belongs to the centres of competence of ETA SA Fabriques d'Ebauches, a company of the SWATCH GROUP. The activities of SMH AUTOMATION are divided in two sections: 1) the development and realisation of special machines and systems for the mass production of micro technological products, which are named and sold as "Micro Assembly", and 2) the production of precision cutting tools. These are sold under the trademark "TECHNICA".

[16] The presentation was made in the general meeting of the Assembly-Net project (Assembly-Net, 2002) in Aachen – Germany

8. *After half a year I add additional standard sockets and a second model of the product will be produced*
9. *After a year I will produce piece per piece and another variant will be produced on the same line (personalized production)*

In the same meeting Mr. Durand presented his vision for today's company requirements, some of which are:

- Changes in consumers demands.
- Saturation of the markets and pressure to reduce costs.
- Reactivity (companies must be able to react promptly).
- Decreasing in product life cycle versus increasing in product diversity.
- Globalisation.

According to him the production system should be able to accommodate different products and versions as well as preserving its value, which means that whenever the production shifts to a new product most of the production unit components should be reused. In addition, the production system should be reliable, easy to maintain, and modifications, when necessary, should be fast and without much interference with production.

2.2.3 Other observations

A clear indication that the world of manufacturing companies is changing very quickly is given by the increasing number of communications from the automotive industry (GM, Fiat, Ford, VW …) to some international conferences like PDT Days (http://www.pdteurope.com/) expressing the challenges those companies are dealing with and the research work that must be done to stay competitive. Among those challenges, automotive companies point out the growth of product complexity, reduced time to market, improved product quality and customisation, and increased complexity of the supply chain network with the establishment of virtual world wide distributed and networked enterprise nodes, which imposes strong co-operative challenges to the different enterprise levels, from management to shop floor. Despite the fact that these challenges are currently being pointed out by big players it does not mean that they are not a problem to SMEs. On the contrary, as sub-contractors, SMEs will be important players as nodes of virtual organisations, which makes the referred challenges important to all manufacturing companies.

Another sign that shows the importance of virtual organisations and enterprise networks as future business paradigms with impact on the way manufacturing is organised is the number of projects related to this subject currently going on, namely THINKcreative (THINKcreative, 2002), VO-Map (VOmap, 2002), COVE (COVE, 2002), CE-NET (CE-NET, 2002), and VOSTER (VOSTER, 2002).

A summary of the manufacturing challenges faced by current manufacturing companies is illustrated in Table 2-8. To stay competitive in an environment characterised by these challenges

manufacturing companies need fast reconfiguration or adaptation (agility). To compete in this complex, changing, and unpredictable environment, manufacturing companies need an agile manufacturing organisation. Their production systems need to be rapidly designed, quickly adapted to new products, and quickly adjustable to market demands.

❑ Evolving market scenarios ❑ Customer desire for exclusive product variety in small quantities (mass customisation) ❑ Demands for higher quality and lower prices, increased product complexity, faster delivery times, and shorter product life cycles ❑ Moves of major assembly companies from one region to another (outsourcing) ❑ New environment protection regulations	❑ Improvements of the logistics and information systems infrastructures, which, although facilitating cooperation among enterprises also contribute to reducing any advantage of being in a particular geographical region ❑ Fast technological evolution ❑ The emergence of the so-called "new economy" privileging intangible things (information, knowledge, relationships, …) in opposition to the material products represents another major challenge to the manufacturing world

Table 2-8 – Manufacturing challenges

In fact, the number of engineering projects within manufacturing enterprises has increased in the last decade, and most of them regard adaptations of the production system. These changes are not only at the physical level. With the growing levels of systems integration any change to the physical infrastructure has consequences for the control and supervision system. This usually requires a hard programming effort, which is an obstacle to agility. It is also well known that most of these changes during the system's life cycle are not appropriately documented, which represents another obstacle whenever new changes are necessary. Frequently the engineers in charge of the reengineering process find out that the documented models of the manufacturing system do not correspond to its current version. Furthermore, participation in collaborative networks such as virtual enterprises will progressively require the possibility of "opening a window" over the shop floor to allow authorised partners to "see" and even take part (remote supervision) in the joint business processes taking place locally. This requires knowledge-based models of the shop floor systems and processes, not only to support the adequate levels of visibility and privacy but also to give the remote partners an understanding of the local system configuration at each moment. There is therefore a need to develop methodologies and supporting tools to help in the task of shop floor reengineering.

2.3 A SCENARIO FOR A SHOP FLOOR REENGINEERING METHODOLOGY

This section describes a suggested shop floor reengineering methodology scenario to face the indicated manufacturing challenges. The objective is to better illustrate the framework under which

this thesis has been developed rather than to propose a complete shop floor reengineering methodology.

The online Dictionary (Dictionary, 2002) defines **methodology** as *"a body of practices, procedures, and rules used by those who work in a discipline or engage in an inquiry"*. The methodology to support the process of change (reengineering methodology) is therefore the set of methods, principles, and rules that will guide the user through the process of changing/adapting the shop floor following an easy, user friendly, and as much as possible error free approach. The tools that support the methodology are the methods, while the principles and rules determine how these tools are used and created.

Any process of change (reengineering) always involves a life cycle change that might be applied partially or wholly to the shop floor. The reengineering methodology is essentially a methodology to guide the user in the process of changing the production system from one life cycle phase to another. The main life cycle phases to be considered associated to any existing production system are:

- System definition,
- System design,
- System installation,
- Production,
- Reengineering, and
- Dismantling.

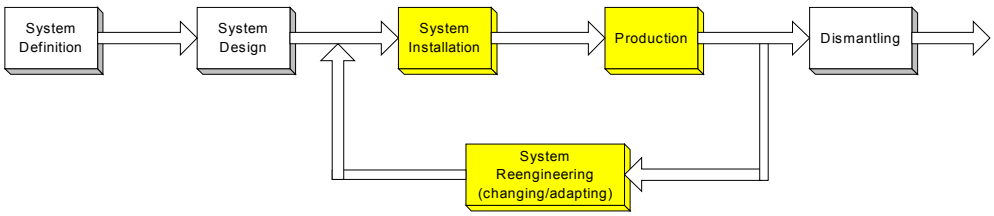

Figure 2-15– Production system life cycle phases

The production phase includes the phases operation, and analysis and evaluation.

A successful and effective implementation of the reengineering methodology requires the use of application tools to guide the user through the process of change. Two main aspects should be considered when developing such tools. The first aspect considers the functionalities required to support the life cycle evolution, while the second one considers how the tools can be helped by a series of other tools. This means an architecture composed of a central tool that represents the core of the methodology and a set of other tools that interact with this one in a loosely coupled way (Figure 2-16).

The Methodology Supporting Tool (a kind of eBook) runs through the various phases and keeps track of the most important activities of each life cycle phase. It gives information about the available tools for some of the activities, and synchronises the supporting tools. The eBook tool works on the

required information models to be used either by the system itself or by the supporting tools. It acts like a checklist and different ways of communication are used to interact with the different actors. The most important aspect in the architecture is to provide a convenient infrastructure to integrate these different and heterogeneous tools.

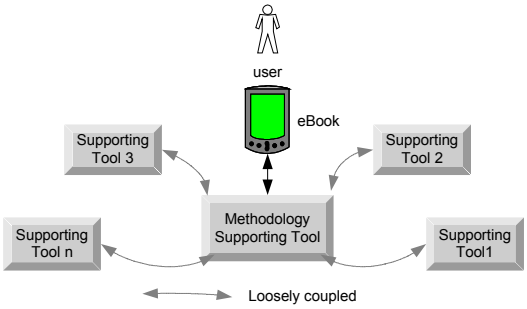

Figure 2-16 – Proposed architecture to support life cycle evolution

Figure 2-17 shows the global context of the reengineering life cycle support process and its main inputs and outputs. If reengineering in dynamic environments is a normal part of the life cycle rather than an exception, the terms "reengineering process" and "life cycle support process" are here used indistinctly.

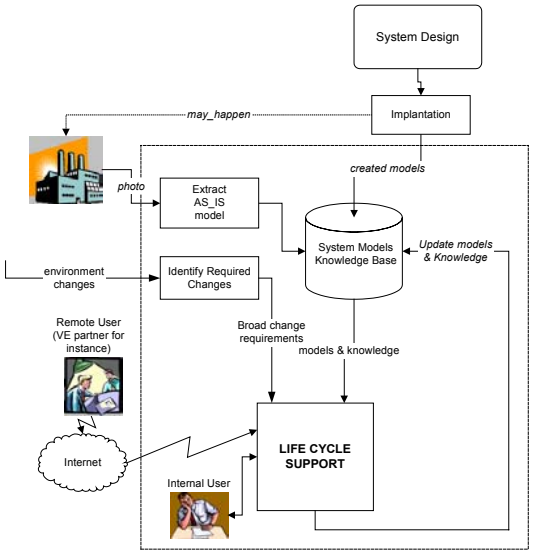

Figure 2-17 – Context of the reengineering life cycle support process

In case of a legacy system for which an updated model does not exist, it is necessary to start the process with the elaboration of a model of the current system (AS-IS). Another important activity that occurs within this framework is the identification of required changes. This step generates a non-formal rough specification of changes that are needed.

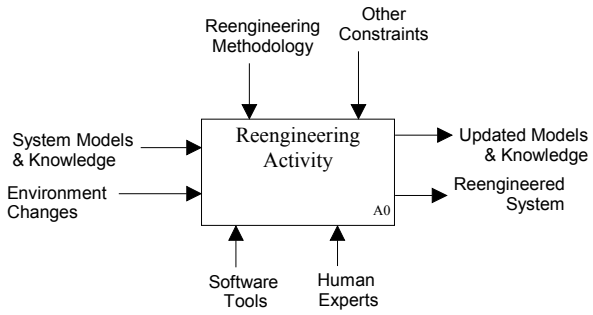

Figure 2-18 – Reengineering activity

The generic representation of the reengineering activity (life-cycle support) is shown in Figure 2-18, using an IDEF0 representation. The reengineering activity transforms system models and knowledge into a reengineered system, based on a reengineering methodology and other constraints, using software tools and human experts. During this process the system models and knowledge are updated. The life cycle support process itself can be decomposed into the following steps:

1. Formal specification of requirements
2. Development/adaptation of processing architecture
3. Development/adaptation of control system
4. Management of the human resources
5. Implantation.

The formal **specification of requirements** is needed to transform the rough definition of required changes into a formal specification that can be used by human experts and computer tools that support the other activities of the re-engineering process.

The development/adaptation of the **processing architecture** is the activity in which the design of the structure and layout of the re-engineered system occurs. The generated architecture is created from the previous architecture and updated in the System Models & Knowledge Base (Figure 2-17). This activity includes important sub-activities already envisaged when the example scenarios were discussed, namely "what-if" analysis, selection of equipment, and reuse of materials. An important sub-activity is creating models for families of systems.

The development/adaptation of **control systems** generates the control system for the manufacturing system defined in the previous point. This is usually a very time consuming activity, and it is also error prone, which increases costs and set up times. The integration of legacy systems is an important problem that should be taken into account.

The **management of human resources** activity generates training programmes and defines operator skills. The successful development/adaptation of any manufacturing system is strongly dependent on the way the human resources are managed. A training program seems to be mandatory before any re-engineered system starts operating again. This is closely related to the skills definition of the people that will operate the system.

During the **implantation** phase the system is physically installed. The tests that must be performed before the operating phase are done here, using information available in the Systems Model & Knowledge base. The definition of operating procedures is also defined in the scope of this activity.

A characteristic that is common to all of the reengineering activities is **modelling**. As it could be seen in the previous points, there was a constant need for creation, modification, and adaptation of different kinds of models. The activity of modelling can be analysed according to two different views:

1. Which models to support the operating and reengineering phases, and
2. How to maintain and keep the models during the system life cycle.

If the shop floor is considered as the main focus, control models are important. They should be developed in a way that facilitate the construction of complex entities starting from basic components.

The same models that support the operation of the system should also support the maintenance of knowledge of that same system.

The life cycle support activities should be considered in the more general framework of enterprise engineering and reference architectures. Some of the most well known are GERAM (IFIP-IFAC, 1999), PERA (Williams, 1994), CIM-OSA (Williams et al., 1994), and GRAI-GIM (Doumeingts, Vallespir, & Chen, 1995) all of which deal with modelling and enterprise engineering aspects, although with different emphasis. For the purpose of this work GERAM (IFIP-IFAC, 1999) offers a generic framework that identifies and characterises the main entities and concepts in the enterprise life cycle. The main components of the GERAM framework are shown in Figure 2-19. The main areas related to this research work are shown in dark (Figure 2-19).

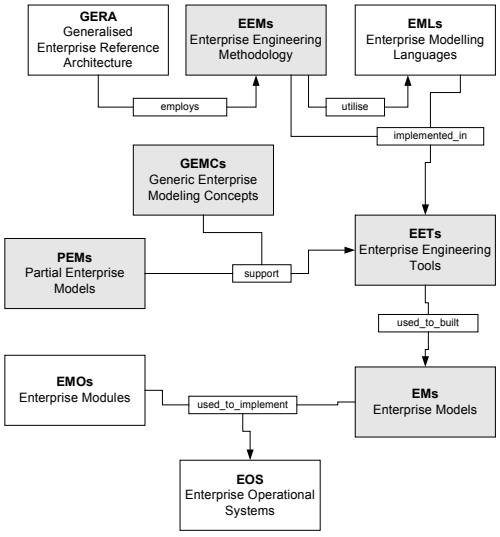

Figure 2-19 – GERAM Framework

2.4 CONCLUSIONS

Although the global reengineering process has been discussed above it should be noted that in this thesis the focus is limited to the process of change/adaptation of the control system. The work of the thesis would be enormous if it covered the whole process that has been mentioned. The idea behind the discussion that has been made is to give the reader the necessary background and framework to this work.

A more flexible, easier and faster generation of the control and supervision system will render the process of change faster and consequently more agile. This implies the development of a supporting architecture as well as an individual methodology to guide the process of creating the control solution.

The requirements to be satisfied by the agile shop floor control reengineering methodology are:

- **Requirement 1**. Departing from a manufacturing cell, with its set of manufacturing components, it should be possible to create dynamic control architectures in order to generate different functional structures.
- **Requirement 2**. The addition or removal of any manufacturing component should be done smoothly, with minimal programming effort or side effects in the others' components.
- **Requirement 3**. Legacy and heterogeneous controllers should be considered in the global architectures and a process should be found out to integrate them in the new agile architecture.

Note that the complexity of the implementation of an approach to satisfy these requirements is further increased because the shop floor is generally composed of various and heterogeneous controllers that must be integrated.

3 Supporting Concepts

The purpose of this chapter is to introduce the underlying concepts used to framework CoBASA. First, it provides the user with insight into the themes that have been used to structure and build CoBASA, which facilitates the reader's understanding of the architecture and the concepts behind it. Second, it states that the problem of shop floor reengineering, or more generically shop floor agility, can only be addressed by covering a large number of domains.

3.1 INTRODUCTION

This chapter describes some of the main concepts that in one way or another are relevant to the understanding of what is proposed in this thesis. The following concepts are discussed: multiagents, ontologies, contracts, shop floor control, collaborative organisations, and integrating infrastructures.

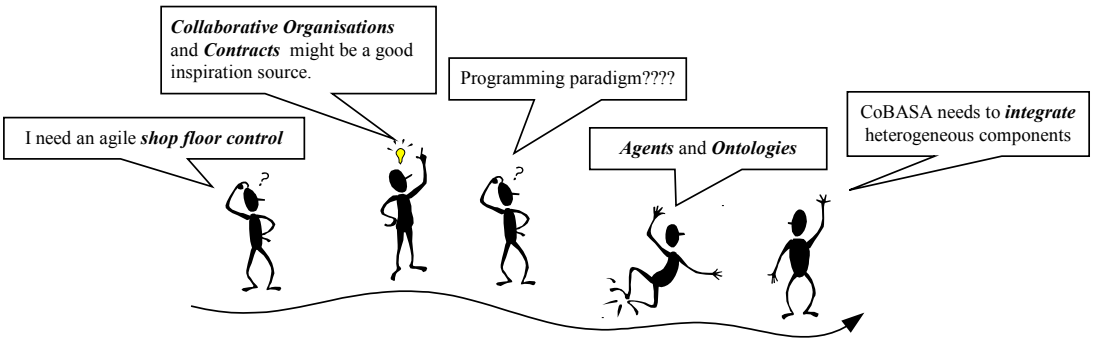

Figure 3-1 – Main supporting concepts

Collaborative organisations and contracts are included because they are the inspiration source used to build the most important CoBASA foundation concepts. Multiagents and ontologies are discussed

because they are the paradigms used to transform CoBASA from an abstract architecture into a computer based system. The multiagent paradigm is the programming model supporting the architecture, and ontologies are used to guarantee the common understanding among the agents. Finally, the integrating infrastructure is included because the shop floor environment is composed of heterogeneous entities that need to work together. Therefore, any reengineering proposal needs to take into account the problem of integrating these different entities in order they can cooperate.

3.2 MULTIAGENTS SYSTEMS

Multiagents systems represent a relatively new area in computer science, which started to be developed in the 1980s but that only in the mid 1990s gained widespread interest. Multiagent systems are compositions of computing elements that possess autonomous action, and which are able to interact among themselves, not only for exchanging messages but also for a more elaborated kind of communication that resembles social activity (cooperation, coordination, negotiation, …).

The multiagent area can be organised into two different sub areas. The first is concerned with the individual agent (micro level), while the second deals with the way these agents form societies of interacting agents (macro or group level). Using an analogy with humans beings, the first one is concerned with the individual human being, while the second one deals with the society in which humans live.

3.2.1 Individual Agents

3.2.1.1 The Agent Entity

Defining an agent is not consensual and many different definitions can be found reflecting the views of the different research communities that have been using the concept. While it is consensual that every agent should have some sort of autonomy, little agreement goes beyond this. For instance, while for some research communities the ability of the agent to learn is a fundamental condition, for others it is almost irrelevant. In this thesis, the adopted definition of agent is adapted from (Wooldridge, 2002) and (Vieira, 2000). This definition can also be compared to the definition of *software agents* in (Ferber, 1999).

Definition 3.1 - Agent

An agent is a computer-based program that is situated in some environment and that is able to take autonomous actions in this environment, based on its goals and perceptions from that environment, in order to meet its design objectives.

These programs must thus be able to interact with the environment (external world) through some actions and must be able to make decisions based on the perceptions they obtain from that world. Agents should never assume they have complete control or knowledge over the world (the world is not deterministic), and perceptions must be considered as giving only partial knowledge. Consequently, agents have to be prepared for the possibility of failure. In the context of this thesis the actions and perceptions are performed and felt through communication. Agents know about the world when they receive messages and *influence* the world by sending messages.

The characteristics that are usually considered important for an agent to be considered an intelligent agent in the framework of the multiagent area are (Camarinha-Matos & Vieira, 1999; Wooldridge & Jennings, 1995):

1. **Autonomy** – This is the most fundamental characteristic since it is the ability to make decisions under its own control, without external intervention, that essentially distinguishes an agent from other types of programs. An important characteristic of autonomy is the ability to keep making decisions even when the world changes.
2. **Reactivity**[17] – This is the capability of an agent to react, in due time, to changes in the world where it is immersed.
3. **Proactiveness** – An agent is proactive when it exhibits goal directed behaviour, i.e. it takes the *initiative* to pursue its design goals (deliberative activity). The agent activity is not limited to answering the stimulus from the environment but by its own initiative to pursue its goals.
4. **Social Ability** – This is the ability to interact with other agents, which can have completely different designs and objectives. It is much more than pure information exchange. The pursuing of the agent's objectives in an environment composed of different agents, which might have either cooperating or competing attitudes and with their own agendas and knowledge, requires *negotiation* and *cooperation*. This complex interaction between different heterogeneous agents can be the basis of a social emergent behaviour, which might be more complex than just the sum of the agents' individual behaviours (Axelrod, 1984, 1997; Johnson, 2001; Watt, 1996). Because human societies have been dealing with the same sort of problems, human societies and organisations become a source of inspiration to solve the agents' interaction problem. The area that deals with these topics is called ***social order*** or ***social intelligent systems*** (Castelfranchi, 1998; Conte & Dellarocas, 2001b), which will be further detailed in section 3.2.2.3.
5. **Adaptability** – The capability of the agent to adapt itself, within some limits, to the environment in which it is immersed.
6. **Continuity** – An agent should have a long-term existence on a continual basis instead of short term.

[17] Do not confuse reactivity with reactive agents, which corresponds to a kind of architecture to create agents that possess more reactivity than proactivity.

7. **Mobility** – Some agents are able to move from one environment to another while keeping their internal status and knowledge.

It is not mandatory that these characteristics appear simultaneously in all agents. In the context of this thesis, for instance, the most important characteristics are 1 to 4 and 6.

The right balance between *proactiveness* and *reactivity* is difficult to achieve, though fundamental to reaching the design goals of agents. A pure proactive robot can be built using, for instance, simply a plan to be blindly followed by the agent execution machine. In this case, the agent executes the plan without taking into consideration the environment. In the case of a controlled environment this can work quite well, as long as the agent has a sufficient model of the environment it is dealing with. However, modelling the environment is in general a difficult task and, consequently, the proactive approach alone can be disastrous since the agent might be acting on the basis of a completely outdated situation. Therefore, some kind of reactivity is needed in order for the agent to base the execution and even the refinement of its plan on the analysis of the current environment situation. On the other hand, developing a too reactive agent can also be troublesome, because creating an agent that reacts only to changes in its perception of the world can make it just a reactive entity that never achieves its goal. This is also what happens with humans when they just react to events and do not plan their activities. The level of *proactiveness* or *reactivity* depends on the tasks the agents are supposed to deal with.

The agent and object oriented paradigms are so often confused, when in fact they are different, that certain points need to be highlighted. People confuse both paradigms because an object is an encapsulation of some abstract concept, which can also be the case for an agent. Even though agent-oriented software engineering[18] (Jennings, 2001; Lind, 2001; Zambonelli, Jennings, Omicini, & Wooldridge, 2001) can be considered an extension of the object oriented programming, an agent is different from an object in the following aspects:

1. ***Higher degree of autonomy***. While objects cannot control when and which objects call their public methods, an agent has complete control over this. Agents decide by themselves when they can perform their actions.
2. ***Higher degree of proactiveness***. While some objects exhibit initiative to some goal, most of the objects are more passive than proactive. Agents, on the contrary, always exhibit some sort of deliberation (action towards their goals).
3. ***Social activity***. Objects were not thought to show social activity as it has been presented for agents. The cooperation they exhibit is limited to messages sent to other objects. It is possible, of course, to build complex interactions with these exchanges of messages, however, this was not in the basis of the object concept.

[18] The main purposes of Agent-Oriented Software Engineering are to create methodologies and tools that enable inexpensive development and maintenance of agent-based software.

4. ***Threaded controlled***. Each agent possesses its thread of control. While some object systems exhibit this property (active objects) and have, therefore, some degree of autonomy, this is not the usual situation.

Further details of this discussion can be found in (Lind, 2001; Tveit, 2001; Wooldridge, 2002).

3.2.1.2 Agent Architectures

The agent architectures can be classified as (Wooldridge & Jennings, 1994): **deliberative**, **reactive**, and **hybrid**.

Deliberative agents. Under this paradigm agents reason and make decisions based on the symbolic representations (model) they have of the external world. Deliberative[19] agents, which have precise goals to be achieved by their actions, need a lot of effort to symbolically represent the complex entities of the external world. On the other hand, the amount of computation these models require in order to make appropriate decisions can be enormous. The vast work on, for instance, knowledge representation, reasoning, and planning from the Artificial Intelligence (AI) area has its origins in this need to represent the world and reasoning about it.

The most well known architectures associated with deliberative agents are:

1. Planning agents
2. The Belief-Desire-Intention (BDI) architecture
3. Epistemic-Deontic-Axiologic (EDA) architecture.

The decisions of deliberative agents rely on planners, which makes planning an important activity of traditional AI agents. In general, a planner is a program that generates other programs or sequences of instructions that represent actions. STRIPS (Fikes & Nilsson, 1971, 1990), a linear planner, is perhaps the most well known because it was the first and it is simple to understand. Later on, hierarchical planning (Knoblock, 1992; Sacerdoti, 1975) as well as partial order planners (Chapman, 1990; Sacerdoti, 1977) were introduced to overcome the weaknesses of linear planners.

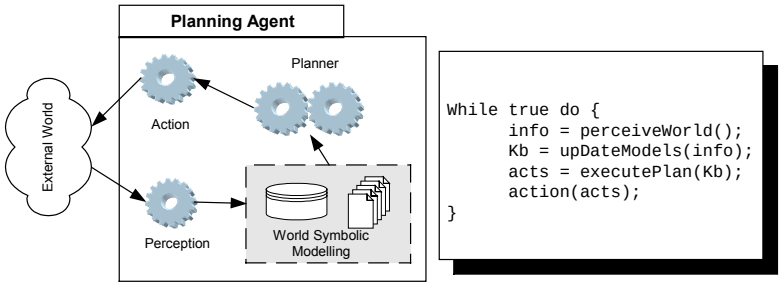

Figure 3-2 – Plannning agent architecture and control algorithm

[19] Genesereth introduced this term (Genesereth & Nilsson, 1987).

A simplified architecture of planning agents is shown in Figure 3-2, which is adapted from (Vieira, 2000).

The BDI architecture has its grounds in the idea that to handle complex problems it is better to create agents whose reasoning is analogous to human practical reasoning[20]. BDI combines philosophy, software architecture, and logic (Wooldridge, 2000), and is based on the assumption that agents maintain and reason about internal representations of their world (cognition). *Rational action* (Bratman, 1987) is the philosophical theory that supports BDI, which is specifically focused on the role that *intentions* play when deciding which action should be performed.

The basic operations of BDI agents are (Georgeff, Pell, Pollack, Tambe, & Wooldridge, 1999; Rao & Georgeff, 1991, 1995; Wooldridge, 2000):

- **Beliefs** – that represent the knowledge about the agent and the external world.
- **Desires** – that represent the objectives to be accomplished or preferences over future world states or course of actions. A desire might be regarded as objectives without a plan, i.e. a long-term goal that the agent does not know yet how to achieve.
- **Intentions** – that represent the currently chosen course of action. Intentions[21] are chosen from the set of desires and represent committed goals. An intention might be regarded as objectives with a plan, i.e. a short-term goal that the agent knows how to reach it.

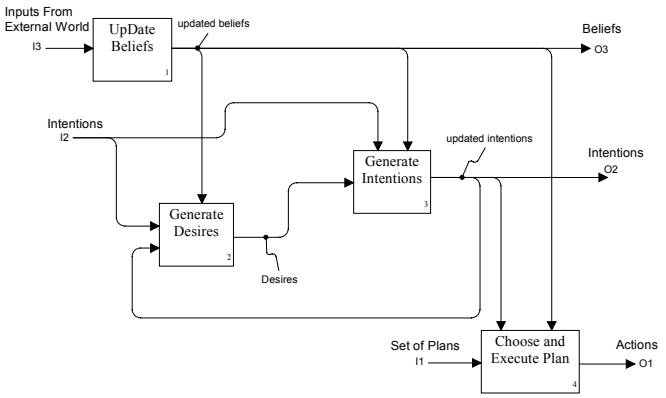

Figure 3-3 – A simplified IDEF0 diagram for the BDI architecture

```
While true do {
      info = perceiveWorld();
      B = upDatebeliefs(info);
      D = generateDesires(B,I);
      I = generateIntentions(B,D,I);
      plan = choosePlan(B,I);
      executePlan(plan);
}
```

Figure 3-4 – Generic algorithm for a BDI agent – adapted from (Wooldridge, 2000)

Each BDI agent has a set of plans that represent the actions that must be performed to reach an intention. The choice of which plan should be executed for a given intention is based on the agent current intentions and beliefs. This means that an intention might be achieved using different plans. Beliefs can be represented by a variable, a database, or symbolic expressions in some logic. It is quite common in BDI agents to find beliefs represented using the Prolog language (Bratko, 2001) facts.

[20] Considering that humans are rational, which can be a subject highly controversial.
[21] Bratman (Bratman, 1987) considered that intentions play a much stronger role in influencing action than desires.

Since desires and intentions are end states, they can also be represented by variables, record structures, or symbolic expressions.

Figure 3-4 shows the generic algorithm for a BDI agent. The agent needs a module to perceive the world and to transform the perceptions into beliefs. Next, desires are generated from the current beliefs and intentions, and then intentions are updated based on the revised desires, beliefs, and intentions. After that, the best plan is chosen and the plan executor module executes it.

In the way to achieve an intention new desires are created and, consequently, new intentions. This might be disadvantageous if the agent does not guarantee that each intention persists until it is achieved. In this case the agent can just be jumping from intention to intention without focusing. However, too stubborn agents might also be inappropriate as they might be insisting on a plan that is no longer valid. The algorithm shown in Figure 3-4 keeps the intention and the chosen plan while it is not finished. To implement an open-minded commitment the algorithm should be changed to the one indicated in Figure 3-5.

```
While true do {
  info = perceiveWorld();
  B = upDatebeliefs(info);
  D = generateDesires(B,I);
  I = generateIntentions(B,D,I);
  plan = choosePlan(B,I);
  while (not empty(plan) and possible(I,B)) {
    action = first(plan);
    execute(action);
    plan = other(plan);
    info = perceiveWorld();
    B = upDatebeliefs(info);
    if not sound(plan,I,B) then
       plan = choosePlan(B,I);
  }
}
```

Figure 3-5 - Algorithm for a BDI agent open-minded committed - adapted from (Wooldridge, 2000)

The function *sound(plan,B,I)* is true while the plan *plan* can reach the intention *I* given the current beliefs *B*. The function *possible(I,B)* is true when the intention I is still possible given beliefs *B*. The function *first(plan)* returns the first action of the plan, while the function *other(plan)* returns the rest of the actions. In the algorithm an intention is maintained while its goal is possible and the plan is not finished. On the other hand, whenever a plan for a certain intention is not sound due to beliefs revision, a new plan is chosen.

Therefore, a formalism is required to represent the complexity behind the BDI architecture. To model beliefs, desires, and intentions first order logic is used, as well as logics based on modal and temporal operators (Barwise & Etchemendy, 1992; Dean, 1997; Gabbay, Finger, & Reynolds, 2000; Girle, 2000). In fact the logic behind BDI, as defined by Rao and Georgeff (Rao & Georgeff, 1995), is a multimodal logic composed of three modal operators: Beliefs (B), Goals (G), and Intentions (I). These theoretical frameworks have, however, some drawbacks, namely when an implementation is needed. The main problem is how to find a suitable language that can reasonably model the logic

behind BDI. An example of a language developed specifically for BDI is AgentSpeak(L) (Rao, 1996), which is based on a restricted first-order language with events and actions, making it very similar to the traditional logic programming. AgentSpeak(L), as introduced by Rao, was an abstract interpreter and no real implementation of it had been done until recently when Machado and Bordini (Machado & Bordini, 2002) implemented the very first interpreter.

An important concept behind any BDI architecture is planning. However, BDI is characterised not by having one generic plan, but several pre-assembled plans – a plan library. The best-fitted plan to achieve the persecuted intention is chosen. It is assumed that the BDI agent has the set of all required plans for the intentions it is supposed to achieve, even if some of the information required for these plans is not yet fully fulfilled. This means that some kind of plan detailing actions are required whenever a plan is chosen since the plan is just a pre-assembled sequence of actions that need to be finished. Each plan is composed of actions, the pre-conditions that express what facts must be true (beliefs) for the plan to be executed, and the post-conditions that include the set of predicates that represent the achievement of an intention.

Although the BDI architecture seems to be an adequate framework to implement complex agents, some disadvantages must be taken into account:

- It is complex and difficult to program, since the formal model behind it is complex.
- It is difficult to find the right balance between the levels of proactivity and reactivity.
- They are not well suited for some types of agent behaviours, namely those that must learn and adapt their behaviour during the agent's life (Georgeff et al., 1999).
- It is difficult to define, in a user-friendly way, how intentions and desires are represented (modelled). Even not so complex beliefs can be difficult to model. Another source of complexity is how to cope with antagonic intentions and desires as well as inconsistent beliefs.
- It is difficult to choose the right intention because, many times, there are many that can be suitable.
- Lack of suitable development tools. Much of the current available agent development tools do not implement, in an easy way, the BDI model.
- Poor architecture to explicitly represent multiagent aspects of behaviour. The BDI model is inherently motivated by mental attitudes, which can be an important advantage in terms of the proactiveness of the agents when considered as individuals. However agents have a strong social nature, especially when the individual agents have to cooperate with others, creating agent societies. In this case the BDI model is poorly indicated to model the social behaviour each agent must have because social aspects are not easy to model into beliefs, desires, and intentions. Nevertheless some work has started to be done to overcome from this limitation (P. Panzarasa & Nicholas R. Jennings, 2002; Panzarasa, Jennings, & Norman, 2001).

Nevertheless, there are some successful implementations of BDI agents as, for instance:

- PRS - Procedural Reasoning System (Georgeff & Ingrand, 1989; Georgeff & Lansky, 1987) - It was the first agent architecture to implement the BDI model, and is so disseminated that is sometimes known as the BDI architecture
- DMARS (d'Inverno, Kinny, Luck, & Wooldridge, 1997) – It is an evolution of the PRS architecture.
- IRMA - Intelligent Resource-bounded Machine Architecture (Bratman, Israel, & Pollack, 1988) – It is another variant of PRS. Besides data structures for beliefs, desires, and intentions, it includes a plan library.

The EDA - Epistemic – Deontic – Axiologic architecture is based on the theory of norms and ontological, epistemic, deontic, and axiological attitudes (Filipe, 2000; Filipe & Liu, 2001). This approach can be considered a response to the lack of "sociality" in the BDI architecture. Instead of focusing on the design of the individual agent's mental states, EDA concentrates on *norms*[22] (social constraints) to govern the cooperation among the agents. In spite of being more oriented towards societies of agents the EDA architecture shares many of the BDI concepts.

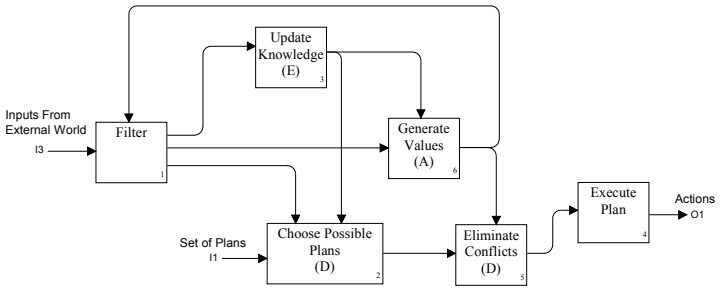

Figure 3-6 – The EDA architecture. Adapted from (Filipe, 2003)

The EDA architecture is composed of the following components: epistemic (E), deontic (D), and axiological (A). The epistemic component represents knowledge about the world, i.e the agent's beliefs. This is similar to the BDI architecture. The axiological component represents values under which the agent operates. This is new when compared to the BDI architecture. The deontic component contains the behaviour (plans). In addition it also contains the rights and obligations.

Figure 3-6 shows how a basic EDA agent operates using an IDEF0 diagram. The letters A - axiological, D – deontic, and E – epistemic in the activity boxes indicate the components to which the activity belongs. The set of actions to be executed, which are represented by a plan (agent behaviour), is chosen from the set of possible actions (plans) the agent can perform, based on the epistemic state of the agent and the axiological norms.

An EDA agent needs to filter the information acquired by the sensors. After this information is transformed into a suitable form for the agent, it is filtered according to the values of the agent

[22] A norm is a kind of ideal behaviour, or recommended behaviour. It is different from a rule because a rule imposes an obligation. Rules are thus enforced norms.

(axiological component) and delivered to the axiological, deontic, and epistemic components. The values of the axiological component constrain the filter and help in deciding which plan should be chosen when more than one plan is found as appropriate (conflict of plans). The knowledge component is used to constrain the generation of new values as well as to restrain the behaviour of the agent (Deontic component). To represent the knowledge associated with the epistemic component, first order logic or modal logics might be used.

Choosing which behaviour should be followed is also similar to the BDI means-end reasoning[23]. The specific point of EDA is that obligations and rights are also used to influence which plan will be chosen. The axiologic aspects are best represented using non-monotonic logics (Ginsberg, 1987), which is the kind of inference in which conclusions are obtained tentatively, reserving the right to retract them whenever further information is available. The interesting fact about this logic is that the set of conclusions that can be obtained from the available knowledge might shrink with an increase in the available knowledge, instead of increasing as happens with traditional logic. This, in fact, is important because the values of the agent might be reduced even if in the presence of more available knowledge.

To represent the normative aspects of the behaviour of the agent, deontic logic is used (McNamara & Prakken, 1999), which is a kind of modal logic for obligations, permissions, and prohibition. The use of deontic logic is quite natural since obligations are an integral part of an EDA agent. In addition deontic logic is indicated to:

- Distinguish between obligation and necessity.
- Express general and special obligations.
- Express what must be done (sanction) when a rule is violated, which are characteristics that normative agents must possess.

Although the EDA seems to be a promising architecture as it seems indicated to be used in complex problems that need social behaviour, there are some disadvantages:

1. EDA architectures are very complex
2. No suitable framework exists for development
3. There is a big gap between the theoretical framework, namely deontic logic, and practical implementations
4. Few implementations, although some work is being done (Filipe, 2002a, 2002b, 2003).

Reactive agents. The agent architectures that have been introduced up to now use symbolic representations to represent the world and to do planning. These knowledge representations are modelled using different kinds of logic that can be difficult to implement. On the other hand, in these architectures, it is assumed that the world is known in advance, or at least as much of it as is required. If the world is not known in advance, the complexity of the planning system is very high because the

[23] This is the process of deciding how to achieve an intention using available means. In other words this is planning.

planner must be highly generic to cope with the different situations that might occur. In the current state of the art these highly generic planners are not available. This difficulty has been particularly relevant in the case of the agents used in mobile robots that typically work in unstructured environments. On the other hand, there is an increasing tendency towards cheaper and simpler mobile robots, which requires simpler computational structures. Therefore, a new approach was developed – reactive agents. The ideas behind this paradigm are:

- Intelligent behaviour does not require explicit representations
- Intelligent behaviour does not require abstract (symbolic) reasoning
- Intelligence is a product of the interaction the agent maintains with its environment
- Intelligence is an emergent property of certain complex systems.

Reactive agents are also known as behavioural agents because they suppose the existence of basic behaviours or sequences of actions that execute concurrently to form the lowest level of intelligence. These behaviours are, in turn, used by more complex ones to create more complex levels of intelligence.

The most well known reactive architecture is the **subsumption** architecture that was proposed by Rodney Brooks in (Brooks, 1986). Eventually other articles were produced that synthesise the thought of Brooks in terms of this new approach to dealing with artificial intelligence (Brooks, 1991a, 1991b, 1999, 2002).

❑ No explicit knowledge representation is used (Brooks, 1991a). Brooks often refers to this, as "*the world is its own best model*". ❑ Response to stimuli is reflexive - the perception-action sequence is not modulated by cognitive deliberation. ❑ The behaviours are organised in a bottom-up fashion. Thus, complex behaviours are composed of the combination of the simpler ones.	❑ Behaviour is distributed rather than centralized. ❑ Individual agents are inexpensive, allowing a domain to be populated by many simple agents rather than a few complex ones. These simple agents individually consume few resources (such as power) and are expendable, making the investment in each agent minimal. This synthesises the vision Brooks has for mobile agents or creatures (Brooks, 2002).

Table 3-1 – Assumptions of the subsumption architecture

The subsumption architecture is characterised by a set of programs/behaviours (finite state machines or rules *if then else*) that describe the independent tasks each agent must perform, and which are hierarchically organised. Each layer includes a set of pre-wired behaviours. The higher levels build upon the lower levels to create more complex behaviours. The behaviour of the system as a whole is the result of many interacting simple behaviours. An important aspect is that these behaviours operate asynchronously.

To clarify better how the subsumption architecture operates, an example of a mobile robot that goes from a start position to a destination point with some obstacles in its way is considered (Figure 3-7). The robot knows the direction of the destination place because a sensor in the robot can detect the emission of a beacon. The control logic of this sensor gives, when requested, the azimuth of the beacon in relation to the longitudinal line of the robot. Since the destination point is marked by a dark area, which can be sensed by another sensor of the robot, it is possible to know when the destination point is reached. The robot also has sensors to detect obstacles.

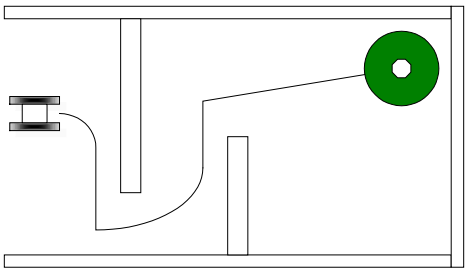

Figure 3-7 – Mobile robot using a subsumption architecture

This robot is composed of three hierarchical behaviours that run concurrently and asynchronously (Figure 3-8). The higher-level task is *GoalOriented*, which guides the robot to its destination. This task is continuously running and, for each interaction, the azimuth sensor is read and a function to align the robot with the read azimuth is called. Simultaneously, the behaviour *AvoidObstacles* is also running as well as the lowest level one *Stop*. If an obstacle is found while the robot is moving, a racing condition happens because *GoalOriented* tries to keep the alignment with the beacon while *AvoidObstacles* try to avoid it and the actions might be in conflict. Under this condition the subsumption architecture determines that the low level behaviour has precedence over the higher level one. This is why the behaviour *AvoidObstacles* inhibits the higher level one. After the obstacle avoidance, the higher-level behaviour is free to run again (instruction *run(GoalOriented)*). When the robot reaches the destination point, the race is between the three behaviours. In this case the lower level behaviour (Stop) takes precedence and the robot is stopped.

Brooks proved the subsumption architecture in the field of mobile robots, by developing several robots that achieved impressive goals when compared to traditional AI. Steels also developed similar work (Steels, 1990). PENGI is another example of a reactive agent architecture, which was developed by Agre and Chapman (Agre & Chapman, 1987). This architecture is based on the idea that most of the decisions made by agents are routine (require little abstract reasoning). Another relevant architecture of reactive agents is *situated automata* by Rosenschein and Kaebling (Kaelbling & Rosenschein, 1990; Rosenschein & Kaelbling, 1996). It follows the similar idea of having simpler behaviours without knowledge representation.

```
Behaviour GoalOriented {
 int azimuth;
 azymuth = readBeaconSensor();
 alignRobot(azimuth);
}
```

```
Behaviour AvoidObstacles {
  boolean obstacles;
  obstacles = readObstSensor();
  if  (obstacles) {
         inhibit(GoalOriented);
         avoidObstacle();
         run(GoalOriented);
     }
}
```

```
Behaviour Stop {
    boolean stopV = readStopSensor();
    if (stopV) {
         inhibit(allBehaviours);
         stopMotors();
    }
}
```

Figure 3-8 – The behaviours of the mobile robot.

Although reactive agents are simple, can be elegantly designed, and are economical, they have some disadvantages:

1. Since the world is not modelled, more sensorial information is needed for agent's decision making.
2. Sensors only provide local and immediate information. The absence of a world model avoids any previewing of what is going to happen in the future.
3. This approach does not accommodate the possibility of agents to evolve through learning.
4. The complex behaviour of reactive agents is emergent (based on the interactions of simple agents), which is very difficult to model and to preview. Consequently, there is no available methodology that can help the creation of a predictable emergent behaviour.
5. When the number of behaviours increases it is very difficult to understand all the involved interactions. This is in part related to the problem of emergence.

Layered architectures – Hybrid Agents. These architectures are simultaneously reactive and deliberative. They contain a set of interacting layers in which some are deliberative and others are reactive. The number of layers varies but at least there should be two: one reactive and another deliberative or pro active. Depending on how the sensorial information is delivered to the layers, and how the layers interact, two types can be identified: horizontal layering, and vertical layering (Wooldridge, 2002). An example of a horizontal layering architecture is ***TouringMachines*** (Ferguson, 1992, 1995), while ***InteRRap*** (Muller & Pischel, 1993) is an example of a vertical layering.

In horizontal layering (Figure 3-9) the sensors are directly connected to each existing layer, which also might drive the outputs directly. Each layer acts like an individual agent that gives suggestions to the output (actuators), what can be dangerous in terms of coherence because more than one layer might want to control the same actuators. It is thus necessary to create a way to avoid these *race conditions* over the actuators. Horizontal layering architectures are easier to design because of the

clear separation between each layer. However, the synchronisation in a race condition situation can be difficult to implement even for simple agents. This difficulty led to the development of the vertical layering architecture.

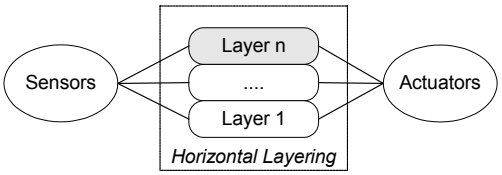

Figure 3-9 – Horizontal layering

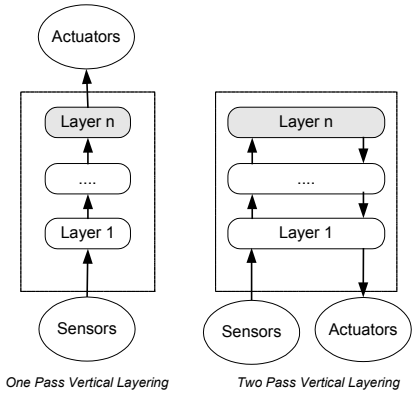

Figure 3-10 - Vertical layering

In vertical layering (Figure 3-10) the sensorial information instead of being connected to each layer is connected to the lower level layer and then this layer passes the necessary information to the next upper level and so on. Each layer uses the information it receives from its lower level layer and passes to its upper level layers only the information this layer needs. The information flows sequentially from the lower level (received sensorial information) to the upper level layer, which is the one that activates the actuators. This is one pass vertical layering. If, on the other hand, the upper level does not activate the actuators, but instead the flow is fired back to the lower levels, it reaches the lowest level layer, which activates the actuators. In this situation the architecture is called two-pass vertical layering. In both situations there are no race conditions since only one layer activates the actuators.

When comparing one horizontal layering to vertical layering architectures the most important difference is that horizontal layering is more complex to implement (due to the race condition between the layers) although more flexible (layers are much more independent) and more robust. Vertical layering, on the other hand, is simpler to implement, but less flexible and robust. When one layer, for instance, has a problem, the whole system cannot work.

3.2.2 MultiAgents

The focus of this section moves from the architecture of the individual agents to the architecture that a group of agents create or form. Individual agents are useless in the large majority of situations, because most of the scenarios involve several interacting agents. When dealing with multiagents the important aspects are now how they communicate and how they interact. Communication and interaction are the mechanisms that let the community of agents have a more complex behaviour than just the sum of their individual behaviours, i.e. a form of emergent behaviour.

3.2.2.1 Coordination

In any society of agents **coordination** (Figure 3-11) is one of the most important aspects, especially because it is likely that any agent will share some goals and interests with the other members of the society and, consequently, coordination is needed to manage the dependencies between activities of multiple agents. Coordination governs interactions rather than simply enabling them. Several works support this idea (Omicini, Zambonelli, Klusch, & Tolksdorf, 2001). An individual agent in a multiagent system may perform its actions devoted to, for example, coalition formation, team building, distribution of tasks, etc. via bilateral and multilateral interactions with other agents. In the same direction (Agha, Jamali, & Varela, 2001) state that coordination fills the gap between autonomously acting agents and the problems they are collectively solving.

Coordination is required at different levels (Papadopoulos, 2001):

- The low level, which deals with the basic infrastructure.
- The middle level, which offers modelling mechanisms to support high-level coordination..
- The high-level or logical coordination, which includes the coordination at such a high semantic level that, instead of pure coordination, it makes more sense to speak about cooperation or competition.

Low-level coordination. Coordination can only happen if some kind of communication mechanism exists. Therefore, agent communication languages (ACL) are used as a basic mechanism to support agent coordination or interaction. Another basic mechanism to support multiagent coordination is the development platforms used to create/generate multiagent systems. This is so because the kind of coordination mechanisms these environments have embedded strongly impact the way agents are designed and implemented, namely the type of coordination that is possible to achieve within the agent society. For instance, the existence of traditional concurrent programming mechanisms such as semaphores, distributed shared memory, message-passing mechanisms (mailboxes), Remote Procedure Calls (RPCs), distributed shared objects, etc., embedded in the development environment allow more robust and better designed agents. In fact the greater the number of high level concurrent mechanisms hidden in the development infrastructure the more the agent designer can concentrate on higher-level coordination.

Middle-level coordination. Emphasis at this level is put on genuine coordination models and languages. Architectures such as CORBA (CORBA, 2001), DCOM (DCOM, 2001), and Java RMI (RMI, 2001) are important because they already implement some sort of coordination mechanisms and allow the implementation of more complex ones. Another middle level mechanism relevant for agent interaction is tuple-based approaches (Rossi, Cabri, & Denti, 2001), which use associative access to shared data spaces for communication and/or synchronisation purposes. Relevant environments, which

are based on Linda-like[24] shared spaces mechanisms (Banville, 1996; Carriero & Gelernter, 1986, 1990; Feng, Gao, & Yuen, 1995; Gelernter & Carriero, 1992; Pinakis, 1991; Rowstron, Douglas, & Wood, 1995) are TuCSoN (Cremonini, Omicini, & Zambonelli, 2000), KLAIM (De Nicola, Ferrari, & Pugliese, 1998), Berlinda (Tolksdorf, 1997), PageSpace (Ciancarini, Tolksdorf, Vitali, Rossi, & Knoche, 1998), T-Spaces (Wyckoff, McLaughry, Lehman, & Ford, 1998), JavaSpaces (Freeman, Hupfer, & Arnold, 1999), and MARS (Cabri, Leonardi, & Zambonelli, 2000).

High-level coordination. The very first important aspect to discuss here is whether coordination is assisted or non-assisted. In non-assisted coordination the agents communicate directly with the agents with whom they want to interact (ccordinate). On the other hand, in assisted coordination the agents rely on other agents, known as middle agents, also known as brokers, facilitators, matchmakers, etc., to achieve coordination.

Assisted coordination is currently very important in the digital economy because middle agents can be considered electronic intermediaries. Middle agents, as defined in (Klusch & Sycara, 2001), are characterised as 1) providing basic mediation services to the agent society, 2) coordinating these services, and 3) ensuring reliable service mediation. An extensive survey on middle agents and brokering can be found in (Klusch & Sycara, 2001).

Despite the differences in terms of whether the interaction is direct or not, one thing is common to both types: the interaction is achieved through the engagement of one or more agents in a conversation, which is a pattern of message exchanges that two or more agents agree to follow when communicating. Some researchers (Cost, Labrou, & Finin, 2001) consider conversations as a pre-arranged coordination protocol. In both cases (assisted and non assisted coordination) the agents may follow fixed or emerging rules, social obligations, and styles of conversation to coordinate their activities. Figure 3-11 summarises a variety of engagements, which correspond to different approaches for coordination.

Cooperation is coordination among agents that share the same goals, while competition is coordination among self-interested agents (Huhns & Stephens, 1999). It is important to note that interaction protocols are the mechanisms that govern coordination (both cooperation and competition).

Cooperating to achieve a certain task or goal consists of two phases. In the first phase the task is decomposed into smaller subtasks and then in the next phase it is necessary to distribute these subtasks among the cooperating entities. Several decomposition plans can be generated according to the available resources and the capabilities of the cooperating entities. The decomposition can be either

[24] Linda uses a data space with associative access. There is a set of coordinating operations combined with a tuple space from where tuples can be written and retrieved. It is out of the scope of this section to provide more details about it and its shared spaces. Further information can be found in the references as well as in (Omicini et al., 2001). It should be noted that shared data spaces have their origin in the *blackboards* (Englemore & Morgan, 1988) (information spaces where information can be put and retrieved by multiple agents).

manual, during the design phase or automatic, using planning systems. The distribution of tasks among agents is a different problem and several strategies exist to deal with it:

1. Planning systems – This is the case when there are agents that plan all current and future interactions among themselves to avoid inconsistent or conflicting actions. In centralized planning a coordinating agent is responsible for the plan generation and its execution. In distributed planning each agent is aware of the plans of the other agents and acts accordingly.
2. Organisational structures –In this case the distribution is hardwired at design time. It provides a framework for activity and interaction through the definition of roles, communication paths and authority relationships. The *blackboard* coordination structure (Englemore & Morgan, 1988) is an example of an organisational structure.
3. Contracting – In this situation there is a manager agent that divides the problem into sub problems and the distribution is done using a kind of contracting approach. This is known as the *contract-net protocol* (Smith, 1980). In this protocol an agent issues a request for bids using multicast or selective cast and then waits for answers. The requester then evaluates the proposals and after deciding which is the agent with the best proposal, awards the contract to that agent.
4. Norms and Social Laws – In this situation the cooperation (coordination) is achieved by each agent following an expected or established pattern of behaviour, which are known as *norms* (Lewis, 1969), are developed inside the society and are expected to be followed by each of its members.

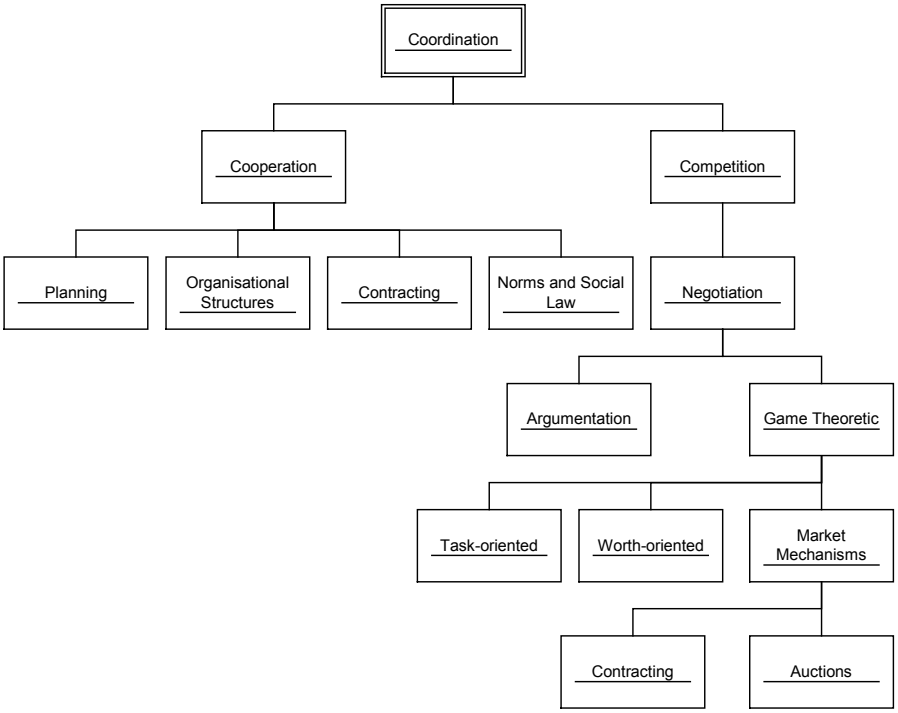

Figure 3-11 - Taxonomy of coordination. Adapted from (Huhns & Stephens, 1999)

Competition, as stated before, happens when a group of agents with different goals need to interact. When these self-interested agents need to reach agreements of mutual interest this process is called negotiation[25] (Lomuscio, Wooldridge, & Jennings, 2001). Agents first inform of their positions and then try to reach an agreement by making concessions or searching for alternatives. The proposals that each agent makes are defined by its strategy, and must follow a protocol. An important domain where negotiations are relevant is the area of electronic commerce where much work is currently going on. Negotiation thus appears frequently associated with words such as e-Commerce, e-Business, e-Contract, etc. Further details about the use of automated negotiations in this area can be found in (Camarinha-Matos, 2002b; Camarinha-Matos, Afsarmanesh, & Rabelo, 2001; Cortés & Dignum, 2001; Dignum & Sierra, 2001; Padget, Shehory, Parkes, Sadeh, & Walsh, 2002).

Negotiation theory covers different aspects, appears in a multitude of forms, and uses many different approaches from different scientific areas such as artificial intelligence, anthropology, social psychology, sociology, political sciences, economy, and game theory. Negotiation is divided according to two different approaches: game theoretic and argumentation. The first approach is based on concepts from the game theory, where aspects such as a utility function and the best strategy to follow when agents encounter are relevant (Wooldridge, 2002). Game theory is not concerned with how alternatives and utility values are obtained but with what alternatives should be selected in the case of multiple decision makers. Argumentation, on the other hand, is an answer to the disadvantages faced by game-theoretic negotiation approaches (Kraus, Sycara, & Evenchik, 1998; Parsons, Sierra, & Jennings, 1998; Sycara, 1989, 1990; Zeng & Sycara, 1996).

The most important types of negotiation using game-theoretic approaches are:

- Market mechanisms – they implement strategies usually used in markets or businesses. This approach is divided into contracting and auctions. The best known protocol for contracting is *contract net* (Smith, 1980), which was already mentioned. Nowadays, auctions are an active research theme strongly used for negotiation in electronic commerce (Kersten, Noronha, & Teich, 2000; Lomuscio et al., 2001; Strobel, 2000, 2001). In general the goal of an auction is for the auctioneer (the entity that allocates the goods or service) to maximise the price at which the goods will be sold, while bidders (the entities that want the goods or services) want to minimise it. The most important types of auctions are: English, Dutch, First-price sealed-bid, and Vickrey (Wooldridge, 2002).
- Task-oriented (Rosenschein & Zlotkin, 1994) – Although the agents might have all the resources and skills to achieve their tasks without the help or interference from each other, they can benefit from sharing some of the tasks. The agent tasks are explicitly defined as well as their cost, which the agent tries to minimise.
- Worth-oriented – In this case the negotiation occurs to maximise the worth of the agent. The various states that can be reached by the agent are quantified by a worth function. The agent

[25] Some authors consider that negotiations occurs both when agents are competing and cooperating.

uses this worth function to negotiate with the other agents to bring about the state with the greatest value.

Game-theoretic approaches allow the implementation of quite simple successful negotiation protocols and are very well suited for conflict resolution. They are effective in cases where it is possible to characterise the preferences and possible strategies of the negotiation participants. However, there are some disadvantages as pointed out by (Jennings et al., 2001):

- It is difficult to characterise complex preferences that are not easily represented by a utilitarian function.
- There is no generic model governing rational choice.
- This approach assumes that each agent's choice can be uniquely described by a utility function (perfect computational rationality).
- The agents need large computational capabilities to analyse the various alternatives.
- The utility function is assumed to be fixed during the negotiation process, which reduces its capability to model real negotiations since humans, in general, change their preferences during the negotiation process. Consequently, agents have strict negotiation protocols.

In argumentation based negotiations an agent tries to convince the other or others about its viewpoints. This is done by exchanging propositions (arguments), with justifications, as to why the arguments are being accepted or rejected. Logical models are thus the framework under which argumentation relies on. Kraus defines argumentation in (Kraus, 2002) as:

> *"Argumentation is the means by which an agent, the persuader, attempts to modify the intention structure of another agent, the persuadee, to include the actions the persuader wants it to do. While an agent tries to influence the intentions of other agents, other agents may try to convince it as well. The role of persuader and persuadee is not fixed, but dynamically assumed during the agent interactions. Thus, during a negotiation process, each agent may update its intentions and goals after receiving a message from another agent."*

The most well known implemented argumentation-based negotiation system is PERSUADER, which was developed by Katia Sycara (Sycara, 1990, 1992) and operates in the domain of labour negotiation.

3.2.2.2 Communication – Agent Communication Languages - ACLs

The study of the animal world (humans included) has shown that the most complex, long-lived communities are those that have developed complex communication mechanisms that allow the establishment of complex interactions. Human language is, in fact, one of the most complex communication mechanisms and, unsurprisingly, humans were able to develop the most complex societies in the animal world. Considering these aspects, it seems unquestionable that complex agent societies can only be created with the aid of Agent Communication Languages (ACL) that, through

interaction, allow the creation of agent communities that tackle problems an individual agent could never handle. ACLs should be public to permit the conversation between a large number of agents.

Agents engaged in negotiations need ACLs that support more than just single message exchanges. These negotiations are supported by agent engagement in *conversations* that are task-oriented, shared sequences of messages that they follow. These conversations are also known as pre-arranged coordination *protocols*. Well-defined and sharable conversation protocols can be used to coordinate agents that attempt to accomplish specific tasks (coordination view). Typically, a conversation protocol is associated with a specific task, such as registration, or a particular type of negotiation. CoBASA agents, for instance, must engage in several conversation protocols, such as joining a coalition or adhering to a group of agents.

The idea behind any communication process among agents is the exchange of information and knowledge. But if that was the only objective of ACLs they would not be needed because computer science has already developed mechanisms that permit the exchange of information and knowledge, as is the case with sockets, remote method invocation such as RPCs (Corbin, 1990) and JAVA RMI (Grosso, 2002; RMI, 2001), CORBA (CORBA, 2001; Siegel, 2000), and DCOM (DCOM, 2001). Therefore, what differentiates ACLs from these mechanisms resides in the objects of discourse and their semantic complexity. While the traditional communication mechanisms exchange simple objects with no semantics associated with them, ACLs exchange propositions, rules, and actions. Propositions should be put into context by describing the propositional attitude under which those propositions should be considered. The propositional attitude represents the meaning of exchanges that are characterised by communicative acts, which are based on the *Speech Act Theory*. This theory treats communication as action. Acts of speech (communication) are treated as actions that change the internal state of the agent. This theory is considered to have been initiated by John Austin (Austin, 1955), who identified a number of *performative* verbs that correspond to various types of speech acts. Later on, Austin's work was extended by John Searle (Searle, 1969), who identified several properties that must hold for a speech act performed between a hearer and a speaker to succeed. The most important categories for communicative acts are, as indicated in (Wooldridge, 2002):

1. ***Representatives*** – The paradigm case is *informing*. This commits the speaker to the truth of an expressed proposition.
2. ***Directives*** – The paradigm case is *requesting*. The speaker attempts to get the hearer to do something.
3. ***Commissives*** – The paradigm case is *promising*. The purpose is to commit the speaker to a course of action.
4. ***Expressives*** – The paradigm case is *thanking*. They express emotions and evaluations.
5. ***Declaratives*** – The paradigm case is *declaring*. They cause events in themselves.

Cohen and Levesque adopted communicative acts to the agent world when they developed a theory in which these acts were modelled as actions performed by rational agents (Cohen & Levesque, 1990).

Agent communication languages can also be regarded as a matter of knowledge sharing because agents exchange knowledge. This is the reason why some ACLs have their origin in the knowledge sharing research area. The Knowledge Sharing Effort (KSE), for instance, was initiated in 1990 by the DARPA agency in the USA (Neches et al., 1991; Patil et al., 1998) with the goal to develop techniques, methodologies, and software tools for knowledge sharing.

Three levels can be considered in a situation of knowledge sharing or communication between agents:

1. Syntax level that describes how the internal models (knowledge) are organised and described. When agents, using one language to represent their models, need to communicate with agents that follow a different language it would be necessary to convert syntactically from one language to another.
2. Semantic level that indicates how the concepts should be interpreted. The semantic content must also be preserved whenever agents exchange information. To guarantee that a concept issued by an agent is understood in the same manner by the receiver and the sender of that message it is necessary that both agents have a common understanding of the meaning of the exchanged concepts. The technical way of achieving this objective is through common ontologies, which allow a uniform representation of meaning for concepts across different agents.
3. Communication level that is about the prepositional attitudes that the agents want to communicate to the others in order that the receiving agent can put the information being exchanged into context. In other words, the agents must be able to communicate complex attitudes about their knowledge and information. For instance, they might need to *inform* or *request* other agents.

ACLs should be able to implement these three layers or levels, which is the case of the two most important ACLs: KQML – Knowldge Query Manipulation Language (Finin, Labrou, & Mayfield, 1997; Finin et al., 1993; *KQML*, 2000; Mayfield, Labrou, & Finin, 1996) and its many dialects, and FIPA ACL – FIPA Agent Communication Language (FIPA, 2000, 2001b, 2002a).and FIPA ACL).

An ACL should not be confused with its content language. An ACL is only concerned with capturing propositional attitudes (performatives). The way the exchanged propositions (knowledge and information) are expressed is completely irrelevant, despite the fact that propositions are what the agents are really exchanging.

An important content language developed by the KSE with the intent of being a standard language for expressing properties of a particular domain was KIF – Knowledge Interchange Format (Genesereth & Fikes, 1992; KIF, 2002). KIF was designed to be applied as an *interlingua*, although some researchers do not agree with this claim or have some objections (Ginsberg, 1991).

KIF. It is a first order computer language using a LISP like notation. With KIF it is possible to express: 1) properties of things like 'robot SONY is a SCARA', 2) relationships between things like the relation *holds* to express that a 'Robot SONY holds Gripper SCHUNK, and 3) general properties of a domain like 'All SCARA robots have 4 axis'.

A variable is any atom that starts with the character '?' as in *?robot* or *?x*. If the variable is a sequence then it should start with '?*' as is the case with *?*list*. Connectives are represented as follows:

(not p) - $\neg p$ (and p1….pn) - $p_1 \wedge p_2 \wedge \ldots \wedge p_n$ (or p1….pn) - $p_1 \vee p_2 \vee \ldots \vee p_n$

(=> p q) – $p \Rightarrow q$ ($\Leftrightarrow$ p q) – $p \Leftrightarrow q$

(forall (?x ?y) (p ?x ?y)) - $\forall_{x,y}\, p(x,y)$ (exists (?x ?y) (p ?x ?y)) - $\exists_{x,y}\, p(x,y)$

In the following expression the function *temperature* returns the temperature value of an object m1 and verifies if its value is equal to 83 degrees Celsius: *(= (temperature m1) (scalar 83 Celsius))*.

The following expression defines the relation *bachelor*. An object is a bachelor if it is a man and is not married: *(defrelation bachelor (?x) := (and (man ?x) (not (married (?x)))))*.

KQML. KQML is a message-based language for agent communication. It is independent of the transport mechanism (TCP/IP, SMTP, IIOP, …), the content language (KIF, SQL, STEP, Prolog, LISP, SL, …), and of the ontology used.

It is possible to identify three conceptual layers in KQML:

1. ***Content*** – The actual message content.
2. ***Communication*** – Includes the features that describe the lower level communication parameters, such as the identity of the sender and recipient, and a unique identifier associated with the communication act.
3. ***Message*** – This layer determines the kinds of interactions one may have with agents that speak KQML. The identification of the protocol that is being used in the communication, and which *performative* or *speech act* is being sent are functions of this layer. In addition this layer covers features that describe the content language and the ontology being assumed. The message is composed of fields represented as *:<field name>*.

```
(ask-one                                    (tell
     :sender brokerAg                            :sender clusterMgAg
     :receiver clusterMgAg                       :receiver brokerAg
     :content members(M)          ⇒              :content members([ag1, ag2, ag3])
     :reply-with brokerRequest                   :in-reply-to brokerRequest
     :language Prolog                            :language Prolog
     :ontology brokerOntology                    :ontology brokerOntology
)                                           )
```

Figure 3-12 - Examples of a KQML message

Figure 3-12 shows two KQML messages that represent a dialogue between the *brokerAg* and the *clusterMgAg*. The *brokerAg* wants to know which are the members that belong to the cluster represented by the *clusterMgAg*. To generate the request the *brokerAg* issues the command with the performative *ask-one* with the content message *members(M)* that is represented in Prolog. The ontology to be followed is the *brokerOntology*.

Several fields are used by KQML to characterise the message. For instance, *:sender* contains the address of the sender, *:receiver* is the address of the destination agent, *:content* contains the message in a suitable language, *:language* indicates which language is being used in *:content*, *:ontology* indicates the ontology being used in the content message, *:reply-with* indicates that the sender wants the receiver to use the value attached to this field when replying to this message, and *:in-reply-to* is used to indicate to which message this one is being replied to. The field *:content* belongs to the content layer. The fields *:reply-with*, *:in-reply-to*, *:sender*, and *:receiver* belong to the communication layer. Finally, the performative, and the fields *:language* and *:ontology*, belong to the message layer.

In the example above the sender expects the agent to include in the field *:in-reply-to* the identifier *brokerRequest*, because the message from the sender includes that identifier in the field *:reply-with.* The reply message is a *tell* performative and the content message includes the answer to the request of the *brokerAg*. There are many performatives in KQML but they are not shown here due to space restrictions. They can be found in (Finin et al., 1993).

KQML has been widely used even if some disadvantages can be pointed out (Wooldridge, 2002):

1. Many dialects exist that make interoperability difficult.
2. The transportation mechanism was never precisely defined, which again made interoperability hard since it is almost impossible to communicate without defining which transportation mechanisms are used or supported.
3. Poor formal semantics definition, which allowed different interpretations and, consequently, agents that were claiming to talk in KQML were not.
4. No support for the *commissives* class of performatives. This is critical since it is difficult to implement multiagent systems without commissives, especially when coordination is required.
5. Large ad-hoc performative set.

These disadvantages led to the development of a new language that tries to overcome some of these aspects without suffering radical changes, the FIPA ACL.

FIPA ACL. FIPA ACL has its origins in the work of the Federation for Intelligent Physical Agents (FIPA, 2002c), which decided to develop an ACL based on the Arcol language that had been developed by France Telecom, and whose main characteristic was its strong formal semantics. The important characteristic of this language, when compared to KQML, is its stronger formal semantics.

It is similar to KQML in the following points:

- It is based on the *speech act theory*.

- It uses the same conceptual layers: content, communication, and message.
- The concrete syntax for the messages closely resembles KQML.
- It is independent of the transportation mechanism, the content language, and the used ontology.

The set of performatives in FIPA ACL is reduced to 22 (FIPA, 2002b). Each one is described with both a narrative form and a formal semantics based on modal logic (Girle, 2000). The most used performatives in the context of CoBASA are:

- ***Inform*** – This is the basic mechanism to communicate information and the most used. The content of an *inform* is a statement that the sender wants the receiver to believe in.
- ***Request*** – This is the second most used. Its content message is a request of the sender for the receiver to perform some action.
- ***Agree*** – It indicates that the sender agrees to carry out a *request* that was made before. This means that after an agent has received a *request* message, it replies with an *agree* message if it is the case that it wants to carry it out. An agree message is thus one of the messages that implements the request protocol.
- ***Cancel*** – Another performative message used to implement the request protocol. Whenever an agent no longer desires a particular action to be carried out it can issue a *cancel* message.
- ***Failure*** – This message is also used in the context of the request protocol and is issued whenever the requested agent is not able to perform the action it was requested for.
- ***Not-understood*** – The most common use of this message is to indicate to another agent that the message this agent has just sent was not understood. The message is one of the most important mechanisms in FIPA-ACL for error handling.
- ***Query-if*** – This is used to send a query to another agent. The content message contains the statement the agents want to enquire about.
- ***Refuse*** – Another performative message used by the request protocol. When an agent does not intend to do an action it is asked for it replies with a *refuse* message. The message content contains both the requested action and the reason why the agent will not perform it.

Although FIPA ACL does not have any commitment to any particular content language, FIPA provides specifications for KIF (FIPA, 2003), RDF - Resource Description Framework (FIPA, 2001c), CCL - Constraint Choice Language (FIPA, 2001a), and SL – Semantic Language (FIPA, 2000). Most of the current implementations in FIPA-ACL follow SL. This is the case of JADE (JADE, 2001), which is the FIPA compliant development environment used to implement CoBASA.

Figure 3-13 shows the various parameters or fields used in FIPA ACL messages. Most of them are similar to KQML messages. The field *:protocol* that is used to denote the interaction protocol that the sender agent is employing is new. These protocols are *conversations* that can be established between the agents. An interaction protocol is defined by FIPA as typical patterns of message exchanges. They have been defined to simplify the engagement of agents in meaningful conversations.

The following protocols are defined by FIPA: FIPA Request, FIPA Query, FIPA Request When, FIPA Contract Net, FIPA English Auction, FIPA Dutch Auction, FIPA Brokering Interaction, FIPA recruiting Interaction, FIPA Subscribe Interaction, and FIPA Propose Interaction. The specifications for these protocols can be found in the FIPA web site (FIPA, 2002c).

Parameter	Category of Parameters
performative	Type of communicative acts
sender	Participant in communication
receiver	Participant in communication
reply-to	Participant in communication
content	Content of message
language	Description of Content
encoding	Description of Content
ontology	Description of Content
protocol	Control of conversation
conversation-id	Control of conversation
reply-with	Control of conversation
in-reply-to	Control of conversation
reply-by	Control of conversation

Figure 3-13 - Parameters of FIPA-ACL messages.

Figure 3-14 shows two messages expressed in FIPA ACL that represent part of the request protocol. Only the REQUEST message and the last INFORM of the request protocol are shown.

```
(REQUEST                                    (INFORM
     :sender ag1                                 :sender clusterMgAg
     :receiver clusterMgAg                       :receiver ag1
     :content                                    :content
       ((joinCluster type robot))                   ((succesfullJoin reason "join accepted"))
     :reply-with joinCluster                     :in-reply-to joinCluster
     :language SL0                               :language SL0
     :ontology agentOntology                     :ontology agentOntology
     :protocol FIPA Request                      :protocol FIPA Request
)                                           )
```

Figure 3-14 – Examples of a FIPA ACL message

FIPA ACL overcomes some of the drawbacks highlighted for KQML, namely the formal aspects and the existence of many dialects. Being defined by an organisation that somehow tries to standardise many aspects related to multiagents, it is likely that this effort has more chances to succeed than if it was a single entity attempting to force the acceptance of a new language.

3.2.2.3 Social Order

As agents become increasingly autonomous and interacting in more complex worlds new organisational and cooperative strategies are required. Under these conditions it is necessary to take into account the individual agent view that needs to interact autonomously with other artificial or human agents in a more complex way and, at the same time, it is necessary to view the society of agents as a whole. Therefore, it is necessary to take into account the individual benefit and the collective benefit.

Humans are inherently complex, need to interact in a very complex way, decide under incomplete information and, in many cases, under limited trust, and yet be able to create societies (group of humans interacting) that are very complex and flexible. In this way, humans and human societies are a good source of inspiration to solve the problems faced by artificial agents and their societies. This is to say that many aspects from socially related disciplines such as economics, politics, psychology, sociology, and philosophy can be adapted to solve problems in the multiagent domain. This new area dedicated to the application of human social aspects in the world of artificial agents is becoming more and more important and is known by different names such as: Social order (Conte & Dellarocas, 2001b), social laws (Shoham & Tennenholtz, 1995), social artificial intelligence , socially intelligent agents (Dautenhahn, 2001), and sociality (Castelfranchi, 1998).

This new domain is concerned with the following aspects:

- How the decision-making process is carried out under two contradictory interests: individual benefit versus group benefit (Hogg & Jennings, 2001; Kalenka & Jennings, 1999; Panzarasa et al., 2001).
- How agents interact to convince other agents of their viewpoints (Pietro Panzarasa & Nicholas R. Jennings, 2002).
- What kind of mechanisms should be developed to facilitate social interaction and the living of agent societies (Castelfranchi, 1998; Conte & Dellarocas, 2001a, 2001b; Dellarocas & Klein, 2001).
- How the mechanisms are enforced on the members of the society (Conte & Castelfranchi, 2001; Shoham & Tennenholtz, 1995).

Please note that the social ability that was meant in (Wooldridge & Jennings, 1995) for an entity to be considered an agent was the possibility of interacting with other agents. Nothing was said about if that interaction was pursuing selfish benefits or group benefit. The concept of social order is a step further in considering that interaction at group level is important. A key point to understand is that the benefit to the society can, on many occasions, be more important than the benefit to the individual. A typical example in human societies occurs when an army is fighting. The individual soldiers (agents) as a whole compose the army, which has the goal to defeat its enemy. Sometimes, a military unit must be sent into a mission that, from the very beginning, is known to suffer severe casualties. The individual members that compose this unit have no individual benefit at all in accomplishing this mission. However, the benefit for the group (the army) might be high because the action of that unit could help an overall victory. The open problem is how to convince the individual agents to sacrifice their individual interests over the group benefit or, in other words, how to include in the individual's decision-making process the group goals. Much work has been done about integrating these two views in the decision making process (Boella & Lesmo, 2001; Egashira & Hashimoto, 2001; Hogg & Jennings, 2001; Kalenka & Jennings, 1999).

Another view of the problem is about coordination. Lets imagine a community of mobile robots that share a road and where each member should avoid other members. One way to solve this coordination problem is considering that one programmer develops the entire community. The programmer can think about all the possible interactions of the robots and code them, avoiding race conditions and guaranteeing coordination. However, this has a serious drawback because increasing the number of agents makes the intricacy of the interactions so complex that it is almost impossible to be previewed by the programmer, and adding agents requires changes in the code of all agents. An alternative solution is considering that the agents will be programmed individually in an unconstrained fashion, but now, as the agents have individual unconstrained behaviour, it is necessary to include some coordination mechanism. One first coordination alternative is to consider a supervisor to whom agents ask for help whenever they find a race condition (for instance, at a crossroad, to decide who should pass first). The disadvantage of this solution is the reliance on only one entity that if failed would stop the entire system. Another alternative is replacing the centralised approach with a decentralised one. In this case, whenever the agents find a race condition they negotiate with each other. This requires the agents to be more complex because they would require more communication abilities and the capacity to engage in negotiations with their peer robots. Nonetheless, there still are some limitations in this approach too because it requires complex agents. If the agents could, instead of engaging in negotiations every time they need to interact, follow some kind of social law/norms then their decision-making process would be easier. An example of this social law is, for instance, the law that each robot should drive on the right side and, when at a crossroad, the robot that is on the right should pass first. Rules/laws like this reduce the complexity because now they can operate without the need to negotiate in every interaction situation. The laws are used to constrain the plans of the agent.

Norms and obligations are examples of mechanisms created by humans to help their interaction. To guarantee their enforcement institutions such as police, judges, courts, etc have been developed.

Definition 3.2 – Norm (Conte & Castelfranchi, 2001; Conte & Dellarocas, 2001a)
Norms are specifications of behaviour that all the members of the society are expected to conform to. A norm prescribes not only what must be done, but also why it must not be done.

A problem when achieving coordination through norms is how one may guarantee that the members of the society follow the norms. The success of a coordination system based on norms is the assumption by each member that everybody follows the norms; otherwise chaos would ensue. An approach to guarantee the enforcement of the norms is by developing institutions (Balzer & Tuomela, 2001) to ensure its accomplishment. Another way is by developing incentives or sanctions for the agents (Conte & Castelfranchi, 2001; Sartor, 2001).

In this context, an interesting aspect of norms is that agents are prepared to lose some autonomy for the benefit of society for the sake of simpler behaviour and safety. This is what happens with human institutions like the police and the law, which, in fact, simplify human life and increase safety. If those institutions were not available, every time someone owed money to another, it would result in the person who had lent the money to pursue the other person, which implies more complex behaviour, unstable result, and less effectiveness. Norms are thus used to solve the dilemma of individual versus global utility. Agents might prefer to follow global objectives if incentives or sanctions make them preferable to their individual ones.

While it is consensual that norms are an important mechanism to achieve coordination and group benefits, there is much less consensus on the way the agents should follow the norms. There is a divide between the researchers coming from the area of artificial intelligence and rational agents, and those coming from the social area. Rational agents researchers think that norms should be followed by imposing incentives or sanctions. According to them this simplifies the decision-making process because it needs only to transform the value of the social utility function (computes the social benefit) into a kind of individual benefit. Social researchers, on the other hand, think that norms should be followed for intrinsic reasons. The agent should follow them not because it has an incentive or penalisation but because it intrinsically thinks it is better. This divide led to the following classification for agents that follow norms as defined by (Conte & Castelfranchi, 2001):

1. ***Rational deciders*** – In these agents the norms change the agents' preferences to accomplish socially desirable actions. Rational decision occurs when the action is chosen based on the value that maximises a subjective utility function. Sanctions and incentives must be provided to such a degree that the individual utility of socially desirable actions is higher than the individual utility of socially undesirable actions. They execute the most convenient norm only in the presence of sanctions or incentives. "*They will comply with the norms to the extent that the (positive or negative) incentive is such that the utility of obedience is higher than the utility of transgression (sanction is higher then the convenience of transgression)*."
2. ***Normative agents*** – These agents execute actions based upon intrinsic motivations. A norm prescribes not only what must be done but also why it must be done. By providing a reason for the adoption, the agent can decide, even if it has no sanction or incentive. These agents choose the most convenient norm. They can violate a norm they consider unfair or 'contrary to duty', which is something rational deciders do not. "*They will comply with a norm as long as either ideal conditions apply (intrinsic motivations) or sub-ideal conditions apply (in this case they will behave as rational deciders) or ideal conditions apply and the norm is not unfair*."

Although rational deciders are simpler to implement, it is argued by (Conte & Castelfranchi, 2001) that in an environment composed of normative agents transgression is reduced when compared to incentive based agents. According to social researchers, incentives can produce negative effects on several aspects of social learning and norm compliance. When an agent, for instance, is acting as a

representative of an organisation or human in a negotiation or commercial transaction, a rational decider agent might be preferable, in case it is benevolent to its users. This happens because they always choose their actions based on some utilitarian function. However, when an agent is participating as a partner, as in a coalition or virtual organisation, it might be preferable to have a normative agent whose actions are, perhaps, more cooperative.

The last relevant aspect to mention is Contractual Agent Societies (CAS) (Dellarocas & Klein, 2001) in which contracts are used to deal with the problem of interacting heterogeneous agents in a social context. As stated in (Dellarocas & Klein, 2001) CAS are:

> *"Open systems where independently developed agents configure themselves automatically and coordinate their behaviour through a set of dynamically negotiated social contracts, which define the shared context of agent interactions, and a system of social control, which is responsible for avoiding, or detecting and resolving exceptions, that is, deviations from the desirable system behaviour."*

Social control is achieved through institutions that enforce the social contracts. The behaviour of CoBASA agents when working together (coalition) is constrained by a contract that in fact is acting as a norm. Furthermore, CoBASA is based on a system of incentives and sanctions to instigate agents to participate in group work and restrain them from behaving selfishly. Finally CoBASA defines a set of institutions that are responsible for the enforcement of group behaviour.

3.2.2.4 Coalitions

Coalitions are discussed in the context of societies of agents because they are an important method for achieving cooperation in multiagent systems, even with self-interested agents, which can increase their ability to satisfy their goals and maximise their own personal payoffs (Shehory & Kraus, 1999). Coalitions and negotiation are related because it is not possible to create alliances or coalitions without involving the potential partners in a negotiation process. Only after the partners reach a common understanding about the terms of their participation is it possible to claim that a coalition or alliance exists.

Definition 3.3 - Coalition
Coalition is a temporarily organised group of agents who agree to jointly accomplish a certain task in a cooperative manner.

Coalitions can be used to solve many practical problems in several domains of the society:

- Business, and especially virtual organisations (Camarinha-Matos, 2002b; Camarinha-Matos et al., 2001; Jain, Aparicio, & Singh, 1999) since a virtual enterprise is itself a coalition of independent and autonomous companies that may have both self-interested and group objectives.

- Applications of ubiquitous and mobile computing in wireless environments (Klusch & Gerber, 2002).
- Electronic markets (Tsvetovat, Sycara, Chen, & Ying, 2001).
- Non-belligerent military operations (peace keeping, non-combatant evacuations) and humanitarian missions (Tate, Bradshaw, & Pechoucek, 2002). These operations, contrary to the classical military operations that are hierarchical, are cooperative and involve several loosely organised groups of people or institutions. These missions require agility in creation, use of limited resources, a kind of virtual organisation, and need frequent changes in both strategies and objectives. They accommodate organisations with different interests, and are relatively flexible and dynamic since new organisations can join in or leave the coalitions whenever they want.

The following assumptions must be considered when discussing coalitions:

- Candidate agents are autonomous, might be self-interested or cooperative, and can change their cooperative attitude. For instance, in humanitarian coalitions, the participating agents are cooperative in the short term because they need to cooperate to achieve their humanitarian goal but, in the long term, they compete for participation (Pechoucek, Marik, & Bárta, 2002). The same situation happens in CoBASA since manufacturing agents cooperate when participating in a coalition (short-term), but compete to participate (long-term).
- A value system is assumed to quantify benefits or payoffs.
- A candidate agent must have a value that measures its individual quality or ability (curriculum). This value can be generated using different types of functions, adapted to the different needs that transform one or more inputs (characteristics) into a value. In CoBASA, manufacturing agents have an individual curriculum.
- A value that quantifies the "quality" of the coalition. The candidate to decide if it is worth joining the coalition or not might use it.
- Participating agents must receive some payoff by their participation in the coalition.
- Candidate agents follow *personal rationality*. An agent has personal rationality if it joins a coalition only in the case its benefit is bigger than it could be if working alone.
- Candidate agents seek *personal payoff*, which is the characteristic to maximise its personal payoff.
- The agents communicate because it is not possible to negotiate or make agreements without it.

Coalitions can be formed according to two different approaches: ***utility-based***, and ***complementary-based*** (Klusch & Gerber, 2002).

The ***utility-based approach*** is used in situations in which the agents that compose coalitions are self-interested and that, above all, are interested in maximising their individual payoffs. They colligate to achieve that objective. Coalition formation based on this approach relies on the work of game theory, and particularly the N-person games (Rapoport, 1970). Game theory discusses which

coalitions can be formed, the way the benefits will be distributed among the members, and the stability of the coalitions. However, it does not provide algorithms and does not consider the specificities of multiagents such as distribution, communications costs, and limited computation time. Shehory and Kraus propose in (Shehory & Kraus, 1999) a multiagent approach to the coalition formation problem based on the utilitarian approach (game theoretic) that follows two different approaches: computation oriented, and negotiation oriented. The trade off between these two approaches is the quality of the solution versus the speed with which a solution can be reached. The computation-oriented approach is indicated for situations where there are a small number of agents or time and computations are cheap. On the other hand the negotiation-oriented approach is indicated when there are a large number of agents, computations are costly, and time is limited.

Complementary based approach relies on the collaborative use of complementary individual skills to enhance the power of each agent to accomplish its goals. This is a different kind of coalition because, in this case, the agents need to cooperate to achieve their goals. This is a different perspective on social interaction, by taking into account the fact that agents may have complementary skills, which may be needed to achieve the agents' own goals. Thus, the existence of other agents *enhances* the autonomy and power of individual agents. Even if an agent cannot achieve some goal on its own, it may achieve it by asking others for help.

This is what happens in CoBASA because manufacturing agents need to colligate to have the necessary skills to be able to perform complex manufacturing operations. In this approach the game theory does not work well because the objective now is not to maximise the payoff of each individual agent but rather the benefit of the whole group. Although this approach has been much less used than the utility based one, it has become ever more important because in many practical situations it is the global result that is really important, either it be the formation of a coalition to deal with a disastrous situation (Pechoucek et al., 2002; Tambe, Pynadath, & Chauvat, 2000) or a coalition of manufacturing agents able to produce a product with some set of manufacturing skills.

An area that covers coalition formation for this type of coalition is social reasoning (Sichman, 1998; Sichman, Conte, Demazeau, & Castelfranchi, 1998), which is based on *social dependence* (Castelfranchi, Micelli, & Cesta, 1992). The idea behind this is that the agents need to reason about the other agents using a social approach. An agent is said to be dependent on another if the latter may facilitate/prevent it from achieving one of its goals. Agents evaluate the susceptibility of other agents to adopt its goals based on the notion of *dependence situation*. (Sichman, 1998) and (Morgado & Gaspar, 2000) present social reasoning mechanisms for creating coalitions.

3.2.2.5 Standardisation Efforts

The most important effort in agent and multiagent standardisation is being pursued by FIPA (FIPA, 2002c), which is a non-profit organisation founded in 1996 and whose purpose is to promote the success of agent based applications, services, and equipment. The core mission of FIPA standards is to

facilitate internetworking of agents and agent systems across multiple vendors' platforms, and maximise interoperability. Although FIPA had, originally, an emphasis on practical and industrial uses of agents, intelligent cognitive agents are also currently being covered.

FIPA is currently offering specifications for the following areas:

- *Applications* – Are example application areas in which FIPA agents can be deployed. They represent ontology and service description specifications for a particular domain.
- *Abstract architecture* – Deals with the abstract entities that are required to build agent services and an agent environment.
- *Agent communication* – Deals with architectures to support the communication of interacting agents, communication and content languages to express the messages, and interaction protocols, which expand the scope from single messages to complete transactions.
- *Agent management* – Deals with the control and management of agents within and across agent platforms.
- *Agent message transport* – Deals with the transport and representation of messages across different network transport protocols, including wireless environments.

FIPA operates using open international collaboration of member organisations that are universities and companies active in the multiagent field.

3.3 ONTOLOGIES

3.3.1 Introduction

The work on ontologies was mainly motivated by the need for sharable and reusable knowledge bases (Gruber, 1995; Guarino, 1995; Uschold & Gruninger, 1996). An ontology produces a common language for sharing and reusing knowledge about phenomena in a particular domain. It is a way of providing a common understanding of concepts, generally focalised in a specific domain, that have different views and interpretations from the different practioners of that domain. By reaching agreement on how these concepts should be formally organised and structured it is possible to have a kind of *Lingua Franca* that enables common understanding and exchanging.

Definition 3.4 - Ontology

An ontology is generically a branch of philosophy dealing with the order and structure of reality. In terms of computer science it is a simplified view of a particular subject area using a formal abstract model composed of the concepts and their interrelationships, which characterise that area.

An ontology is usually organised in taxonomies and contains modelling primitives such as classes, relations, functions, and axioms as well as a set of inference rules. The taxonomy defines classes of objects and the relationships among them. Inference rules provide ways to help create new concepts or actions from the defined ones, what increase the expressiveness of ontologies.

To show how important an ontology can be in a multiagent society consider a scenario in which an agent (*Querying Agent*) is querying different types of agents representing different types of robots (Figure 3-15). Each of the agents uses a different name to represent the concept of who produced the robot, i.e, different names for the same concept. *Agent Robot Fanuc* knows it as the name Made_By, the *Agent Robot Cartesian* as *Company*, *Agent Robot Mitsubishi* as *Manufactured_By*, and the *Querying Agent* as *Maker*. Whenever the querying agent needs to know from a certain robot agent who was the company that produced it, it would like to do it by issuing a query using the concept it knows, *Maker* in this case. However this is only possible if the robot agents understand what the concept *Maker* means. If an ontology exists that groups these different names into one concept, the agents could interact and communicate in a very easy way. The adopted terms in the ontology can then be used in asking and answering questions, making assertions, offering insights, and describing practices.

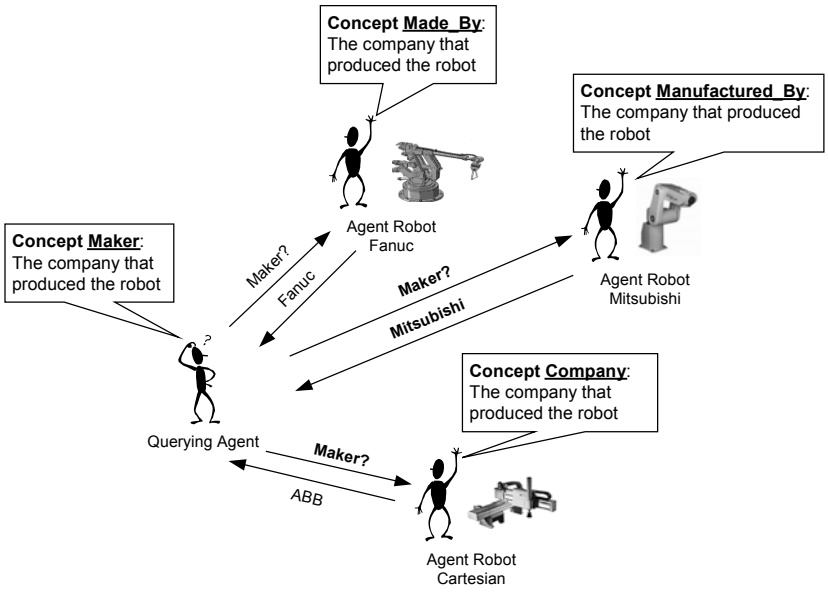

Figure 3-15 - Agents communicating

3.3.2 Important aspects

A few years ago researchers, if asked about the meaning of ontologies would reply that it was an esoteric field in philosophy that studies beings or what type of things exist. However, nowadays a simple Internet search for this term will give thousands of hits and, among the first hits, it is possible to find web pages containing terms such as “web semantics”, “enterprise ontology”, “virtual

enterprise", which are more practical rather than philosophical. The probable reasons why ontologies are becoming more and more important are:

1. Increasing use of computer agents - Computer systems are moving away from human centred to agent oriented computing (Castel, 2002), and an increasing number of software agents acting on behalf of humans are interacting more and more, not only with humans but also with other software agents. Important types of software agents that are strongly related to ontologies are those created for collecting web content from various sources.
2. More Knowledge Management practices – As knowledge management practices among organisations are increasing so is the need to structure and organise their information and knowledge. To successfully create and maintain knowledge within organisations it is necessary to create the terms and definitions deemed to be important in characterising those organisations at a desired level of detail.
3. The importance of the World Wide Web – With the growing importance of the web to support various businesses, scientific, and personal activities a new web is being defined under the name of ***Semantic Web*** (Berners-Lee & Fischetti, 1999; Berners-Lee, Hendler, & Lassila, 2001; Gómez-Pérez & Corcho, 2002; Hendler, 2001; Lassila & McGuinness, 2001; McIlraith, Son, & Zeng, 2001; Noy et al., 2001; *Semantic Web*, 2002)[26]. Most information on the Web is designed solely for human consumption. Computers are better at handling carefully structured and well-designed data, and yet even when information is derived from a database with well-defined meanings, the implications of the data are not evident to a robot browsing the web. Nowadays, the information placed on the web needs to be more and more in a form that machines can 'understand' rather than simply display. The concept of machine-understandable documents does not imply some magical artificial intelligence allowing machines to understand human mumblings. It relies solely on a machine's ability to solve well-defined problems by performing well-defined operations on well-defined data, which can be facilitated with the support of Ontologies[27]. For instance Tim Barnes Lee, the creator of the Web, considers that ontologies are the key point for his last work about semantic Web (Berners-Lee et al., 2001).

Ontologies are used in the following activities as defined in (Gruninger & Lee, 2002):

- ***Communication***
    a. Between implemented computational systems
    b. Between humans
    c. Between humans and implemented computational systems

[26] The Semantic Web brings structure to the content of web pages that will allow those software agents roaming from page to page to carry out their sophisticated tasks on behalf of humans.

[27] Semantic Web is much more than ontologies, which are only part of the problem. Other concepts important for a Semantic Web are Knowledge Management. Important technologies are eXtensible Markup Language (XML), and Resource Description Framework (RDF).

- ***Computational Inference***
    a. To represent and manipulate plans and planning information
    b. To analyse the internal structures, algorithms, inputs and outputs of implemented systems in theoretical and conceptual terms
- ***Reuse and Organisation of Knowledge***
    a. To structure or organise libraries or repositories of plans and domain information.

The progress of work in ontologies is leading to the emergence of a new discipline called ontological engineering that considers the entire ontology life cycle: design, evaluation, validation, maintenance, deployment, mapping, integration, sharing, and reuse (Fernández, Gómez-Pérez, & Juristo, 1997).

Ontologies are typically designed using five approaches (Holsapple & Joshi, 2002):

- ***Inspiration*** – The ontology is developed according to the individual viewpoint of its creator about the domain. When developed to be used by a community of agents this approach can have a serious drawback in terms of its acceptance by the whole community, unless the potential adopters share the same personal views of the developer.
- ***Induction*** – The ontology development starts by analysing a specific case in the domain of interest. Then the resulting ontology is applied to other cases of the same domain. This is an inductive process since the ontology starts from a particular case, and as long as its acceptance becomes wider it becomes more generic. This approach is very well indicated for the cases in which a good specific case in a certain domain exists, which can then be used as the basis for creating other ontologies for the same or interrelated domains. Its adoption is dependent on how generic the concepts are.
- ***Deduction*** – Contrary to the inductive process, the deductive approach is based on a philosophy that privileges the adoption of generic concepts that can later be used to generate new concepts when a new ontology is created for a particular case. Its adoption is largely dependent on how generic the concepts are, namely how well the specificities of the specific problem might be accommodated by the previously developed generic concepts.
- ***Synthesis*** – The ontology is developed starting from a set of existing ontologies, each one providing a partial characterisation of the domain but not subsuming the others, which are then synthesised and unified. This approach is particularly relevant in today's situation of a world constituted by several dispersed ontologies in the same or interrelated domains. Synthesising involves systematic integration, elimination of sketchier characterisations in favour of more fully developed ones, and reconciliation of different terminologies. Its success depends largely on how well the starting ontologies were defined as well as on the ability of the creator to develop a coherent new set of concepts that clearly represent the various views presented in the starting ontologies. A good example could be the synthesis of a new ontology

in the assembly domain. The existing ontologies covering the different areas, e.g. product, production systems, could be unified and this development could be used as an opportunity to structure and characterise that domain in a coherent way.

- ***Collaboration*** - The ontology is developed starting from multiple individuals' viewpoints about the domain. In many cases these viewpoints are connected with an initial ontology that acts as an anchor for the new ontology. The development is a joint effort of intentionally cooperating people that interacts to reflect their experiences and viewpoints. The advantage of this approach is that it can achieve a wider acceptance, especially if resulting from several contributors with diversified experiences and knowledge. The big challenge is how to create a development environment to appropriately coordinate all contributions. Environments to accommodate several and diverse people require consensus building mechanisms to gather, discuss, and approve all the proposed contributions. With the correct development environment, of which a proposed example can be found in (Holsapple & Joshi, 2002), this approach offers unquestionable advantages over others in terms of wider audience acceptance and richer semantics.

3.3.3 Challenges and Implementations

The most relevant challenges or open questions when designing and creating ontologies are:

- ***Ontological commitment*** – This corresponds to the agreement by the multiple actors involved in the design of a particular ontology. This is a complex problem because it requires accommodating the different viewpoints from the various contributors. A suitable collaborative environment helps ontological commitment. Experience shows that the problem of building sharable ontologies lies more in the social process than in the technological aspects.
- ***Difficulty in formalising some concepts*** – Some concepts in the world are inherently more complex to formalise than others. The intended semantics of those concepts whose nature is more subjective are more difficult to capture and represent. Moreover, deciding which attributes and inference rules characterise a concept or terms depends very much on the objective for the ontology. Therefore, creating generic ontologies is more complex since the future use of each term must be foreseen.
- ***Some domains are poorly structured, formalised, and studied*** – Some domains are not yet sufficiently studied and analysed to support the development of an ontology. An ontology is much more than a common glossary because it involves not only a common understanding of terms but its formalisation through a set of formal axioms that constrain interpretation and well formed use of its terms (concepts). Consequently the development of ontologies for domains not yet conveniently formalised is difficult.

- ***Modelling languages*** – The languages to formalise the concepts should be simultaneously easier to understand, and formally correct and rich, which is, in many cases, a difficult obstacle to overcome. The problem of readability versus semantics and formality is important when choosing a language to model an ontology. Ideally, people without expertise in logic should be able to produce ontologies.
- ***Generic versus domain specific ontologies*** – When creating an ontology a question always present is deciding if it will be generic (covering general world or common sense knowledge) or domain specific. A generic ontology is always an ideal objective because it can be easily extended and applied to other domains. However, this is a difficult task and, consequently, researchers have abandoned this goal for the more realistic domain specific approach.
- ***Integration*** – As more and more ontologies are being developed, namely domain specific ones, an integration strategy is needed in order to integrate them with other ones that have been previously developed. Since the word "integration" has been abused by several researchers, the term is clarified in (Pinto, Gómez-Pérez, & Martins, 1999) as being the case of building a new ontology reusing (by composing) other available ontologies.
- ***Difficulties in addressing realistic problems*** – The use of real ontologies operating effectively and efficiently is not very wide because realistic problems are difficult to address by current design methodologies for ontologies.
- ***Requires specialised people*** – Due to the lack of efficient design methodologies, despite the efforts made by several researchers (Farquhar, Fikes, & Rice, 1997; Fernández et al., 1997; Guarino, 1997; Holsapple & Joshi, 2002; Noy & McGuinness, 2001; Uschold & Gruninger, 1996), designing and maintaining an ontology is still a task that must be done by specialised people. This is another cause for narrow acceptance and dissemination of ontologies.
- ***Poor use among industry*** – Ontologies are not widely used in the industrial world probably because of the challenges and open questions referred to in the previous points. Research on ontology applications for industrial problems is therefore necessary.

Most of the current efforts in the ontology domain are focused on the web (Berners-Lee et al., 2001; Gómez-Pérez & Corcho, 2002; Noy et al., 2001; *Semantic Web*, 2002), which is not surprising because the web is becoming increasingly used by computer systems instead of humans. Another area particularly active in terms of ontology development is the health domain, namely medicine, pharmacology, and biology, where considerable research effort has been made in the last years (Abernethy, Wu, Hewet, & Altman, 1999; Altman, 2001; Altman et al., 1999; Hendler & Stoffel, 1999; Lathrop, 2001; Musen, 1998; Oliver, Shahar, Musen, & Shortliffe, 1999; Rubin, Hewet, Oliver, Klein, & Altman, 2001).

In order to provide better insight into the current importance of ontologies the Languages used to design ontologies, the Tools or development frameworks used to create and edit ontologies, the

Ontologies that have been created so far, and the Application tools that use ontologies are listed. Each element in the list is only briefly explained. For further details, bibliographic references are provided.

Languages

- ***KIF-Knowledge Interchange Format*** (Genesereth & Fikes, 1992; KIF, 2002) – It is a monotonic first order logic that has declarative semantics and some minor extensions to support reasoning about relations.
- ***Ontolingua*** (Farquhar et al., 1997; Gruber, 1993; *Ontolingua*, 2002) – It is an extension to KIF with additional syntax to capture intuitive bundling of axioms into definitional forms and a *Frame Ontology* to define object-oriented and frame-language terms.
- ***Loom*** (Brill, 1993; *Loom*, 2002; Preece et al., 2001) – Loom is a Lisp based knowledge representation language that supports a "description" language for modelling objects and relationships, procedural programming, and deductive reasoning based on production-based and classification-based inference capabilities.
- ***CommonKADS Conceptual Modelling Language*** (Schreiber, 1999; Studer, Benjamins, & Fensel, 1998) – It is a semi-formal notation for the specification of CommonKADS knowledge models.
- ***CycL*** (Guha & Lenat, 1994; Lenat, 1998; Lenat & Guha, 1989) – Its syntax is derived from first order predicate calculus and is used by the Cyc framework.
- ***DAML+OIL DARPA Agent Markup Language+Ontology Interchange Language*** (DAML, 2000; Fensel, van Harmelen, Horrocks, McGuinness, & Patel-Schneider, 2001; Hendler, 2001) – The DAML objective is to develop a language able to represent semantic relations in machine readable ways to facilitate the concept of the Semantic Web.

Development Frameworks

- ***Ontolingua server*** (Farquhar et al., 1997; *Ontolingua Server*, 2002) – The Ontolingua server should not be confused with the Ontolingua language. The server is the software system or general collaborative environment to facilitate the development and sharing of portable ontologies represented in the Ontolingua language. It uses the World Wide Web to enable wide access and provide users with the ability to publish, browse, create, and edit ontologies.
- ***Cyc*** (*CYC*, 2002; Lenat, 1995, 1998; Lenat & Guha, 1989) – Since the ontology is so connected to its development framework no distinction will be made here. It is a generic ontology for common sense knowledge, and one of the earliest works in computational ontologies. Natural language was one of the main motivations behind its development.
- ***WebOnto*** (Domingue, 1998; WebOnto, 2002) – A completely graphical collaborative Web based tool. "*WebOnto was designed to support the collaborative browsing, creation and editing of ontologies. (...) WebOnto was aimed to be easy-to-use, yet have facilities for scaling up to large ontologies.*" (Domingue, 1998).

- ***Protégé-2000*** (Gennari et al., 2003; Noy, Fergerson, & Musen, 2000; Noy et al., 2001; Pinto, Peralta, & Mamede, 2002) – This is an Open Knowledge-Base Connectivity (OKBC) compatible ontology environment developed at the Stanford Medical Informatics department. Protégé is a generic graphical open-source Java-based tool for ontology and knowledge editing and acquisition**,** which also provides an extensible architecture for the creation of customized knowledge-based applications.
- ***RiboWeb*** (Altman et al., 1999) – This is an environment targeted for creating ontologies in the biology domain, specifically to the study of the Ribosome. This environment, together with its ontologies, permits a collaborative scientific tool.
- ***KADS22*** (Schreiber, 1999; Schreiber, Wielinga, De Hoog, Akkermans, & Van de Velde, 1994) – Supports the construction of ontologies based on the CommonKADS methodology, which has been developed at SWI at the University of Amsterdam. KADS22 is an interactive graphical interface for the Conceptual Modelling Language (CML), which is the CommonKADS underlying language for ontology development.
- ***Apeckcs (Adaptive Presentation Environment for Collaborative Knowledge)*** (Tennison, O'Hara, & Shadbolt, 2002) – An ontology server developed at the University of Nottingham that supports collaboration by allowing individuals to create personal ontologies.
- ***ONTOCLEAN*** (Guarino & Welty, 2002) – It provides guidance for good ontology engineering practices as well as evaluating how well engineered an ontology is.

Ontologies

- ***SUO (Standard Upper Ontology)*** (*SUO*, 2002) – This is an ontology whose development is being supported by the IEEE. As the name *Upper* suggests it is a generic ontology. It aims to develop meta, generic, abstract, and Philosophical concepts to cover several domains of interest.
- ***PILNIUS*** (Van der Vet, Speel, & Mars, 1993) – A domain ontology for representing mechanical properties of ceramic materials.
- ***Sowa*** (Sowa, 2000; *Sowa Ontology*, 2000) – This is an ontology based on conceptual graphs that is developed more for theoretical investigations about the philosophy behind ontologies than to produce a real working ontology.
- ***TOVE (Toronto Virtual Enterprise)*** (Fox, 1992; Fox & Gruninger, 1998) – This is a domain specific ontology for enterprise modelling. Although it uses the name Virtual Enterprise it is not an ontology for virtual enterprises but it provides instead terminology for enterprises. The TOVE test bed provides a model of an enterprise and tools for browsing, visualisation, simulation, and deductive queries.

- ***WordNet*** (Guarino, Masolo, & Vetere, 1999; Miller, 1995; Noy & Hafner, 1997) – It is not really an ontology but more a linguistic database formed by terms grouped into equivalence sets, called *synsets*, which are sets of synonyms.
- ***Enterprise*** (Uschold, King, Moralee, & Zorgios, 1998) – It is a collection of terms and definitions relevant to business enterprises developed within the Enterprise project. It provides a common language to help business software applications to communicate. The ontology is represented in Ontolingua.

Applications

- ***OntoSeek*** (Guarino et al., 1999) – It is a system designed for content based information retrieval from online yellow pages and product catalogs by combining an ontology-driven content matching mechanism with a representation formalism. The ontology behind OntoSeek is WordNet.
- ***WebKb*** (Martin & Eklund, 2000) – It is essentially a tool that interprets semantic statements stored in web accessible documents. By combining lexical, structural, and knowledge based techniques it is possible to exploit web documents and allow easier web page searching.
- ***OntoBroker*** (Decker, Erdmann, Fensel, & Studer, 1999) – It is considered to have pioneered the work on the Semantic Web. It is an integrated system to extract, reason, and generate domain specific data. The ontologies are created using a frame logic language that has the advantage of also being used for querying.
- ***PlanetOnto*** (Domingue & Motta, 2000) – It is an integrated suite of tools that supports publishing, ontology driven document formalisation, and augments standard browsing and search facilities with deductive knowledge retrieval.

3.4 CONTRACTS

Contracts are used in human societies to shape behaviour, settle disputes, secure rights, and protect liberties. They can be used in the same way in artificial agents societies. Since contracts are used in CoBASA agents as the inspiration source for the coordination of its agents (Barata & Camarinha-Matos, 2002, 2003) it is relevant to analyse and study how contracts are formed as well as the basic mechanisms that govern them to understand how they can be adapted to a world where artificial agents replace human actors.

Although not directly useful for CoBASA since its contracts need not to be legally sound, there are two aspects related to contracts that deserve some attention:

1. How the increasingly extensive use of computers as conduits for electronic trading affects the law of contracts.
2. How computer based systems establish electronic contracts automatically in a trading process, and which tools or applications already exist that make use of them. Actually, this second point is a direct consequence of the increasing use of computers for trading and the development of electronic commerce on the Internet in particular.

3.4.1 Background on legal issues

Most of the legal issues here are based on the Common Law, as generically followed by Britain, United States of America, New Zealand, and Australia because these law systems are flexible and with a long tradition in contract law. In addition it was easier to find information about their basic principles. For instance, the only available book in Portuguese specifically about contracts is "Contratos I" from (Almeida, 2000).

A contract is defined by the Merriam-Webster's Dictionary of Law as *"an agreement between two or more parties that creates in each party a duty to do or not do something and a right to performance of the other's duty or a remedy for the breach of the other's duty"*. A contract is made up of a promise of one entity to do a certain thing in exchange for a promise from another entity to do another thing. Some law researchers (Almeida 2001) claim that the contractual statements (promises) are performing acts in the sense that they have effects. This means that the existence of a contract between two or more entities imposes constraints on their behaviour and can produce outcomes that were not possible without a contract, mainly due to the performing nature of the statements or promises.

There are several types of contracts, but in this work only two are considered: generic **multilateral contracts** and **adhesion contracts**. An adhesion contract is a standardised contract form that in general is offered to consumers of goods and services without affording them a realistic opportunity to bargain and under such conditions that they cannot obtain the desired product or services except by accepting all the contract terms. Typical examples of this type of contracts are the ones that are signed between individual consumers and a big electricity supplier, or between a car owner and an insurance company. Multilateral contracts are normal contracts involving more than two parties in which all of them have equal bargaining power and, therefore, the negotiation process for contract formation is more complex.

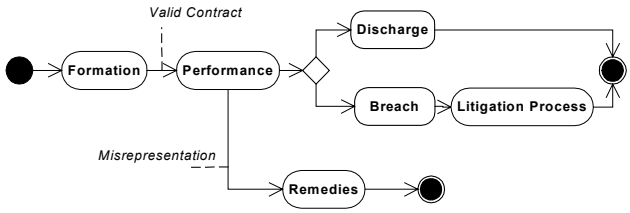

Figure 3-16 - Life-cycle of a contract

In every legal system there exists rules that identify the types of agreement that are to be treated as enforceable contracts. In the Common Law this is mainly done by the doctrine of consideration, and by formalities. **Consideration** was defined by an 1875 English decision as "*some right, interest, profit or benefit accruing to the one party, or some forbearance, detriment, loss or responsibility given, suffered or undertaken by the other*". **Formalities** guarantee that a contract is enforceable whenever the agreements are entered into in a certain form. This states that the validity of a contract must follow certain procedures. Figure 3-16 shows the life cycle of a contract. The most important phases are formation, performance (the execution phase), and termination. A valid contract exists if the formation phase went well. During the execution of the contract it might be detected that one of the parties caused misrepresentation, which may result in the offended party asking for remedies. The termination of a contract may occur either by discharge or by breach. A breach process is an abnormal termination and consequently involves a litigation process.

Formation. This is essentially a negotiation process in which the involved parties try to reach an agreement on the terms of the contract. The work that has been done in automated negotiation can be extremely helpful (Beam & Segev, 1997; Cortés & Dignum, 2001; Dignum & Sierra, 2001; Kraus, 1997; Lomuscio et al., 2001; Padget et al., 2002; Strobel, 2001). Automated negotiation is difficult, because of reasons amongst others, such as ontology variations and the need for a strategy. An ontology is required to assure that the agents are referring to exactly the same terms. On the other hand, a strategy is required to assure that the party gets the highest possible benefit.

In most law systems, at least two sequential statements are required to **create a contract**: an *offer* followed by an *acceptance*. An *offer* can be followed by a *counter-offer*, which in turn can also be followed by another *counter-offer* and so on. The process terminates when one of the partners sends an acceptance. The acceptance validates and gives life to the contract. The contract starts at the moment the acceptance reaches the offeror. An *offer*, once made, can be revoked before acceptance. An *offer* can also expire if a deadline for acceptance passes. If there is no specified deadline, then the offer expires in a "reasonable time", depending on the subject matter of the contract (Almeida, 2000; McKendrick, 2000). In the approach being followed in CoBASA an offer is made without specifying a deadline because prompt response from the enquired agents is assumed. This indicates that it must be answered in a “reasonable time”, which is the normal time-out imposed to the global architecture for communication among the agents. An offer that was rejected cannot be subsequently accepted.

An alternative is to reach an agreement other than the offer-acceptance protocol using joint contractual terms, which express the agreements of the parts in only one text. This modality is specifically used for creating contracts that involve more than two partners (multi-lateral contracts). In this case, the parts reach agreement on the final terms of the contract using different kinds of communicative acts in a preliminary phase. Afterwards, the final contract is put in written form (final agreement) and all the partners must subscribe to the contract. The contract becomes effective when the last partner subscribes to the document.

One final issue related to the formation phase is **misrepresentation** that happens when a party discloses untruthfully information to the other party in order to increase its negotiable position, or just to cheat him. A duty is imposed not to make any false statements of fact to the other contracting party and, thereby, to induce him to enter into a contract. When it is proved that misrepresentation has been carried out by one party it can ask for remedies. Remedies can be of two types: rescission or damages. Further details of this can be found in (McKendrick, 2000).

Performance. This phase corresponds to the execution of the contract. It is expected that during normal contract performance its terms be respected according to what was accorded in the formation phase. In the specific case of CoBASA this means that the agents are obliged to respect the conditions they have promised during the contract formation. To decide if a contract is being performed well, some kind of supervision of the operations related to the contract is necessary.

Termination. A contract might be terminated either under normal, or abnormal conditions. Contracts that terminate under normal conditions are known as terminated by discharge, which may occur under the following situations:

- *By performance* – The vast majority of contracts terminate under this type. In fact this is what happens when a contract reaches its validity, or when a buyer, for instance, receives what he has bought. The contract terminates naturally by being fully performed by the involved parties.
- *By agreement* – This is what happens when the contract was not fully performed but all the involved parties agree to terminate it.
- *By operation of law* – The most well known example of a termination by operation of law is termination by frustration. As (McKendrick, 2000) states, a "*contract is frustrated where, after the contract was concluded, events occur which make performance of the contract impossible, illegal or something radically different from that which was in the contemplation of the parties at the time they entered into the contract*". For instance, when an agent in CoBASA cannot perform its contract because of something that is not under its control it can ask to terminate its contract by frustration. When a contract is frustrated there is no liability for breach of contract because both parties have been provided with a *lawful excuse* for their non-performance.

A contract may terminate under an abnormal condition whenever one party does not perform its terms. In this situation the other party can breach the contract and is entitled to an appropriate remedy. As in (McKendrick, 2000) "*a breach is committed when a party without lawful excuse fails or refuses to perform what is due from him under the contract, or performs defectively or incapacitates himself from performing*". When a breach occurs there is a litigation process, during which the breach is

analysed to decide the reasons why the contract was breached and who is guilty. The innocent party is entitled to recover damages in respect of the loss, which he has suffered because of the breach.

Contracts should use non-ambiguous language to avoid future misunderstandings that, to be solved, need to be interpreted in a court. Because the natural language used in contracts suffers from ambiguity it would be helpful if the terms of a contract could be represented using a formal language, such as first order predicate language. Electronic contracts can avoid this problem if the terms of the contract are expressed using a formal language that by definition is unambiguous.

The typical sections that compose a contract are:

- ***Identification of the parties*** – This section identifies the parties that participate in the contract.
- ***Terms*** – A contract consists of a number of terms that correspond to the duties and obligations that must be performed by the parties. Terms are the statements that appear in the contract that will be considered for analysis, if there is a violation in the performance of the contract. There are two important classes of terms: warranties and conditions. A condition is an essential term of the contract. For instance, when a car is sold, the price and the roadworthiness are essential terms of the contract. In the case of CoBASA a condition is, for instance, the skills that each agent brings to the coalition. A warranty is a lesser important term. For instance, in the case of the selling of a car, a warranty may be the colour of the car. A breach of a condition enables the innocent party either to terminate the contract or affirm the contract, and then recover damages for any loss suffered. A breach of a warranty, on the other hand, does not allow the innocent party to terminate the contract although it might claim damages.
- ***Consideration*** – It was already defined at the beginning of this section.
- ***Exclusion clauses*** – A clause that appears to exclude or restrict a liability or a legal duty that would otherwise arise (Yates, 1978). The most frequent exclusion clauses are those that seek to exclude liability for breach of contract or for negligence.
- ***Termination clauses*** – These clauses ensure that either or both parties have the right to terminate the contract under certain circumstances. Generally, termination clauses describe breach of contract events that trigger the right to terminate the contract.
- ***Remedy clauses*** –These clauses state what rights the non-breaching party has if the other party breaches the contract. In contracts for the sale of goods, for instance, remedy clauses are usually designed to limit the seller's liability for damages.
- ***Arbitration clauses*** – An arbitration clause states that disputes arising under the contract must be settled through arbitration rather than through court litigation. Such clauses generally include the name of the organization that will conduct the arbitration (for instance, a consumers association), the city in which the arbitration will be held, and the method for selecting arbitrators.

3.4.2 How Computers are Changing the Law

Undoubtedly the dramatic dissemination of computers, computer networks, and information technologies in general affect the law. The various books that have been published dealing with this subject confirm such comments (Katsh, 1995; Susskind, 1998, 2000). Katsh, for instance, states that the "*future of law is not to be found in expressive buildings or books but in small pieces of silicon*". According to him law is now on a journey from a printing world to an electronic one.

Information technology is changing the law in three different directions:

1. As supporting tools – How information technology supports the operation of law institutions, and the law itself. This view accommodates, for instance, the use of computers to expedite the various processes involved in law systems, databases to permit lawyers to consult and search for past judicial cases, computer based infrastructures to permit separated people to be heard at courts, etc. People employ powerful tools to acquire, work with, and create information.
2. Change the Law – The law itself needs to be changed to accommodate the impact of information technologies in human behaviours, and consequently in one of the most important institutions that constrain human behaviour: the law. In (Katsh, 1995) it is stated that "*the new information technologies are particularly relevant to law because law is oriented around information and communication. Whatever definition one gives to the law, whether it is considered a profession, or a method of resolving disputes, or a process to bring about justice, or a facade to protect the status quo, or a means to secure rights and regulate behaviour, it is always concerned with information*".
3. As legal entities – Artificial agents are becoming actors in law related systems. This is what happens when artificial agents start to be involved in the process of buying and selling goods. Some research is currently being done in the area of whether a contract established by two artificial entities may still be valid (Allen & Widdison, 1996). As agents become increasingly autonomous, which from a legal point of view is evidence of free will, the implications of machine autonomy for contracting become obviously important.

Law systems must be adapted to cope with the new situation of ubiquitous computing and a digital world. The most important computer related issues that need to be solved by law are: copyrights, the way electronic contracts are formed, electronic legal entities, and how to cope with a world where information and knowledge, which are less stable and permanent, are becoming more and more important at the expense of tangible things. On the other hand, computer science also needs to be able to create artificial agents that must be conforming to the law.

3.4.3 Electronic Contracts – state of the art

The driving force behind electronic contracts is business, and in particular the area of electronic commerce. Therefore, almost all the work on electronic contracts comes from researchers in this field

(Angelov & Grefen, 2001). An exception is the work of this thesis, which pioneers the use of contracts to regulate the behaviour of artificial agents in a manufacturing environment.

As business interactions become more dynamic and involving heterogeneous companies, the use of traditional paper contracts becomes an important constraint for these business relationships. Electronic contracts might improve the effectiveness and efficiency of these relationships. However, it should not be assumed that electronic contracts can be obtained by just mapping paper contracts to electronic contracts. Important points that need to be addressed when discussing electronic contracts include:

- ***Meta information*** – Since electronic contracts are to be processed by artificial agents and diversified in their content, it is necessary to include additional information to help computers understand the context within which the information should be understood because it is not possible to create completely standardised or fixed content contracts in all situations. This additional machine readable information can be XML (Lenz, Oberweis, & Schneider, 2001; XML, 2000) or any other format suitable to provide computer based meta information. Not all electronic contracts are human readable. This is what happens within CoBASA in which contracts are modelled using the frame paradigm (Hayes, 1980; Minsky, 1975; Noy et al., 2000). Which ontology is used is one meta information that should be included in order for all parties to have a common understanding of the concepts used.
- ***Standardised structure*** – To optimise the speed and efficiency at which contracts are established it is important to develop mechanisms to improve their reusability. By having standard components it is possible to reuse them as needed. Different modelling approaches can be used. XML, for instance, seems to be a good candidate because it is quite easy to generate XML documents and process them, despite its lack of semantics. Several methods are used, as defined in (Angelov & Grefen, 2001):
    - *Standard Contract Clauses (SCC)* – In this case some of the clauses and terms of the contract are standardised, and are instantiated as needed.
    - *Contract Templates (CT)* – These are predefined contracts that can be used as a basis for new instances. Templates define essentially the structure and they are not considered as the final result of the contracting process. This is the approach used in CoBASA that, for one of the two different types of contracts, has templates in which the specific terms are negotiated.
    - *Standard Form Contracts (SFC)* – These are contracts that do not change in every instantiation. These are the contracts typically offered by a powerful partner on an all or nothing basis. In the legal jargon they are known as *adhesion contracts*.
    - *Partially Filled Contracts (PFC)* – These act as templates in which some of the data is preliminary filled.
- ***Actors*** – Electronic contracts can be performed between machines or between machines and humans. It can involve direct negotiation between the parties or using intermediaries. In the

case of CoBASA the two approaches are followed. Some contracts use direct negotiation while others use a broker as an intermediary.

- ***Topology of the negotiation*** – The possible contracting situations are: 1) one to one, 2) many to many, 3) one to many, 4) compound topology, which is a composition of the previous ways.
- ***Ontologies*** – The ontology under which the contractual concepts apply should always be defined, understood or interpreted.

The protocols, languages specifications, frameworks, and organisations that are relevant in the context of electronic contracts are listed below:

- ***UN/CEFACT*** (UN/CEFACT, 2003) – This is the United Nations body to develop and promote solutions for the facilitation of global business processes. One well known standard that emerged out of this organisation is EDIFACT (UN/EDIFACT, 2003), which is a standard widely used for electronic data interchange.
- ***Organisation for Advancement of Structured Information Standard - OASIS*** (*OASIS*, 2003) – It is a non-profit organisation whose mission is to drive the development, convergence, and adoption of structured information standards in the areas of e-business, web services, etc. Relevant for contracts is the technical committee *legalXML* whose purpose is to develop open XML standards for the markup of contract documents.
- ***CommerceNet*** (*CommerceNet*, 2003) – The CommerceNet is an open, interoperable network linking many commerce communities through a common architectural framework.
- ***XML/EDI Group*** (*XML/EDI*, 2003) – This group tries to combine the advantages of a standard like EDI (Electronic Data Interchange) with the advantage of a web-wide usage of the standard.
- ***Trading Partners Agreement Markup Language – tpaML*** (Dan et al., 2001; Sachs et al., 2000) – Submitted to OASIS, the tpaML specification enables companies to standardize their way of exchanging contracts with their trading partners.
- ***XML Common Business Library – xCBL*** (xCBL, 2003) – It is a set of XML building blocks and a document framework that allows the creation of robust, reusable, XML documents to facilitate global trading.
- ***Universal Business Language – UBL*** (UBL, 2003) – This is one of the technical committees of OASIS. UBL is intended to become an international standard for electronic commerce freely available to everyone without licensing or other fees by developing a standard library of XML business documents.
- ***Electronic Business XML – ebXML*** (Chiu, 2002; ebXML, 2003; Kotok & Webber, 2001) – It is sponsored by UN/CEFACT and OASIS. It is a modular suite of specifications that enables distributed companies to conduct business over the Internet.

The following areas have been active in the research on contracts:

- ***Generic contract frameworks*** **-** This topic covers the aspects related to generic descriptions of the research problems associated with contracts. Relevant work in this area are (Barata & Camarinha-Matos, 2002, 2003; Gisler, Stanoevska-Slabeva, & Greunz, 2000; Goodchild, Herring, & Milosevic, 2000; Lindemann & Runge, 1997; Milosevic & Bond, 1995).
- ***Negotiation*** **-** It covers works on the negotiation aspects that are required to create a contract. The vast work on negotiation from the multiagent domain has also been used for contract negotiation. Most of the negotiations in electronic contracts follows auction theory (Kambil & van Heck, 1998; Kersten et al., 2000; Klein, 1997; McAfee & McMillan, 1987; Milgrom, 1989; Strobel, 2000; Teich, Wallenius, & Wallenius, 1999; Vickrey, 1961; Wurman, Walsh, & Wellman, 1998). Two approaches may be followed when developing artificial agents (intelligent agents) for negotiation. The first considers that the agents should be taught the effective strategies of negotiation. The second considers that the agents require rather little teaching because the focus of the negotiation is on environmental rules that the agents need to follow. Relevant work on negotiations in electronic contracts include (Bichler, Beam, & Segev, 1998; Dan et al., 1998; Daoud, 1998; Lindemann & Runge, 1997; Milosevic & Bond, 1995; Reeves, Wellman, & Grosof, 2001).
- ***Contract life cycle*** **–** It covers works on the contract life cycle, either by defining it or by presenting solutions or methodologies to cover the phases (Gisler et al., 2000; Goodchild et al., 2000; Griffel, Boger, Weinreich, Lamersdorf, & Merz, 1998; Lee, 1998; Lindemann, 1997; Milosevic & Bond, 1995; Runge, 1998; Runge, Schopp, & Stanoevska-Slabeva, 1999; Schmid & Lindemann, 1998).
- ***Contract modelling*** **-** Some research work has been done on finding the right computer model for use in automated contract systems. Some contracts are modelled using XML (Gisler et al., 2000; Goodchild et al., 2000; Greunz, Schopp, & Stanoevska-Slabeva, 2000; Koetsier, Grefen, & Vonk, 2000). Furthermore, in (Griffel et al., 1998) an UML class diagram is used to describe a model for a generic contract that is implemented with XML. CoBASA follows a different approach due to the fact that, although its contracts are represented in UML, they are implemented using a frame-based tool (Barata & Camarinha-Matos, 2002). Petri Nets are used in (Lee, 1998) to capture the temporal/dynamic aspects while in (Reeves, Wellman, & Grosof, 2000; Reeves et al., 2001) contracts are represented using Courteous Logic Programs (CLP), which is a logic programming language suited to express business contracts (Grosof, Labrou, & Chan, 1999).
- ***Implementation*** **–** It covers specific implementations of contracts (Barata & Camarinha-Matos, 2003; Bichler et al., 1998; Griffel et al., 1998).
- ***Contract performance*** **–** In (Daskalopulu & Maibaum, 2001) a service for automated contract execution, based on modal action logic and deontic action logic, is introduced.
- ***Formal languages for contracts*** **–** This is the research work on formal languages to specify contracts. A well-formed contract document is one that accurately implements the agreement

on which the parties negotiated. In (Daskalopulu, 1998) four approaches are presented that address the problem of well formed contract documents: logic programming, normalised legal drafting, case analysis, and normative positions. The use of logic in legal reasoning and modelling is discussed in (Kowalski, 1995; Prakken, 1997; Prakken & Sartor, 2001; Sergot, Sadri, Kowalski, Kriwaczek, & Cory, 1986). The work presented in (Jones & Sergot, 1992, 1993; Prakken, 2001, 2002; Prakken & Sartor, 1996; Prakken & Sergot, 1997) also deals with formal systems in contracts.

3.5 SHOP FLOOR CONTROL

The Shop Floor is an integral part of any manufacturing system. It is here that the needs of the customer and the efforts of a manufacturing firm are inter-linked by combining raw materials, processes, and labour to make the desired finished product. Consequently, it is important to have effective means for controlling the activities of the shop floor in order to achieve the greatest efficiency and predictability of the manufacturing process.

Shop floor control occurs at a different abstraction level or levels. The actions in each control level are then decomposed into sub-actions or commands that can be further decomposed. The important aspect to keep in mind is not so much the kind of hierarchical nature of these levels but instead the presence of different conceptual levels of control, which, in fact, correspond also to different levels of responsibility. Of importance, in terms of control, are also the time frames these events follow, some being real-time and others week-based.

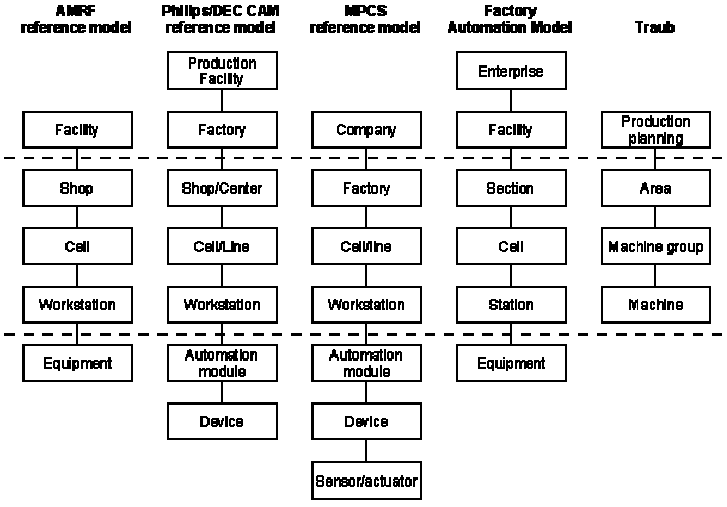

Figure 3-17 - Reference models for production control. Source (Zwegers, 1998)

Figure 3-17 shows several reference models (AMRF (Jones & McLean, 1986), CAM (Philips, 1987), MPCS-Manufacturing Planning and Control Systems (Biemans & Vissers, 1989), Factory Automation Model (Graefe & Thomson, 1989), and TRAUB) and their different control or responsibility levels.

The higher control levels are responsible for production planning functions, while the lower levels for equipment control functions. Although in the context of this section it is not important to discuss in detail these reference models, the figure is shown to illustrate that several control levels exist and must be taken into account in any shop floor control architecture. The reference models are noticeably hierarchical and are associated with hierarchical control solutions. CoBASA, from the shop floor control point of view, is focused at the cell level, but it may be scalable to higher levels.

Four main functions in shop floor control are identified in (Hopp & Spearman, 2001):

1. ***Coordinating the manufacturing resources*** (material, knowledge, humans and information) on the shop floor. Material flow control, which is a fundamental activity in most systems, falls under this category. This function provides a mechanism that decides which job to release to the factory, which job to work on at the individual workstations and what material to move between workstations.
2. ***Real time control***. The scheduling, dispatching and monitoring of real-time operations. This is primarily the control of non-interruptable operations (NIOs) executed by the system hardware (robots, PLCs, sensors, etc.).
3. ***Capacity feedback***. This involves the collection of data to update capacity estimates so as to ensure consistency between high level planning modules and low-level execution ones.
4. ***Quality control***. Testing should be added in which the operation just executed is checked for validity that is, for instance, the approach largely used in the electronics industry.

Definition 3.5 – Shop floor control. Adapted from (Wyns, 1999)
Shop floor control is that group of activities directly responsible for managing the transformation of planned order into a set of outputs in addition to those indirect activities that should be done to cover the changes or adaptations (reengineering phase) of the control solution in an easy way. During the operative phase it governs the short-term (on-line) detailed planning, execution, and monitoring of process preparation and resource allocation activities needed to control the flow of an order from the moment that it is released by the planning system for execution until the order is filled and completed.

Scherer describes in (Scherer, 1998) the characteristics for a good shop floor design. He notes that in order to achieve control within the shop floor, the designer's goal should be that of developing a dynamic and flexible organization as opposed to finding an optimal design.

3.5.1 Existing Shop Floor Reference Architectures

Figure 3-18 shows the evolution of shop floor control architectures according to its temporal evolution rather than applicability in industry. An interesting aspect of this evolution is the increasing level of autonomy in the controllers and the relaxation of the master-slave relationships. It can be noticed that

in the centralised approach the executing elements are merely slaves of the controlling element, and this tendency is decreased in the hierarchical approach where the nodes have more autonomy. Finally, in the multiagent case, the agents have maximum autonomy and the relationships between them are not characterised by master-slave relationships. Another aspect is the dynamics of the relationships established between the various controllers. Multiagent architectures are characterized by dynamic relationships, which means that different kinds of control architectures can be established by merely reorganising the way the agents interact, by using different techniques, such as negotiation or contractual relationships. This is what happens in CoBASA.

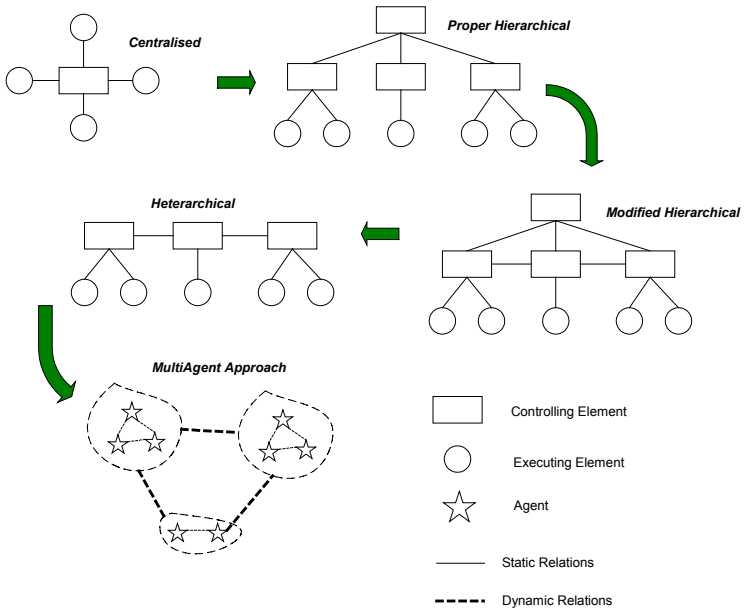

Figure 3-18 - Evolution of shop floor architectures. Adapted from (Dilts, Boyd, & Whorms, 1991)

3.5.1.1 Centralised approach

Advantages	Disadvantages
1. Simpler coordinating algorithms	1. Difficult to modify. Any modification implies immediately big changes in the program 2. Difficult to extend. Introducing any new element also implies big changes in the program. 3. Complex control logic. Everything is centralised in the central node. Therefore the control logic is complex. 4. Error prone due to its dependence on the central node.

Table 3-2 – Advantages/disadvantages for the centralised approach

Interestingly, before the advent of computer dissemination through industry, the prevailing approach was decentralised since there was no notion of supervision or coordination among the various islands of automation. When IT started to invade the manufacturing world and started to be perceived as the

right technology to integrate the various tasks and activities within the shop floor, the centralised approach immediately emerged as the dominant control architecture.

3.5.1.2 Proper Hierarchical

To overcome the disadvantages of the centralised architecture, the proper hierarchical form was introduced (Figure 3-18), in which the complexity of the centralised node is decentralised over several other nodes organised in a hierarchical way. The level of decision varies according to the position of the controller in the hierarchy (higher level nodes have higher level decisions). This kind of organisation reduces the level of complexity because instead of a central node containing all control logic it is distributed among the various nodes in the hierarchy. Each node is simultaneously the coordinator of the lower level nodes and coordinated by the higher level ones.

In this approach, the autonomy level of the nodes are increased when compared to the centralised approach. At the top of the hierarchy there is a single controller that sets the global goals and the long-range strategies. These strategies are then accomplished by sending commands to the lower levels, which in turn decompose these commands into new ones that are then sent to a further lower level. The information detail increases with level lowering, whereas the time period for its consideration decreases.

Advantages - based on (Dilts et al., 1991; Zwegers, 1998)	**Disadvantages -** based on (Conant, 1976; Dilts et al., 1991; Hatvany, 1985)
1. Reduces the functionality and complexity of each individual controller. 2. Can be modified. It is possible to integrate new nodes in existing layers. 3. Works near optimal performance under stable situations. 4. Adaptive behaviour. It is possible to obtain feedback from lower levels and use it to close the feedback loop.	1. Difficult to introduce new layers. While it was possible to add new nodes into the existing layers, changing the layers is very complex – structural rigidity. 2. Difficult to evolve due to its structural rigidity. 3. Respond badly to unstable situations such as rush orders or machine breakdowns. Machine breakdown affect severely the system. The entire system might be in danger since the higher levels need the information from the damaged branch to make decisions. 4. Information overload because the controllers need to constantly filter information from higher and lower levels. 5. Poor fault tolerance. This is related to the explanation already given in point two.

Table 3-3 – Advantages/disadvantages for the proper hierarchical

Relevant architectures

1. ***AMRF – Automated Manufacturing Research Facility*** (Jones & McLean, 1986) – The NIST reference architecture. It has five layers: facility, shop, cell, workstation, and equipment. Although longer term planning is possible in the higher levels only the lower three layers have been implemented.

2. ***PAC – Production Activity Control*** (Bauer, Bowden, Browne, Duggan, & Lyons, 1991) – It was described in chapter 2. The structure of PAC was improved by Maglica (Maglica, 1996a, 1996b) and called PAC+. This new architecture was applied within FACE – Flexible Assembly Control Environemnt (Onori, 1996) in the Mark III FAS system developed at KTH (Onori, Langbeck, & Grondahl, 1997).

3.5.1.3 Modified hierarchical

Nodes with little autonomy increase information overload since many abnormal situations detected by a node, which cannot be solved with the information it obtains from its lower level nodes, imply a request for help from its higher-level node. If this higher level has enough information the node makes the decision and the process stops. If, on the other hand, the node does not yet have the necessary information a higher level is further called causing more information overload. This problem can be reduced if, at each level in the hierarchy, the nodes are allowed to communicate with the nodes of the same level because, whenever an abnormal situation occurs, it is possible that the situation might be solved with the information gathered at that level, without requiring the attention of the higher levels.

The modified hierarchical architecture is essentially characterised by vertical relations (hierarchies) and horizontal peer-to-peer relationships (Figure 3-18). This architecture moves the level of decision making authority down to the level where the available information is sufficient to solve the problem rather than bringing it up to the point where it is just overhead. The master-slave kind of relationship between the higher and lower levels still exists but is reduced since the degree of autonomy for each node has increased. A controller acts as an intelligent assistant to the host and not as a slave.

Advantages	Disadvantages
1. Reduces information overload. 2. Reacts faster to abnormal situations.	1. More complex nodes, which must be able to cooperate with their horizontal peer nodes. 2. The same disadvantages presented by the proper hierarchical architecture associated with the hierarchical aspects.

Table 3-4 – Advantages/disadvantages for the modified hierarchical approach

Relevant architectures

1. ***NASREM – NASA/NBS Standard Reference Model for Telerobot Control System Architecture*** (Albus, Lumia, Fiala, & Wavering, 1989) – This control architecture, which was developed at the NIST, is based on the AMRF model but focuses on teleoperation.
2. ***FACT – Factory Activity Control Technology*** (Arentsen, 1995) – It was developed by the Laboratory of Design, Production and Management at the University of Twente-Netherlands. It is based on four hierarchical levels: company (client order and master production planning),

factory (capacity planning on the basis of the available manufacturing resources), cell, and station.

3. ***CHAMP – Chalmers Architecture and Methodology for Flexible Production*** (Gullander, 1999; Lennartson, Fabian, & Gullander, 2002) – See chapter 2.
4. ***CCP's Manufacturing Cell Controller*** (Leitão & Quintas, 1997) – See chapter 2.

3.5.1.4 Heterarchical control architectures

The heterarchical approach has been mainly developed by Neil Duffie (Duffie, 1990; Duffie, Chitturi, & Mou, 1988; Duffie & Piper, 1986, 1987; Duffie & Prabhu, 1994, 1996) based on earlier findings of József Hatvany (Hatvany, 1985, 1990). Hatvany in (Hatvany, 1985) states that:

> *"Highly centralised and hierarchically ordered systems tend to be rigid, constrained by their very formalism to follow predetermined courses of action. However carefully "optimised" their conduct may be, it has been shown that this very property of inherent resistance to organisational change itself necessarily leads in due course to catastrophic collapse. (...) On the other hand, fragmentation of a system into small, completely autonomous units, each pursuing its own selfish goals according to its own, self-made laws, is the absurdity of primitive anarchy. (...) In the place of either of these, we suggest cooperative heterarchies."*

From the previous statement it is clear that the most important characteristics of a heterarchical system are the absence of hierarchies and the autonomic cooperative nature of each node, which typically represents resources and/or tasks from the shop floor. Flatness, cooperation, distribution, and autonomy are the keywords in this approach. Because of the autonomy of its nodes, control decisions beyond the scope of each individual node can only be reached through negotiation between all the participating nodes. In this approach the decision making is confined as far as possible to the individual agents or as Duffie pointed out (Duffie & Piper, 1987) "where the information originates". This avoids the overhead and complexity of having flow of information through the various nodes in the hierarchy. In addition, a local change has no influence on the other controllers, at least theoretically. Therefore any unpredictable event caused by various disturbances can be corrected rapidly and efficiently through negotiation. For instance, when an agent is not working it just does not participate in the negotiation and, for all the other agents, the faulty state of this agent is transparent.

When it was first introduced in the 1980s, the heterarchical approach was quite advanced for its time and required a completely new way of thinking that companies were not yet ready for because they were too tied up with the hierarchical model. Furthermore, the technology was not yet mature enough to provide the right solutions to implement it. It was difficult to convince manufacturing professionals that heterarchies could be advantageous over hierarchies. This started to change when recognised business management gurus such as Peter Drucker claimed that organisational hierarchies had grown too tall and consequently organisations should adopt a flatter, more responsive, and more knowledge oriented architecture (Drucker, 1988, 1990, 1992). These organisational changes, in

conjunction with the dissemination of distributed computer systems, rendered the heterarchical approach much more appealing.

In the current state of the art, the most suitable technological paradigm to implement the heterarchical architecture is the agent-oriented software engineering. The importance of distributed systems and particularly the area of agent-oriented systems and/or multiagents, has led to the development of an important related research area in the shop floor control domain: multiagent systems for shop floor control. This area in a way has dimmed the heterarchical architecture because it uses most of the concepts introduced by the heterarchical approach.

Disadvantages – based on (Bongaerts, Monostori, McFarlane, & Kádár, 2000; Dilts et al., 1991; Duffie & Piper, 1986; Van Brussel, Wyns, Valckenaers, Bongaerts, & Peeters, 1998; Wyns, 1999)
1. Problems with optimisation. The independence of the nodes without hierarchy makes it difficult to optimise its performance due to some redundancy and difficulty in optimising the best approach without global information. 2. Chaotic behaviour can occur. The behaviour of the system can be unpredictable because it lacks a central entity to supervise and conduct what is happening. 3. The system performance depends very much on the negotiation rules that have been defined for task allocation. In fact these rules have direct impact on load balancing between the different resources that are associated with the nodes that compete for tasks. 4. Impossible to predict the behaviour of individual orders. This happens because there is no fixed control strategy that depends on the negotiation for task allocation. 5. When it was first introduced, the technology was not yet fully matured to support its software and hardware requirements. This aspect was a serious drawback for its dissemination. 6. Although its programming complexity can be lesser than in the case of a hierarchical solution, it is still high. This happens because there is a changing in the programming paradigm that requires a new *way of thinking*. 7. It is better implemented in applications that have a homogeneous set of resources.

Table 3-5 – Disadvantages for the heterarchical approach

Advantages – based on (Duffie, 1990; Duffie & Piper, 1987; Duffie & Prabhu, 1996)
1. The overall complexity is reduced for the programmer. By eliminating the hierarchical levels and relying on the autonomy of the modes the global software architectures becomes simpler. 2. Reacts faster to abnormal situations. It must be pointed out that with hierarchical systems it is possible to achieve faster response times because of the rigid structure of the hierarchical system. However this only happens under normal conditions. 3. Improved fault-tolerance since a failure node does not have impact on the control strategy. 4. Reduced software development costs by eliminating supervisory levels. 5. Higher scalability. Due to improved modularity, the addition or removal of nodes is quite straightforward. On the other hand because the nodes are independent and autonomous changing them does not have significant impact on the other ones.

Table 3-6 – Advantages for the heterarchical approach

Relevant architectures

There are very few implementations of the heterarchical approach in the form proposed by Hatvany and Duffie. The most relevant implementations have been done by Duffie at the University of Wiscosin-Madison (Duffie, 1990; Duffie & Piper, 1987; Duffie & Prabhu, 1994, 1996).

The relevance of this architecture relies much more on the concepts that were introduced and later used by multiagent approaches than on effective industrial applications.

3.5.1.5 Holonic manufacturing

Holonic manufacturing has its origins in the Holonic Manufacturing Systems (HMS, 2000), which is an IMS – Intelligent Manufacturing Systems (IMS, 2000) project. Holonic manufacturing aims at developing reconfigurable, scalable, flexible, and responsive manufacturing systems. These are requirements that CoBASA also aims to deal with. However, while the requirements for holonic manufacturing systems were mainly developed to take into consideration the life cycle phase related to the operation of the system (operative phase), CoBASA was developed to deal both with this phase and the reengineering phase.

The basic concept behind the holonic approach is the ***holon***, which was first introduced by the Hungarian philosopher Arthur Koestler in 1967 (Koestler, 1989) to describe a basic unit of organisation in biological and social systems. The word *holon* is originated from the Greek language and it is a combination of the words ***holos***, which means whole, and the suffix ***on***, which means a particle or part.

The problem Koestler wanted to address was how to classify those concepts in a hierarchy that is simultaneously a part and a whole. When analysing a complex structure it is likely that this structure is composed of parts. If each of these parts is analysed in detail it is quite easy to see that they are simultaneously a part of something (the higher level) and a whole, when considered from the point of view of its composed parts. While part and whole suggest something absolute, the fact is that things are not so absolute because a part can be simultaneously a part of something and a whole for its parts. Koestler used the word ***holon*** precisely to describe this hybrid nature.

A *holon* in the manufacturing context according to (Van Brussel et al., 1998) is:

> *"An **autonomous** and **cooperative** building block of a manufacturing system for transforming, transporting, storing and/or validating information and physical objects. The holon consists of an information processing part and often a physical processing part. A holon can be part of another holon."*

Autonomy and willingness to cooperate immediately suggest that HMS can be successfully implemented using multiagent systems. There is an evident similarity between a *holon* and an agent as defined in (Wooldridge & Jennings, 1995). The set of cooperating *holons* in the manufacturing system form the multiagent system. This comparison between *holons* and agents is also expressed in

(Bongaerts, Van Brussel, & Valckenaers, 1999). Taking this similarity into consideration from now on the terms *holon* and agent will be used inter-changeably. It is interesting to contrast HMS with the heterarchical approach because although a holonic system is based on the distributivity of the nodes, as in the heterarchical case, it assumes that hierarchy is fundamental to overcome some of the limitations of the heterarchical approach, namely that its behaviour is unpredictable (Bongaerts et al., 2000).

The PROSA architecture (Van Brussel et al., 1998) is considered to describe the type of *holons* that can be found in a HMS, because it is the most well known, even if some others exist such as the one developed by (Heragu, Graves, Kim, & Onge, 2002). PROSA considers the following types of agents:

- ***Product*** – This is the agent that has knowledge about the product and the process to manufacture it. It contains the product model.
- ***Resource*** – This is the agent that represents production resources such as robots or CNC machines. The agent contains two distinct parts. One deals with the manufacturing resource (on-line control in Figure 3-19), while the other is the information processing part that controls the resource. This is the agent responsible for the effective execution of the process.
- ***Order*** – This agent represents a task in the manufacturing system. It is responsible for performing the assigned task correctly and on time.

These three agents form the basic set of agents also known as ***local agents***. The local agents negotiate among them to achieve their production goals, forming, in this way, a heterarchical approach since they are autonomously cooperating agents. Another type of agent known as the ***staff agent*** composes the PROSA architecture. An agent of this type acts as a centralised agent, which is used to coordinate the agents of the other three types. With the addition of this central agent the architecture becomes hierarchical.

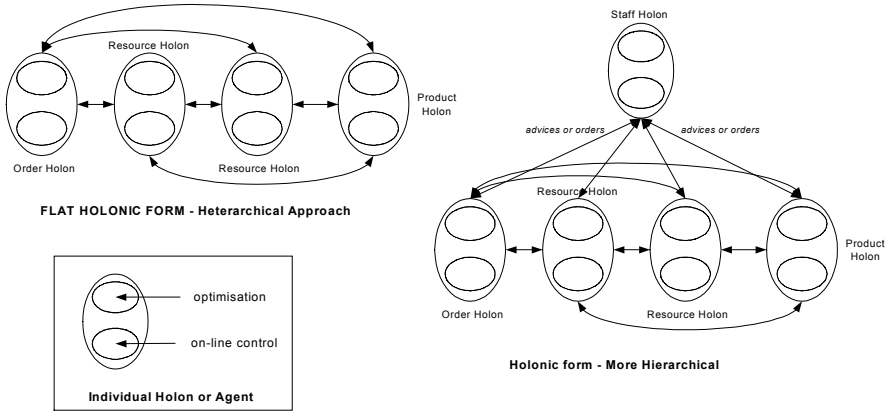

Figure 3-19 - Holonic manufacturing holons. Adapted from (Bongaerts et al., 2000)

Figure 3-19 shows the duality between hierarchy and heterarchy in PROSA (Figure 3-19). The type of relation established between staff agents and local agents determines the type of architecture. If the

messages sent by the staff agents to the local agents are considered merely as advice then the architecture is more heterarchical, because in this case the local agents have greater autonomy and depend much less on higher levels. If, on the other hand, instead of advice the staff agent sends orders then the architecture is more hierarchical.

Advantages	Disadvantages
1. It is possible to choose between a heterarchical and a hierarchical approach. 2. Scalable, modular, and reconfigurable. 3. Fault tolerant.	1. Not developed to cope with life cycle phases other than the operative phase. 2. Poor industry acceptance because it is not so easy to implement its architecture. 3. People not used to the paradigm. 4. Despite some effort in the last years the architecture for the local holons is not yet recognised as a pure agent architecture by the leading experts in the multiagent and distributed AI fields.

Table 3-7 – Advantages/disadvantages for the holonic approach

3.5.1.6 Multiagent based approaches

The dissemination of multiagent applications in industry is highly conditioned by the traditional resistance industrialists have to unproved technology in terms of cost and speed, which they prefer to an elegant or sophisticated solution. Only after convincing them that the multiagent paradigm is not just another fashionable buzz word but it is, instead, a promising technology to successfully implement solutions for many of the requirements imposed by agile manufacturing systems, will it be possible to disseminate this paradigm in industry. However, there still is a long way for this dissemination to occur because the development of applications that respect the requirements of speed, user friendliness, and reliability needed by industry is not yet completely available.

There are several good surveys about the use of multiagent systems in manufacturing. (Shen & Norrie, 1999) is one of the most cited, which makes an extensive list of multiagent architectures classified according to the different manufacturing areas for which multiagent applications have been developed. (Heragu et al., 2002) is also an overview of agents in the manufacturing environment and makes a comparison between hierarchical, heterarchical, and hybrid approaches using multiagent systems. Parunak in (Parunak, 1998, 1999) also describes some multiagent applications in manufacturing systems.

According to (Shen & Norrie, 1999) agents in manufacturing systems are used to:

- Encapsulate existing software systems. This is a technique used to integrate legacy systems, because it is the only way to integrate existing tools into a multiagent system.
- Represent manufacturing resources such as machines, workers, etc. This is what CoBASA and the holonic manufacturing approach do.

- Model special areas of activity such as planning and scheduling (Rabelo, 1997; Rabelo, Afsarmanesh, & Camarinha-Matos, 2000).
- Model special services such as facilitators, mediators, and brokers as well as other auxiliary activities. Examples are the CoBASA architecture (Barata & Camarinha-Matos, 2003), which uses a broker agent to create coalitions, the holos methodology (Rabelo, 1997; Rabelo, Camarinha-Matos, & Afsarmanesh, 1999), and the MetaMorph architecture (Maturana, Shen, & Norrie, 1999) that uses a mediator.

Although multiagents have been applied to develop solutions to different manufacturing areas, the area with most applications has been upper planning and scheduling level of control. Little work has been done to the lower level of control, the exception being CoBASA, the work of Leitão and Restivo, which have developed an architecture for shop floor control combining an hierarchical approach with a heterarchical one (Leitão & Restivo, 2002), and the work of Brennan, Fletcher and Norrie, which have developed an architecture for reconfiguration of real time control systems (Brennan, Fletcher, & Norrie, 2002). Until recently the agents were fixed in the sense that they were kept in the same computational infrastructure during their full life cycle. However, this situation is changing and the mobile agents paradigm is increasingly used to implement manufacturing solutions, namely in telesupervision and teleoperation (Camarinha-Matos & Vieira, 1999), and reconfiguration (Fletcher, Brennan, & Norrie, 2003).

As Parunak states "*Agents are not a panacea for industrial software. Like any other technology, they have certain capabilities, and are best used for problems whose characteristics require those capabilities*" (Parunak, 1998). The following types of manufacturing applications are well suited to agents: modular, decentralised, changeable, ill-structured, and complex (Parunak, 1999).

Modular. Agents are best suited to applications whose structure is naturally modular because it becomes easier to decide the agents that compose the system. Two approaches are usually followed to decide which agents form the system: functional decomposition , and physical decomposition. In the functional approach agents encapsulate modules assigned to functions, while in physical decomposition agents are used to represent physical entities such as manufacturing resources. The functional decomposition tends to be more complex since agents that represent functions need to share many of the state variables that cover different functions, and consequently the consistency of those shared state variables needs to be maintained. On the other hand, in the physical decomposition, the agents are easier to implement since their state variables are individualised and confined to the agents.

Privileging physical decomposition leads to more modular applications since whenever it is necessary to move a manufacturing resource modelled by an agent from one place to another, it can be moved without changes in the agent's code. In addition, changes in the manufacturing resource that imply changes in the agent's code are only confined to the agent, without disturbing or changing the

code of the other members of the multiagent system. This results in better reconfigurability and better reusability of manufacturing components.

Decentralised. Agents are entities that represent decentralised systems in a natural way because they are, by definition, autonomous pro-active entities that pursue their goals by doing actions under their own control rather than being externally activated or ordered. Consequently they are particularly suited for applications that can be decomposed into independent autonomous tasks. The relevant problem associated to decentralisation is coordination. The agents might be organised using a centralised, hierarchical, federation (with facilitators, brokers, and mediators), or completely distributed approach (heterarchical).

Changeable. This should be also called modifiable or reconfigurable because this means that the manufacturing problem under study requires easy changes or adaptations (modifications). The agents' ability to deal with changeable problem is due to their modular and decentralised nature. Modularity allows that any change is done in clearly identified pieces of software (agents). Decentralisation implies that the impact of any change in the agents has little impact in the behaviour of the other manufacturing agents.

Ill-Structured. An ill-structured problem is one from which full information cannot be retrieved, and consequently, is not possible to fully structure it. Unfortunately more and more ill-structured problems can be found in today's manufacturing environment because poorly structured problems are closely associated to unpredictable dynamic environments, which characterise the current manufacturing situation. Traditional programming paradigms cannot cope effectively with ill-structured information because these kinds of paradigms require knowing, in advance, all the possible interactions between the various software components. The agent paradigm is better indicated for this kind of problems because it is possible to develop agents that are independent of any specific knowledge about the other agents. In a multiagent environment the knowledge of the agent about the world can be focused on its external environment rather than on the other agents.

Complex. Complexity is an inherent characteristic of modern manufacturing systems, and several approaches have been used to deal with it. In the traditional approach the problem is analysed in its various perspectives, a complete model is created, and all behaviours in the system are codified. The big problem with this approach is that, as the number of behaviours increases, the complexity of how to coordinate them also increases. Furthermore, adding any new set of behaviours will have an impact within the code of the already developed behaviours (scalability problem). Another approach is considering that complex behaviour can be achieved by the interaction of entities with fewer and simpler behaviours as happen with ants. This is related to emergent behaviour (Johnson, 2001) or swarm intelligence (Tarasewich & McMullen, 2002). Agents are good at implementing emergent

behaviour because of their decentralised nature and communication ability. A complex problem can thus be tackled by considering that it can be decomposed into a set of simpler interacting agents. Instead of being concentrated on thousands of rigid behaviours that represent the whole complex problem, it is possible to concentrate on individual agents that show fewer and simpler behaviours.

For the agent paradigm to be widely accepted in the industrial world it is mandatory to have tools and methods to guide the engineers while developing multiagent solutions. The engineers working in the industry are and must be specialists mainly in the manufacturing process because they are there to produce the best possible product, on time and at a reduced cost. If they are overwhelmed with software implementation details they will never create the effective solutions industry needs and is looking for.

3.6 COLLABORATIVE NETWORKED ORGANISATIONS

Although the concept of collaborative networked organisation is sometimes confused with the concept of virtual organisation (VO), they are not the same because the latter is just one type of the former. Collaborative networked organisations are not circumscribed to the business domain.

Definition 3.6 – Collaborative networked organisation

A collaborative networked organisation is any group of autonomous entities, which may be organisations, people, or artificial agents that have together formed a cooperative dynamic network to reach individual or group benefits.

With this definition different types of collaborative organisation can be considered. For instance, the work on coalitions that was discussed under agents is a perfect example of a collaborative organisation. Camarinha-Matos in (Camarinha-Matos, 2002a) considered the following forms:

1. ***Virtual Organisation (VOs)*** – It includes both business and non-business organisations. Example of the latter is the formation of humanitarian relief operations, military operations, and governmental and non-governmental organisations.
2. ***Professional Virtual Communities (PVCs)***– These are organisations of people that share the same professional interests. The members are bound to some social rules resulting from the commitment of their members to the PVC that constrain their behaviour. From within PVCs virtual teams can emerge to deal with subjects relevant to the organisation, for example, ethical issues.
3. ***E-science*** – These are communities of scientists that use virtual labs to support their work.

4. ***Advanced supply chains*** – New advanced supply chains are composed of companies whose relationships between them are more equalitarian than based on the predominance of one over the others as it was the tradition.

Considering that the life cycle of a VO is composed of VO planning and creation, VO operation, and VO dissolution the main difficulties associated with each phase are (Camarinha-Matos, 2002a):

- ***VO planning and creation*** – Some of the obstacles include the lack of appropriate support tools, namely for: partners search and selection (e-procurement), VO contract bidding and negotiation, competencies and resources management, task allocation.
- ***VO operation*** – The main challenges here are: well-established distributed business process management practices, monitoring and coordination of task execution according to contracts, performance assessment, inter-operation and information integration protocols and mechanisms, etc. Further problems include the lack of common ontologies among the cooperating organizations, derivation of the information visibility regulations based on contracts, the proper support for socio-organizational aspects e.g. lack of a culture of cooperation, the time required for trust building processes, the need for BP reengineering and training of people, etc.
- ***VO dissolution*** – This phase is practically not covered by research projects, so far.

Since CoBASA is inspired in collaborative networked organisations, particularly in Virtual Organisation, the problems highlighted above are also expected to be found when forming coalitions or consortia of manufacturing components. Undoubtedly CoBASA coalitions need to be planned and created, operated, and dissolved. The following research topics from the VO world have influenced CoBASA:

- Selection of partners with the right skills to answer the requirements needed by the VO.
- The coordination mechanisms used by VO members.
- Value system. This includes the "quality" of each company and the quantification of their performance within the virtual organisation.
- Contracts to regulate behaviour.
- Mechanisms to enable and regulate the dynamic interaction that is needed by VO participants. This includes the aspects that facilitate the interconnection between different entities in a fast way (methods and tools for interoperability).

3.6.1 Clusters

A relevant question for the VO creation phase is which strategies best improve and facilitate the creation of VOs. There are two important issues regarding this phase:

- Searching the companies or organisations that are willing to participate in a coalition. The open question is what are the strategies that improve this search.
- How companies overcome the trust problem. It is quite well known that consortia are easier to establish with partners that trust one another.

In addition, a common language, common practices and the sharing of some resources or infrastructures might also help in the effort of obtaining business opportunities, which is the ultimate reason to create a VO or Virtual Enterprise (VE).

The concept of cluster, which is borrowed from the area of economic development (Raines, 2001), is usually regarded as a suitable mechanism to help the creation of VOs/VEs. The concept *industry cluster* can be found in many research works related to the economical area (Bergman & Feser, 1999; Boari, 2001; Piore & Sabel, 1984). Despite these various definitions the one used by Camarinha-Matos will be used.

Definition 3.6 – Cluster (Camarinha-Matos, 2002a)
A cluster represents a group or pool of enterprises and related supporting institutions that have both the potential and the will to cooperate with each other through the establishment of a long-term cooperation agreement. This concept is also known as breeding environment for VOs.

This is an old concept that has existed for a long time in Europe or USA. In (Piore & Sabel, 1984) it is discussed why some industrial clusters, called *industrial districts*, in Italy and in France were a source of survival for the companies that formed them, enabling them to keep a high degree of innovation and ability to cope with a turbulent world. Several researchers have always pointed out innovation as one important key advantage that results from clusters. Probably the most well known is Michael Porter who claims that spatial proximity brings competitive advantage if the firm has to manage a complex set of interdependencies with clients, suppliers and other institutions (Porter, 1990, 1998). The other advantage also mentioned by researchers is that trust can be improved among the companies participating in clusters.

The advances in ICT bring new opportunities to this old concept by leveraging its potential, namely by providing adequate support for the rapid formation of VOs/VEs. Furthermore, with ICT support, it is now possible to avoid the strict dependence of industrial clusters to geographical proximity. It is now possible to create clusters that are composed of geographically distributed companies.

The glue that binds the cluster together is economic self interest. Companies participate in a cluster to keep competitiveness. Buyer-supplier relationships, common technologies, common markets or distribution channels, common resources, or even common labour pools are elements that typically bind the cluster together. With a cluster, which may be regarded as a "breeding" or "nesting" environment, it is now possible to have common infrastructures and agreed upon business practices. In

addition, a sense of community and some sense of stability are possible to create. As Camarinha-Matos pointed out in (Camarinha-Matos, 2002a):

> *"A cluster represents a long-term organization and, therefore, presents an adequate environment for the establishment of cooperation agreements, common infrastructures, common ontologies, and mutual trust, which are the facilitating elements when building a new VE."*

Figure 3-20 shows how a cluster or breeding environement helps the creation of a VO/VE. It is quite clear that searching is easier in the cluster because, when the companies join the cluster they provide details about their resources and competencies. On the other hand the cluster acts as a place where companies start to increasingly acquire confidence in their mutual partners. The *obligation* to follow common practices, as well as general operating principles, is of great help to the increase of trust and confidence.

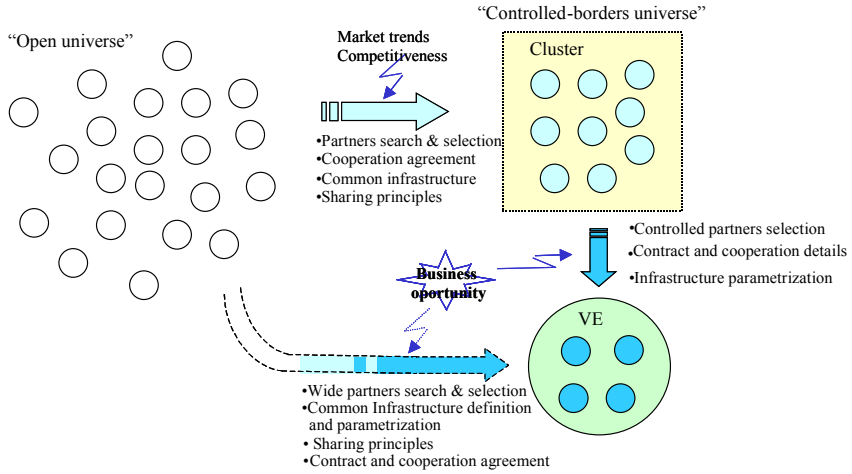

Figure 3-20 - Relationship between cluster and VE/VO – from (Camarinha-Matos, 2002a)

For each business opportunity found by one of the cluster members, a subset of the cluster enterprises may be chosen to form a VE for that specific business opportunity. In this perspective, the expected competitive advantage of cooperative development of products and services becomes a more relevant tie among the cluster members. The more frequent situation is the case in which the cluster is formed by organizations located in a common region, although geography is not a major facet when cooperation is supported by computer networks. Nevertheless, the geographical closeness has some advantages for cooperation as it may facilitate better adaptation to the local (culture) needs and an easier creation of a *sense of community*.

The cluster enterprises are normally *registered* in a directory, in which their core competencies are *declared*. Based on this information, the VO/VE initiator / creator (**broker**), which is usually a member of the cluster enterprises, can select partners when a new business opportunity is detected. Clearly, several VOs/VEs can coexist at the same time within a cluster, even with some members in common. The cluster does not need to be a closed organization; new members can adhere but they

have to comply with the general operating principles of the cluster. Similarly, for the formation of a VO/VE, preference will be given to the cluster members but it might be necessary to find an external partner in case some skills or capacities are not available in the cluster. The external partner will naturally have to adhere to the common infrastructure and cooperation principles. In addition to enterprises, a cluster might include other organizations (such as research organizations, sector associations, etc.) and even free-lancer workers. The establishment and management of clusters through adequate infrastructures represent, therefore, an important support for the creation of agile virtual enterprises.

The cluster concept is very important under the CoBASA framework because agentified manufacturing components, before participating in any manufacturing coalition, need to be registered in the cluster that groups all the manufacturing components that physically belong to a certain manufacturing structure, such as an assembly cell or line.

3.6.2 Cooperation Forms

To better understand the aspects related to networked cooperation among autonomous entities several examples that characterise different ways of organising cooperation among enterprises are introduced.

An ***Agreement*** is an arrangement between parties regarding a method of action. The goal of this arrangement is to regulate the cooperation actions among partners, and it is always associated with a contract. Examples of agreement clauses are 1) ways of communication, 2) reporting procedures, 3) representations, etc. A ***Partnership Agreement*** establishes the provisions that regulate the partnership co-operation. Examples of such provisions are the manner in which profits are to be distributed, or the supporting infrastructures. Figure 3-21 to Figure 3-24 illustrate typical cooperation arrangements (Camarinha-Matos & Barata, 2001).

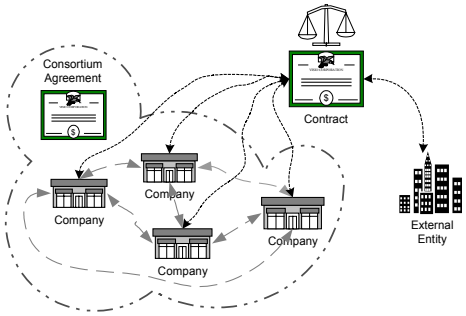

Figure 3-21 - Open Consortium

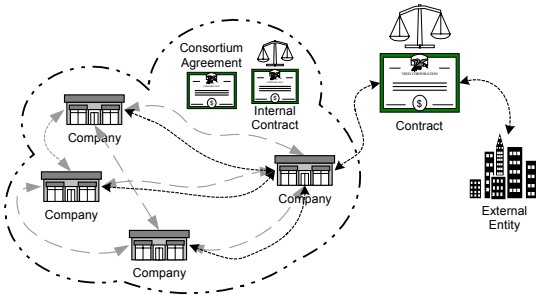

Figure 3-22 - Internal Consortium

The main characteristics of the Open Consortium are:

- There is a contract and a consortium agreement
- All partners become committed to the External Entity because they all sign the contract

- The agreement can be established either before the contract or at contract time
- The External Entity cares about who is part of the consortium
- There are cooperation relationships among partners
- Apart from the commitments represented by the contract and agreement, partners are autonomous
- The consortium is dissolved at the end of the contract.

The main characteristics of the Internal Consortium are:

- There is a contract between one representative of the consortium and the External Entity
- The External Entity does not necessarily know about the way the consortium is organised
- The consortium is also formalised using an agreement and an internal contract
- Only one partner (the one that signs the contract) is committed to the External Entity. The other partners are committed to the one that signs the contract
- There are cooperation relationships among partners
- Apart from the commitments represented by the contract and agreement, partners are autonomous
- The consortium is dissolved at the end of the contract.

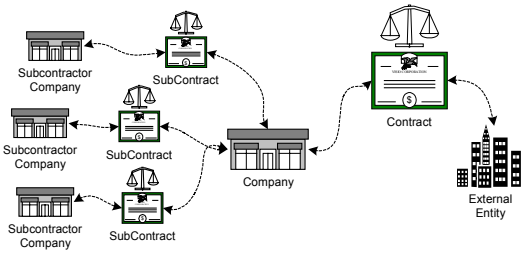

Figure 3-23 - Sub-Contracting

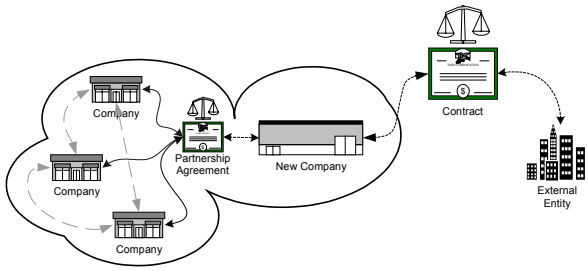

Figure 3-24 - Partnership

The main characteristics of Sub-Contracting are:

- There exist a contract and subcontracts
- The customer does not necessarily know about the way the contracted partner is organised
- Only the contracted company is committed to the External Entity
- The contracted company creates all the subcontracts that are needed to perform its contract
- There is no need for cooperation among the subcontractor partners
- Apart from the commitments represented by the contract/subcontract, partners are autonomous
- A long-term cooperation relationship may exist.

The main characteristics of Partnership are:

- There is a contract and a partnership agreement

- The partnership creates an entity (new company) using a partnership agreement
- The External Entity (customer) deals with one entity (new company)
- Only the new company is committed to the External Entity.
- There are co-operation relationships among partners
- There is a tightly-coupled relationship among partners
- The partnership may continue after the end of the initial client contract.

3.7 INTEGRATION INFRASTRUCTURES

Integration is addressed because of the following main reasons:

- Highlighting that more than ever, agile manufacturing systems are only possible with the effective integration of enterprises' business processes, and computer hardware and software.
- An effective integration approach can only be possible if the already existing applications and computer systems (legacy systems) are integrated in the new approaches. In the particular case of manufacturing systems it is of fundamental importance that existing manufacturing components such as robots are reused and integrated in the new advanced control strategies.
- It is not possible to develop CoBASA without considering integration aspects, such as FIPA standards for agent interaction or standards like CORBA (Bellavista & Magedanz, 2001; CORBA, 2001; Siegel, 2000) or DCOM (DCOM, 2001) for middleware interoperability.
- To show that the work developed under this thesis does not intend to be another island of automation within a manufacturing company. On the contrary, CoBASA was always thought of as a tool that can be easily integrated in the global operation of any manufacturing company that uses it.

The introduction of computer based information systems within companies was not made in an organised way. Each company department used their own computer-based systems to support the specific activities of that area. Consequently it was not possible to have a coherent view of the company global information system. This might be acceptable while the needs for information support are not very demanding. However, this changed radically when computer based systems invaded the various departments and a global view of the information became necessary. Managers started to require updated information of the production systems and shop floor information was needed at the higher management levels or even externally to the company.

Integration must occur at different levels and with different views. For instance, business process reengineering (Davenport, 1993; Hammer & Champy, 1995; Hammer & Stanton, 1995; Zhang & Cao, 2002) has been acclaimed as an important methodology that addresses the issue of structural integration by reorganising companies along critical business processes (see chapter 2). In addition,

the issues of operation and information integration must also be addressed. (Hansen, 1991) defined the following principles related to behavioural and information integration:

1. *When people understand the vision, or larger task, of an enterprise and are given the right information, the resources, and the responsibility, they will 'do the right thing'.*
2. *Empowered people - and with good leadership, empowered groups will have not only the ability but also the desire to participate in the decision process.*
3. *The existence of a comprehensive and effective communications network (...) This network must distribute knowledge and information widely, embracing the openness and trust that allow the individual to feel empowered to affect the 'real' problems.*
4. *The democratisation and dissemination of information throughout the network in all directions irrespective of organisational position.*
5. *Information freely shared with empowered people who are motivated to make decisions will naturally distribute the decision making process throughout the entire organisation.*

Integration is much more than a problem of interconnecting physical components and software applications since global business integration is needed, aiming at the use of the existing or new enterprise resources in order to better achieve the overall business objectives. Tools and methods that provide an integrated view of the various information systems, and which allow the company, from the point of view of IT, to be seen as a whole, is the objective of any enterprise integration infrastructure. It requires the development of an information infrastructure that supports the communication of information and knowledge, the making of decisions, and the coordination of actions. Integration, although an old problem, is still far from being solved.

Management of change is another theme related to integration. Any successful integration needs to accommodate evolution, i.e., the business processes that describe the enterprise must be easily changeable without much influence on the way the information is organised and exchanged. Instead of considering monolithic business processes it is better to consider a set of cooperating processes in order to be able to implement changes, operational modifications, and extensions in 'real time'. ICT is the technology that permits the integration of business processes with these characteristics. In a world of agile manufacturing, evolution becomes the norm rather than the exception. CoBASA is an architecture that supports the management of change on a shop floor.

The following aspects are relevant to the discussion when addressing the issue of integration:

- ***Enterprise modeling*** – At the heart of any integration infrastructure lies the enterprise model.
- ***Ontologies*** – A common understanding of the concepts is the only way to integrate the various areas, which cover different domains.

- ***Standards*** – Different types of standards exist, from standards for common interconnection of machines within the shop floor to standards on how to integrate product related information.
- ***Information integration*** – Within companies, different kinds of information and knowledge are produced. Most of the time, the places where the information is needed is not where it was produced. Consequently it is necessary to make the information available throughout the places where it is needed, which is only possible with information integration.
- ***Application integration*** – This is related to interoperability that may be defined as the ability to operate multiple devices, regardless of the manufacturer, without loss of functionality.
- ***Legacy systems*** – The work on integrating existing applications or systems.

3.7.1 Enterprise Modeling

Any integration accommodating the different aspects of an enterprise is not possible without abstraction. The complexity is so vast that only with abstract models that capture the important facets and views is it possible to think about integration. The development of enterprise models is thus a way to achieve integration. An interesting aspect that is noticed when modelling an enterprise is that many of these models (structural and operational) are in fact independent of the type of company. This means that they can be used in a large variety of companies, and more than that, they can be standardised. This fact is the foundation behind the idea of Enterprise Reference Architecture.

Definition 3.7 – Enterprise reference architecture (narrow sense) (Williams et al., 1994)
Describes the integrated enterprise at a generic level. These architectures describe the integrated enterprise as it is going to function from various points of view.

Definition 3.8 – Enterprise reference architecture (broad sense) (Williams et al., 1994)
Describes the enterprise in various stages of its development, each stage being possibly described from many points of view.

The most relevant Reference Architectures are: CIMOSA, PERA, GRAI-GIM, and GERAM.

Open System Architecture for Computer Integrated Manufacturing – CIMOSA (ESPRIT Consortium AMICE, 1993; Jorysz & Vernadat, 1990a, 1990b, 1990c; Vernadat, 1996; Williams et al., 1994; Williams, Bernus, & Nemes, 1996)
The CIMOSA objective is the use of enterprise models for the monitoring and control of daily enterprise operations. To achieve this objective CIMOSA developed two major parts: An executable enterprise model and an integrating infrastructure. It attempted to support all phases of CIM life cycles, from requirements specification through system design, implementation, operation, and

maintenance. It defines four different views: 1) function, 2) information, 3) resources, and 4) organisation.

Some relevant remarks about CIMOSA:

- It is the Reference Architecture most formally described and strongly committed towards complete computer executability.
- The initial phase of *User Requirements* is not covered by CIMOSA, which assumes these requirements as an input.
- The operation phase considers only system changes that occur because of software changes. This means that CIMOSA is not a good reference model to cope with the problem of change or continuous evolution of systems (including hardware).
- Easily understood by people with information technology background.

The Purdue Enterprise Reference Architecture – PERA (Williams, 1994; Williams et al., 1994; Williams et al., 1996)

The PERA Reference Architecture and methodology provides guidelines for the integration of applications (tools) through all the phases of an integration program from initial concept through use to final decommissioning. It considers the full life cycle of the enterprises. This architecture clearly separates the functions related to information, which includes decision, control, and information, manufacturing, and human and organisational factors.

Some relevant remarks about PERA:

- Like the GRAI-GIM architecture, described below, PERA includes a specific life cycle diagram.
- It is informal when compared to GRAI-GIM and CIMOSA, which makes it the most accessible one to be understood by non-computer science users. Easily understood by industrial engineers.
- The Production Equipment (or customer service) view permits support for integrating machines and energy flow, equipment layout and other relevant aspects of production, moving the focus of integration away from information systems alone.
- Another important characteristic of the PERA is the explicit representation of humans and their related activities in the architecture, however it is poorly formalised.
- It is the most extensively documented methodology.
- Because of being informally defined it lacks mathematical models necessary when the implementations are computer-based.

GRAI-GIM (Doumeingts, Vallespir, & Chen, 1995; Fox & Gruninger, 1998; Williams et al., 1996)

The life cycle of the GRAI-GIM methodology has five phases: 1) Analysis, 2) Design, 3) Technical Design, 4) Development, and 5) Operation.

Some remarks about GRAI-GIM:

- Like the Purdue architecture, GRAI-GIM includes specific life cycle diagrams.
- GRAI-GIM is well suited to guide the user through the specification stages.
- It is very much oriented to information integration. Most of the implementation cases that used this architecture were in the development of computer systems and related software and hardware to implement the factory integration.
- The human worker is considered as another resource that should have some physical capability and skill. The methodology does not discuss human related aspects.
- According to (Williams et al., 1994) GRAI-GIM is positioned between CIMOSA and PERA in terms of formality. This implies that GRAI-GIM is better understood by non computer science educated users than CIMOSA.

Generic Enterprise Reference Architecture – GERAM (IFIP-IFAC, 1999; Leger & Morel, 2001; Santos, Ferreira, & Mendonca, 2000; Szegheo & Petersen, 2000)

The Reference Models GRAI-GIM, PERA, and CIMOSA were analysed by the IFIP-IFAC task force and it was concluded that besides some overlap none of the models subsumed the others and each of them has something unique to offer. Therefore GERAM was established as a generalisation of the existing architectures and other necessary elements. It also facilitates the unification of methods of several disciplines, such as methods of industrial engineering, management science, control engineering, communication and information technology and others.

As defined in (IFIP-IFAC, 1999) "*GERAM defines a tool-kit of concepts for designing and maintaining enterprises for their entire life-history. GERAM is not yet-another proposal for an enterprise reference architecture, but is meant* to organise existing enterprise knowledge. *The framework has the potential for application to all types of enterprises. Previously published reference architectures can keep their own identity, while identifying through GERAM their overlaps and complementing benefits compared to others*."

Relevant aspects of GERAM are:

- It unifies two distinct approaches of enterprise integration: 1) based on product models, and 2) based on business process design.
- Reengineering activities are supported by the architecture, which means that management of evolution is possible.
- It combines the modelling capabilities of CIMOSA and the PERA associated tools.

Other important initiatives related to Enterprise modelling include:

- ***Integrated Computer Aided manufacturing - ICAM*** – An effort led by the United States Air Force in the early eighties.
- ***Workflow Management Coalition*** – WfMC (Fischer, 2003; WFMC, 2002) – This organisation supports the development of a workflow reference model to guarantee that

different workflow engines can communicate among each other. Workflows are suitable for integration because they automatise the execution of tasks associated with business processes.

- ***The Process Specification Language*** (PSL, 2002) defines a neutral representation for manufacturing processes to help the integration of multiple process-related applications throughout the manufacturing life cycle. This is a potential candidate to establish a standard and platform independent business process (BP) modelling language to allow BP exchange that is especially relevant to VOs/VEs.
- ***Unified Enterprise Modelling Language- UEML*** (UEML, 2003) – UEML is common modelling language that facilitates inter-operability within the frame of ongoing standardisation efforts.

3.7.2 Ontologies

Ontologies facilitate the development of integration in a two-fold approach. First, the development of common concepts facilitates the exchange of information and knowledge. On the other hand, the thorough analysis that is required to develop an ontology is important for the integration because it permits the development of better models that help integration.

3.7.3 Standards

Standardisation is probably the immediate answer to any problem that involves interoperability between two systems. It must be remembered that interoperability is one of the most important facets of integration, because it is not possible to do any integration without it being possible to connect[28] two or more different systems.

If a standard exists that is followed by a large number of manufactures, then integrating applications and systems becomes easier. However, this acceptance does not always occur and, in many cases, standards are not followed. On the other hand, some initiatives that were not thought to be a standard because they were developed individually became, later on, de-facto standards. Probably the most well known example is TCP-IP, the communication protocol that is the basis of the Internet.

The biggest drawback associated to standards is the time usually needed to achieve a consensus. In addition, often this consensus does not reflect technological advances but rather political consensus. This is why many standards are far from being state of the art solutions. Open software approaches used by the Open Software Foundation in its open standards work reduces the dependence of standards on political decisions in which big partners have more influence and speeds up the process.

The proliferation and availability of lower cost commercial off-the shelf information technology is pushing for new ways of integrating automation systems on the shop floor. Increasingly Ethernet

[28] Please note that connection here is used in the broad sense of the word. It covers physical connection, and exchanging of information between two or more different applications.

TCP/IP protocols are being used on the shop floor. A lot of effort has been dedicated to the process of standardising communication between systems:

- ***AS-Interface*** (AS-I, 2002) – A standard for interconnecting shop floor equipment. A low-cost electromechanical connection system designed to operate over a two-wire cable carrying data and power over a distance of up to 100m.
- ***FieldBus*** (FieldBus, 2002) – Fieldbus describes a digital communications network. The network is a digital, bi-directional, multidrop, serial-bus, communications network used to link isolated field devices, such as controllers, transducers, actuators and sensors. Each field device has low cost computing power installed in it, making each device a 'smart' device. Each device will be able to execute simple functions on its own such as diagnostic, control, and maintenance functions as well as providing bi-directional communication capabilities.
- ***Process Field Bus – PROFIBUS*** (PROFIBUS, 2002) **–** This organisation supports one of the derivations of fieldbus. It is considered a de-facto standard for industrial networks. Depending on the application, the transmission technologies (Physical Profiles) RS-485, IEC 1158-2 or fiber optics are available.
- ***Controller Area Network – CAN*** (CAN, 2002; Lawrenz, 1997) – This ISO standard (ISO 11898) was developed in Europe for use in passenger cars. It standardises real time communication among different devices over a two-wire serial data bus.
- ***DeviceNet*** (*DeviceNet*, 2002) - DeviceNet is an Open Network Standard based on the CAN standard.
- ***ControlNet*** (*ControlNet*, 2002) – ControlNet is a 5 Mbit/sec serial communication system for communication between devices that wish to exchange time-critical application information in a deterministic and predictable manner.
- ***ETHERNET/IP*** - It is an industrial networking standard that takes advantage of commercial off-the-shelf Ethernet communication chips and physical media. IP stands for 'industrial protocol' and is what distinguishes this network. It uses an open protocol at the application layer. It is supported by three networking organizations: ControlNet International (CI), the Industrial Ethernet Association (IEA) and the Open DeviceNet Vendor Association (ODVA).
- ***Bluetooth*** (Bluetooth, 2003) **–** This is a wireless standard to enable links between mobile computers, mobile phones, other portable handheld devices, and connectivity to Internet.
- ***UPnP –*** *Universal Plug and Play (UPnP, 2003)* – This is a standard for pervasive peer-to-peer network connectivity of PCs and intelligent devices or appliances, particularly within the home. UPnP builds on Internet standards and technologies, such as TCP/IP, HTTP, and XML, to enable these devices to automatically connect with one another and work together.

Other standardisation efforts relevant for the shop floor include the Manufacturing Execution Systems (MES) (MESA, 2002), and the OPC foundation work (*OPC*, 2002) that is dedicated to ensuring

interoperability in automation by creating and maintaining open specifications that standardize the communication of acquired process data, alarm and event records, historical data, and batch data to multi-vendor enterprise systems and between production devices. The standardisation work on EXPRESS/STEP, already identified in chapter 2, is also relevant in manufacturing. Finally, standardisation efforts in the multiagent area led by FIPA are becoming more important due to the increasing use of agents in the manufacturing domain.

3.7.4 Information Integration

Information integration is one of the classical tasks that are required in any systems integration. Information might be produced and consumed in different places and using different abstract models and, consequently, some kind of common abstraction is needed in order that information can be conveniently used. Eliminating redundancies and inconsistencies as well as guaranteeing that all the produced information can be accessed, in case the user has convenient rights, are some of the steps needed for information integration. Three approaches are usually used for information integration: centralised, distributed homogeneous, and federated.

Centralised approach. The integration is achieved by having only one central repository; both producers and consumers need to rely on the same data model. Therefore, there is no need to deal with heterogeneous information, which makes, for instance, centralised databases answer queries more efficiently. However, the approach has some important disadvantages. First, it obliges any consumer or producer of information to follow the data model of the central repository, which makes the integration of legacy systems, for instance, quite difficult. Second, a centralised database might be less fault-tolerant. Finally, the centralised approach requires more powerful equipment to drive the generally huge database.

The inherent decentralisation of business and manufacturing operations also contributes to this approach becoming less appealing. Although it is possible to keep distributed access to a centralised data store, this approach has some disadvantages such as: 1) performance degradation due to a growing number of remote locations over greater distance, 2) high costs associated with maintaining and operating large central database systems, 3) the already mentioned reliability problems created by dependence on a central site, and 4) this approach gives the people that work with the database the feeling that they are the owners of the information, which is a serious drawback when agility is required.

Distributed homogeneous databases. They are multiple sources, multiple location database environments that can be defined as a collection of multiple logically interrelated databases distributed over a computer network. This dispersed information (databases) over several computers is regarded as a single logical database. Distributed homogeneous databases use the same type of management

system (DBMS) and the same kind of application at each node. In addition, they have the same schemas and in general the nodes are not too autonomous in terms of rights to change schemas and software.

In this approach it is possible to gain the advantage of having several databases, which are more fault tolerant, and require less computer power individually, while keeping the advantage of having a global view. Furthermore, because it is composed of several databases, it is possible to better tune each node, and its capacity can also be extended without spending huge amounts of money in expensive database servers.

Federated databases. As the name suggests a federated database system is a collection of cooperating database systems that are autonomous and possibly heterogeneous. In comparison to the distributed database the important thing to retain is the autonomy of the database components that compose the federation, each one with their own database management system. For instance, a federated database might be composed of centralised database components coexisting with distributed databases components. The component databases simultaneously have control over the data they manage (autonomy) and allow partial and controlled sharing of their data, which makes possible the component databases to continue its local operations while participating in a federation. Therefore the administrator of each component database controls the level of integration.

The level of local information available to the federation depends on the autonomy of the local node (one of the databases belonging to the federation). A federated database represents a compromise between a situation in which a user explicitly accesses individual databases (no integration) and a situation in which the autonomy of each component is sacrificed so that users can access data through a single global interface but cannot directly access a local database as a local user (total integration). Consequently federated databases are a well indicated architecture to integrate legacy databases because standalone databases can be migrated to a system that allows partial and controlled sharing of data without affecting existing local applications.

One important characteristic in any database is the schemas as they describe how data is represented. In (Sheth & Larson, 1990) five types of schemas are considered as an adequate federated database architecture: local schema, which is the conceptual schema of the component database, component schema, which is a schema created to translate local schemas into another data model (common model), export schema, which represents a subset of a component schema made available to the federation, federated schema, which is an integration of multiple export schemas, and external schema, which defines a schema for a user or application. Although all schemas are relevant the export schema is particularly important because it include access control information regarding its use by specific federation users. The information dealt within federated systems consists of different pieces of information gathered from disparate local and remote sources and integrated into one coherent view (federated schema).

Federated databases have been used in the manufacturing domain mainly due to adequateness to integrate heterogeneous legacy databases. It is particularly relevant the work led by Prof. Afsarmanesh which have been applied to the virtual enterprise domain (Garita, Afsarmanesh, & Hertzberger, 2001, 2002; Rabelo et al., 2000).

3.7.5 Application Integration

In this subsection the goal is to describe technologies used to support the integration of applications. Much of the contents are based on (Barata, Camarinha-Matos, Boissier et al., 2001).

Integration in the IP world. The Internet protocol (IP) routes packets of data up to 64 Kbytes large, and supports essentially two transport level protocols:

- TCP (Transport Control Protocol), which is a reliable stream mode connection based protocol (partners negotiate to open a communication session on a particular *port number*), near to the OSI TP4 protocol, well applicable for applications such as file transfer (FTP) or remote connection (Telnet). No limit is set to the size of transmitted data. Fragmentation and packet numbering take place for sizes larger than 64 Kb.
- UDP (User Datagram Protocol), which is a connectionless protocol with no flow control or error recovery, applicable for local and small volume interactions such as data value reports. It is used in the trivial file transfer protocol (TFTP).

Sockets are a de facto standard interface to the TCP and UDP protocols. Opening a socket returns a handle. Because of their very low level, programmers have to care about platform specific details (byte ordering, etc.) such that sockets are limited to system's programming. A possible improvement is to standardise the description and encoding of transmitted data. This is the case with the *Abstract Syntax Number One* (ASN.1) and the *Basic Encoding Scheme* (BER) used in the ISO protocols or in the TCP/IP Simple Network Management Protocol.

Remote Procedure Calls (RPC), originally developed by Sun, and available on Unix and Unix-like platforms, have been extended and standardised by the Open Group (formerly OSF-X/Open) within the *Distributed Computing Environmentc(DCE)* frame. A slightly altered version of DCE/RPC has been integrated to the Microsoft Windows family platforms.

RPC is a high level mechanism that allows a client program to invoke a procedure on a remote server in a manner as transparent as possible to the programmer. To achieve such a goal, the interface (calling and return type) of the called procedures must be described in an *Interface Definition Language* (not CORBA's IDL) and then compiled to generate the so-called local *stubs*. At runtime the calling program actually invokes a RPC package generated image of the distant function in the local stub, the stub packs this call into network messages to the distant stub, which in turns invokes the real remote procedure. The remote procedure executes and returns its possible value to the calling

program along the same path. Port mapping, conversion of arguments into messages, etc. are managed by the RPC runtime infrastructure.

Integration using middleware. A common approach to integrating applications is by using middleware, which, in its essence, is just software that connects applications by transferring data between them or by managing how an application handles requests from the others (such as requests for information or for performing specific tasks). It is a higher abstraction mechanism than RPCs.

The idea common to Object Request Brokers (ORB), also designated as software buses or middleware, is to allow a local object to invoke methods on distant (remote) ones. In order to hide remoteness and networking details, the middleware runtime installs end points on the client side and on the server side. The client object submits an invocation to the local image of the distant server object that is then forwarded to the distant host where it is converted into an actual method invocation on the server object; the results are returned to the client object in the same indirect way. The behaviour is very similar to the RPC mechanism. However, a RPC remote procedure is, in this case, offered by a dedicated server, while ORB methods are attached to an object (note that a server can handle *many objects*).

The Common Object Request Broker Architecture (CORBA), is a distributed programming architecture specification corresponding to the post client-server paradigm, set up by the *Object Management Group* (www.omg.org) (OMG 2000). CORBA allows objects interaction independently of the source language. Besides the Object Request Broker (ORB) specification, the CORBA architecture defines general low level object services such as Naming, Object life cycle, Event notification, Security, Persistence, etc. CORBA ORBs from various origins are interoperable through the Internet Inter-ORB Protocol.

DCOM, Microsoft's Distributed Component Object Model, is the result of a progressive strategy to permit data interchange within the MS WINDOWS environment. It began with the simple clipboard *cut and paste* concept, continued with *Dynamic Data Exchange (DDE)* that was improved with the OLE-COM. At this stage the so-called *component* objects could register and later be activated from within other applications. A COM client object can activate methods attached to remote COM objects. Microsoft promotes the extension of the DCOM scheme to Unix-like platforms, and CORBA-to-DCOM gateways do exist.

A mechanism similar to DCOM is the Java Remote Method Invocation (JRMI), which operates in the same way. JRMI permits the invocation of a method from a distributed object and in this way allows the exchange of information between separated entities (in this case objects) in Java.

The most important disadvantage of JRMI as well as DCOM when compared to CORBA is their tightness to specific environments such as the Java language and the Microsoft world respectively.

JavaSpaces (Bishop & Warren, 2003; Freeman et al., 1999; Halter, 2002) supports the exchange of information among distributed applications via the flow of objects into and out of one or more entities called *spaces*. *Spaces* are network-accessible repositories for objects. Processes coordinate with each

other by using *spaces* as persistent object storage and exchange mechanisms, instead of communicating directly. Processes can *write* new objects into a *space*, *take* (i.e. remove) objects from a *space*, *read* (i.e. copy) objects from a space, or *wait* for objects not yet in the space. Objects in the space are passive and can be found by value-matching lookup. To modify an object, it has to be taken, updated, and reinserted. This model is based on the Linda Coordination language (Carriero & Gelernter, 1989; Gelernter & Carriero, 1992). It is suitable for the development of loosely coupled distributed applications in which the server and the client are not coupled in any way.

Integration using MMS. The Manufacturing Message Specification (MMS) (ISO 1990) (ESPRIT Consortium CCE-CNMA, 1995) brought together many IT and Manufacturing specialists to define a common framework for developing communication support between industrial computerised equipment. It is a standardized messaging system for exchanging real-time data and supervisory control information between networked devices and/or computer applications in a manner that is independent of: 1) the application function being performed, or 2) the developer of the device or application. The key concept of MMS is the *Virtual Manufacturing Device* associated with every real device (machine or cell). A VMD offers all services concerning itself and its related abstractions, mainly: *Domains* (which represent resources, possibly downloadable), *Named variables* (which can be of domain or VMD scope), *Program invocations* (corresponding to the execution of a machine task), and *Events* (with various associated mechanisms). Adaptations of this very general and abstract model to particular classes of devices (machine-tools, robots, logic controllers, process control) are proposed in *Companion Standards*.

This standard is one example of a standard that was never very popular. Although its concepts were quite advanced and interesting at the time it first appeared, the costs charged for the equipment needed to implement the standard were so prohibitive that few people used it. Nowadays, it is practically non-existent, mainly because it is possible to interconnect industrial equipment using low cost off the shelf Ethernet based equipment.

Integration using web technologies. Web technologies are increasingly used for integration purposes. An approach that has been used to facilitate the integration of applications is by developing a common hypermedia interface that can be accessed by normal browsers. The only requisite for this achievement is that the web pages must be dynamic, which can be achieved if the applications support one of the following web technologies: CGI – Common Gateway Interface, ASPs – Active Server Pages, or JSPs – Java Server Pages.

Web services are becoming an important paradigm to facilitate interoperability. The W3C (W3C, 2002) uses the following definition:

> *"A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format*

(specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards."

A Web service is viewed as an abstract notion that must be implemented by a concrete piece of software that sends and receives messages. The purpose of a Web service is to provide some functionality on behalf of its owner, a person or organisation (the *service supplier*) that provides an appropriate entity to implement a particular service. A *client* is an entity that wishes to make use of the service supplier's Web service by exchanging messages with the *service supplier*. In order for this message exchange to be successful, the client and the supplier entities must first agree on both the semantics and mechanics of the message exchange, which is made using WSDL (*WSDL - Web Services Description Language*, 2002). This is a service description language that describes the Web Service being offered in greater detail using XML (XML, 2000). The offered service is published using UDDI – Universal Description, Discovery, and Integration (UDDI, 2003) that acts like a yellow pages service. Clients eventually use these yellow pages to find services and get the WSDL interface they need to access the service, which they invoke using the SOAP protocol (SOAP, 2003). SOAP is an XML/HTTP-based protocol for accessing services, objects and servers in a platform-independent manner.

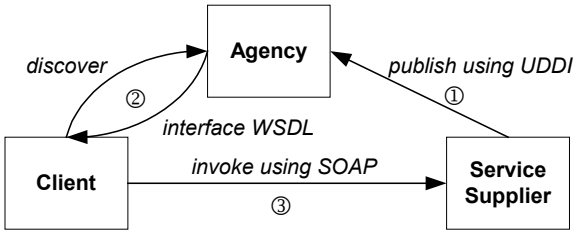

Figure 3-25 – Web services paradigm

Integration using agents. The agent approach can facilitate integration of applications primarily due to the inherent autonomy of agents and their capability to negotiate (Genesereth & Ketchpel, 1994). Autonomous units, if properly habilitated with social skills that allow interaction, are self-integrated. Since agents have already been discussed they will not be addressed again.

3.7.6 Legacy Systems

Integration of legacy systems is still one of the most important tasks in integration. Legacy refers to existing pieces of software or hardware usually developed in an older technology that needs to interact with new software or hardware. It does not only involve applications but can also be, as frequently happens in a manufacturing environment, manufacturing components such as robots or PLCs.

The most relevant approaches to deal with legacy systems in the context of manufacturing are:

- Creating a software wrapper or transducer enclosing the legacy system or application.
- Creating software components that act like an integration layer for the legacy system.
- Using an agent approach.

Software wrapper. A software wrapper or transducer is usually implemented as a server whose offered services mimic the functionalities of the legacy system (Figure 3-26). The server is implemented using traditional RPC mechanisms. The important aspect of this approach is developing the transducer layer that converts the requests for services of the legacy system using the language of the target system into the language of the legacy system. The big advantage of this approach is that the only knowledge that is required about the legacy system is its communication behaviour. It is, therefore, especially useful for situations in which the code for the program is unavailable or too delicate to modify. The integration can be made without taking into consideration the details of the implementation.

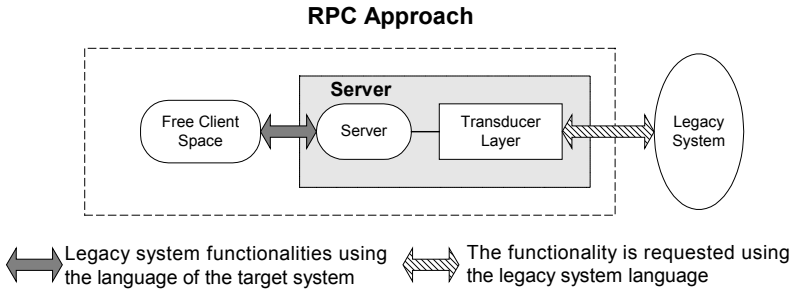

Figure 3-26 - A software wrapper

Software wrapping is a practice that is well suited to the integration of legacy systems for several reasons:

- Legacy systems are often closed systems not allowing access to source code or implementation documentation.
- Wrapping allows for information hiding.
- Wrappers provide access to a legacy system through abstract application program interfaces, regardless of the internal implementation complexity of the legacy system.
- Wrappers provide technology migration paths for legacy systems, allowing component upgrade without affecting the rest of the system.

Several works have been done integrating different heterogeneous robot controllers using this approach (Barata, 1995; Barata & Camarinha-Matos, 1995a, 1995b; Barata, Camarinha-Matos, & Chavarria, 1994; Barata, Camarinha-Matos, & Freitas, 1999; Barata, Vieira, & Camarinha-Matos, 1996; Camarinha-Matos, Barata, & Flores, 1997; Camarinha-Matos, Seabra Lopes, & Barata, 1994, 1996; Miyagi, Camarinha-Matos, Santos-Filho, Barata, & Arakaki, 1995, 1996). Nevertheless it shall

be noted that wrappers have to be developed in specific for each case, what in some cases requires considerable effort.

Component approach. Although this approach is similar to the previous one, the wrapper or transducer, rather than being implemented using RPCs, is implemented with distributed objects supported by a common middleware environment such as DCOM or CORBA (Figure 3-27). The transducer or wrapper is thus modelled as an object whose public methods represent the services (functionality) used to be offered by the legacy system, and whose internal methods make the connection to the legacy component. The reason why this approach is called component is because the objects are also considered in the middleware field as components that can be reused whenever they are needed.

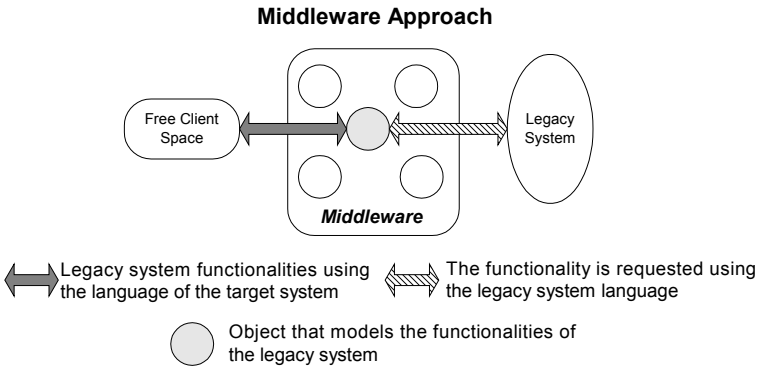

Figure 3-27 - Integration using wrapping with middleware

Several implementations in the shop floor domain were also done using this approach (Barata, Camarinha-Matos, & Colombo, 2001; Colombo, Camarinha-Matos, & Barata, 1999).

Agent Approach

Agents can be appropriate to integrate legacy systems using one of the following approaches (Figure 3-28):

- A transducer agent (middle "interpreter" agent).
- A wrapper (interface program with direct access to program's data structures).

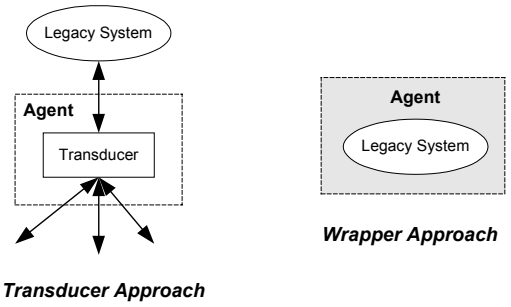

Figure 3-28 - Approaches to agentification

In the first approach the transducer is implemented using an agent that acts like a mediator between the other agents and the legacy system. This is the approach being followed in CoBASA, in which there is a mediator agent associated with each manufacturing component. In the second approach the legacy application, for instance, is altered by adding to its own code new functionalities to transform the application into an agent. This approach has greater efficiency over the transduction approach since there is less serial communication. However, full access to the code is required in this situation.

CoBASA uses two approaches: the agent (transducer) and the component. The transducer translates the requests from the CoBASA language into requests that could be understood by the robot controller. However, since legacy manufacturing controllers in general cannot be directly connected to a TCP/IP environment, it was necessary to create another abstract layer that converts TCP/IP requests (DCOM objects) into requests understood by the legacy controller. For instance, if a legacy robot controller is considered, the object component converts the TCP/IP requests into serial commands understood by the robot controller (Figure 3-29), while the transducer Agent Machine Interface converts CoBASA commands into TCP/IP requests to the component that represents the robot.

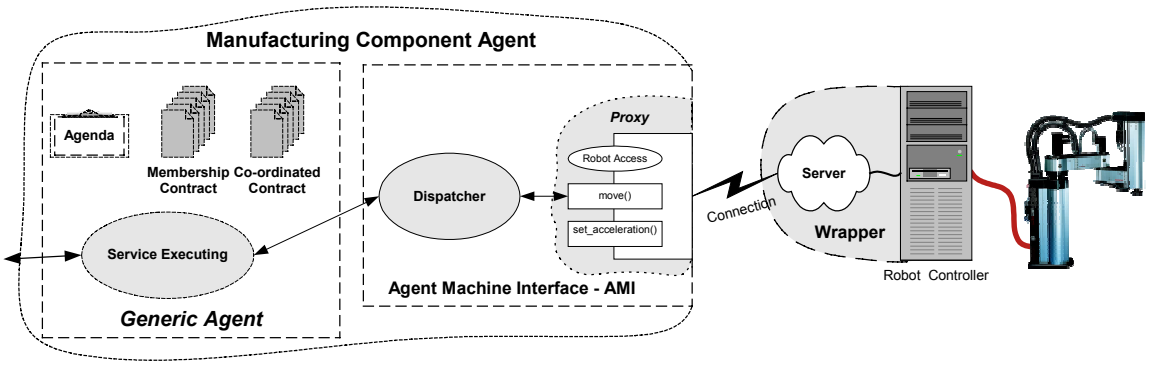

Figure 3-29 - CoBASA integration approach using agents

4 System Architecture to Support Shop Floor Reengineering

This chapter is mainly dedicated to the description of the proposed system architecture developed to support shop floor reengineering: the Coalition Based Approach for Shop floor Agility – CoBASA. The first section outlines the inspiration source and the philosophy that is behind the architecture. The second section provides details of the CoBASA architecture by identifying its main components and their interactions. The final section details the individual components that compose CoBASA, describing their individual functionalities and structure.

4.1 INTRODUCTION AND BASIC APPROACH

As mentioned in chapter 3, new business organisational forms like Virtual Enterprises (VEs) and other networks of enterprises, strongly demand agile shop floors. Creating a system to support a shop floor reengineering methodology is a hard and complex task for which no satisfactory solution has yet been found. This solution should support agility by facilitating the design of manufacturing systems out of heterogeneous manufacturing components using mainly configuration rather than programming. When dealing with such complex problems, a good approach might be to try to find a similar problem in another domain and then analyse it to see if the solution that was found for that problem can be applied to the initial problem. The challenge is to determine which similar problem to consider and how the solution found for that problem can be adapted to the new domain.

Human organisations are a good source of inspiration for complex problem solving because they are intrinsically complex and humans are used to creating highly dynamic complex structures to cope with complex problems. The approach followed in the design of a reengineering system architecture

assumes that there are similarities between the reengineering process and the formation of consortia regulated by contracts in networked organisations. Therefore, looking at the approaches being adopted in that domain can provide a source of inspiration for addressing the shop floor reengineering problem. The challenges a company faces to be agile are similar to the shop floor adaptation problem. Furthermore, the problems a company faces in order to join a consortium have some similarity to the adaptation of a manufacturing component (resource) on a shop floor. The proposed approach is therefore to use the mechanisms and principles developed to support enterprise's integration into dynamic enterprise networks as inspiration for an agile shop floor reengineering process. An interesting fact is that the world of networked organisations provides both the reason why an agile shop floor is needed and the inspiration source for the proposed solution.

Individual companies have a basic set of core competencies or skills. To be able to create/produce complex services or products, when working alone, companies must have a wide range of skills. It is assumed that a service/product is created/produced by the application of a set of skills. However, due to the increasing level of worldwide competition, companies need to focus only on those skills they are best at. The drawback of this decision lies on a lesser capability to create/produce complex services/products by themselves. The solution to survival is cooperating with other companies. Consequently, one or several cooperating partners are called upon to bring the missing skills and resources required to create/produce a complex service/product. At the same time, making cooperation work is not an easy task especially when played by partners that do not have previous knowledge of each other. Some kind of trust is almost mandatory for a successful cooperation.

Accordingly, cooperation can be promoted by a structure called **cluster** or a VE breading environment, already identified in chapter 3. This long-term aggregation of companies with similar interests or affinities, willing to cooperate, increases the trust level and can better accommodate business disturbances. The potential of skills resulting from the whole cluster is bigger than the sum of the skills that were brought in by each individual company because new skills can be composed of the basic ones. This is an interesting characteristic that renders clusters even more attractive, because the whole community being cooperative, enables much more potential to create/produce things. Although the cluster might have a potentially large set of skills, nothing is created/produced by the cluster, which simply possesses a potential for doing things. The cooperating structure that companies use to create/produce things is the **consortium**. A cooperative consortium or Virtual Enterprise is a group of companies that cooperate to reach a common objective. The formation of a consortium is generally triggered by a business opportunity. Different consortia can be formed with subsets of the cluster members. The capabilities of a consortium depend not on the global skills (potential) of each member but on the specific skills they agree to bring into the consortium. This means that the consortium global capabilities might be either larger (because of skill composition in which new skills can be formed from the basic ones) or smaller than the sum of the individual capabilities of its members.

Contracts are the mechanism that regulates the behavioural relationships among consortium members or between consortium members and the "external" client that generated the business

opportunity. The same entity constrained by different contracts can have different behaviours. If, for some reason, a company participating in a consortium reduces or increases its core competencies, this change might have an impact on higher-level consortia, which can see their capabilities (skills and capacities) maintained, reduced or increased. This situation obviously implies a renegotiation of the established contracts.

Similarly, in the manufacturing shop floor the manufacturing components, which are controlled by a diversity of controllers and correspond to companies in the Virtual Enterprise world, are the basic set from which everything is built up. A shop floor can be seen as a micro-society, made up of manufacturing components. The components have basic core capabilities or core competencies (skills) and, through cooperation, can build new capabilities. A robot, for instance, is capable of moving its tool centre point (TCP) and setting different values for speed and acceleration. Its core competencies are represented in Figure 4-1. A gripper tool, on the other hand, has as basic skills the capability to close (grasp) or open (ungrasp) its jaws. These two components when acting alone can only perform their core skills. However, when they cooperate, it is possible to have a pick-and-place operation that is a composition of the move with the open and close skills. The greater the diversity and complexity of individual capabilities, the greater are the chances of building more complex capabilities.

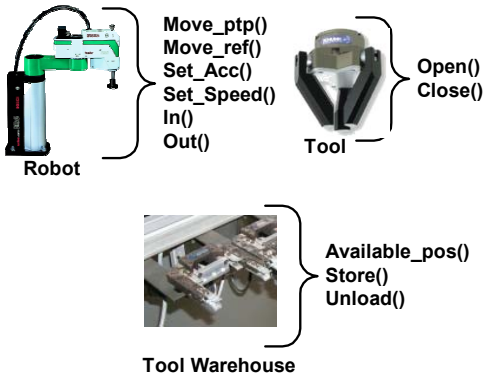

Figure 4-1 – Example of basic manufacturing components and core competencies

In the architecture being proposed every manufacturing component e.g. robots, tools, fixing devices, is associated with an agent that represents its behaviour (agentified manufacturing component). When these agents interact or cooperate they can generate aggregated functionalities that are compositions of their individual capabilities. This is what happens when, for instance, several manufacturing components are working together in a manufacturing cell.

Definition 4.1 - Manufacturing component

A manufacturing component is a physical piece of equipment that can perform a set of specific functions or basic production actions on the shop floor such as moving, transforming, fixing or grabbing.

Definition 4.2 – Agentified manufacturing component

An agentified manufacturing component is composed of a manufacturing component and the agent that represents it. The agent's skills are those offered by the manufacturing component, which is connected to the agent through middleware.

Definition 4.3 – Coalition/Consortium

A coalition/consortium is an aggregated group of agentified manufacturing components, whose cooperation is regulated by a coalition contract, interacting in order to generate aggregated functionalities that, in some cases, are more complex than the simple addition of their individual capabilities.

A coalition is usually regarded in the multiagent community as an organisational structure that gathers groups of agents cooperating to satisfy a common goal. On the other hand, the term consortium is more usual in the business area where it is defined as an association of companies for some definite purpose. The definitions are quite similar because in both situations there is the notion of a group of entities cooperating towards a common goal. This common definition is adapted to the context of the architecture being proposed here. From now on the terms consortium and coalition are used with the same meaning. Nevertheless, to emphasise that the architecture being introduced here is composed of manufacturing components and not of companies the term coalition will be favoured.

The coalition is the basic organisational form of cooperation in the architecture being proposed. A coalition is able to execute complex operations that are composed of simpler operations offered by coalition members. A new coalition can be established with either individual members or other existing coalitions (Figure 4-3). A robot cooperating with a gripper chosen from a tools' warehouse illustrates a simple example of a coalition. The better the way coalitions can be changed, the better the agility of the manufacturing systems they represent will be. If agility is seen as the capability to easily change the shop floor as a reaction to unforeseen changes in the environment, then an easy way to create and change coalitions is an important supporting feature for the manufacturing system's agility.

When forming a group of collaborative agents there are no limitations on the type of agents that can be involved in it but there is an important restriction which limits their cooperation capability – their spatial relationship. Manufacturing agents that are not spatially related cannot cooperate, as it is in the case of, for instance, a robot and a tool. If the tool is not within the reachability space of the robot it will be impossible to create a cooperative relationship. Another example of constraint is the technological capability. In order to be usable by the robot, the tool has to be technologically compatible with the robot wrist. Therefore, when creating a coalition it is mandatory to know what the available and "willing" to participate agents are that should present some compatibility among them (for instance spatial or technological compatibility). The manufacturing agents that can establish

coalitions should be grouped together because of these aspects of compatibility. This is analogous to the long-term collaborative alliances of enterprises. The objective of these clusters is to facilitate the creation of temporary consortia to respond to business opportunities. Similarly, in the case of the architecture being described there is a need for a structure (cluster) that groups the agentified manufacturing components willing/able to cooperate.

Definition 4.4 - Shop floor cluster

A shop floor cluster is a group of agentified manufacturing components which can participate in coalitions and share some relationships, like belonging to the same manufacturing structure and possessing some form of technological compatibility.

A community of agents belonging to the same physical structure – a manufacturing cell, thus forms a cluster, and when a business opportunity (i.e. a task to be executed by the shop-floor) arises, those agents with the required capabilities (skills and capacities) and compatibility are chosen to participate in a coalition. The limitation for an agentified manufacturing component to be accepted in a shop floor cluster is that it must be compatible with the others physically installed in the cell. For instance, an agentified robot installed far from a cell is not a good candidate to join the cluster that represents that cell, because it can never participate in any coalition. Since all the manufacturing components installed in a cell answer the requirements for compatibility a shop floor cluster is associated with a physical cell.

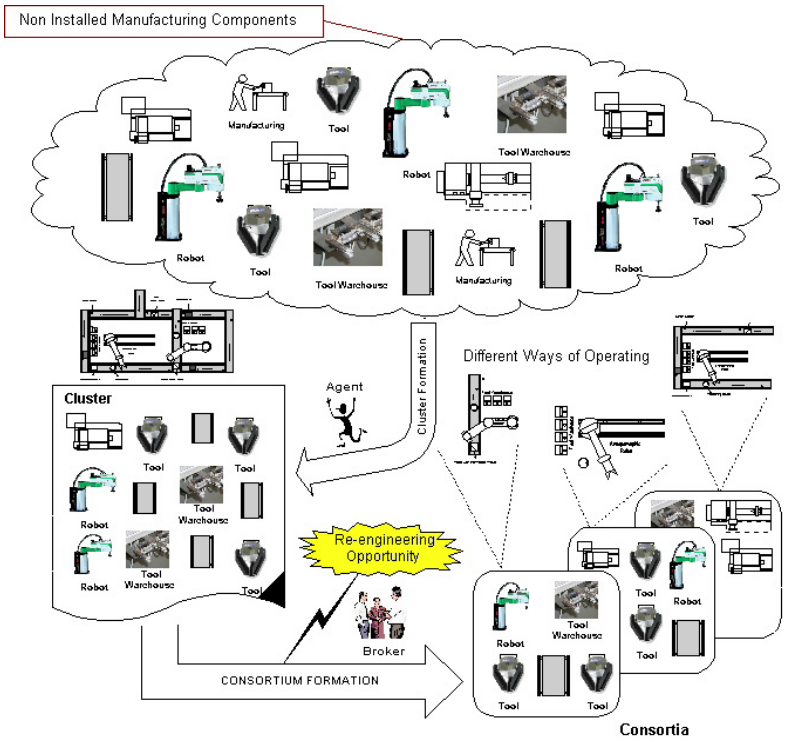

Figure 4-2 – Consortia formation

Figure 4-2 shows how manufacturing agents, cluster, and coalition interrelate. Agentified components in the same "geographical" area of the shop-floor join the same cluster. The different coalitions that can be created out of a cluster represent the different ways of exploiting/operating a manufacturing system. Adding or removing a component from the physical manufacturing system also implies that the corresponding agent must be removed from the cluster, which can also have an impact on the established coalitions. A broker is used to help the formation of coalitions to reduce the complexity of the individual agents in terms of coalition formation. By delegating this responsibility to the broker, the individual agents can be simpler because all they have to do is negotiate the terms of their participation with the broker rather than carrying out all complex details of coalition formation such as deciding which members are better indicated to answer the requirements of a coalition being formed.

The interactions between the cluster and its members are regulated by a contract. This contract establishes the terms under which the cooperation is established. It includes terms such as the ontologies that must be used by the candidate, the duration, the consideration (a law term that describes what the candidate should give in exchange for joining the cluster, usually the skills that the candidate is bringing to the cluster). The behaviour of a coalition is regulated by another contract that is "signed" by all its members. The important terms of this type of contract, other than the usual ones like duration, names of the members, penalties, etc., are the consideration and the individual skills that each member brings to the coalition. The importance of contracts as a mechanism to create/change flexible and agile control structures (consortia) lays in the fact that the generic behaviours presented by generic agents are constrained by the contracts that each agent has signed. This calls forth the idea that different coalition behaviours can be achieved by just changing the terms of the coalition contract, namely the skills brought to the coalition.

The expectation at this point is that coalitions of agentified manufacturing components, if regulated by contracts, that are declarative and configurable information structures, may lead to significantly more agile manufacturing systems. It is expected that the different ways of exploiting a system depend only on how coalitions are organised and managed. This approach solves the problem of how to create dynamic (agile) structures, but not the problem of how to integrate heterogeneous manufacturing components' local controllers. In order to overcome this difficulty, the process used to transform a manufacturing component into an agent (agentification) follows a methodology to allow their integration (Camarinha-Matos, Barata, & Flores, 1997; Camarinha-Matos, Seabra Lopes, & Barata, 1996).

4.1.1 Cooperation and contracts applied to the shop-floor

The way cooperation is established depends on which tasks each member of the coalition agrees to do. The agreed tasks are nothing other than the skills brought into the coalition. An agentified manufacturing component that wishes to join or leave a coalition should be able to do it without much

of an impact on the coalition's internal structure. Leaving agents might have a stronger impact because if they take with them skills that are important for the coalition operation, this might inhibit its operation. On the other hand, adding new members can be easier because the impact might be just an increase in the potential of the coalition as new skills might be added.

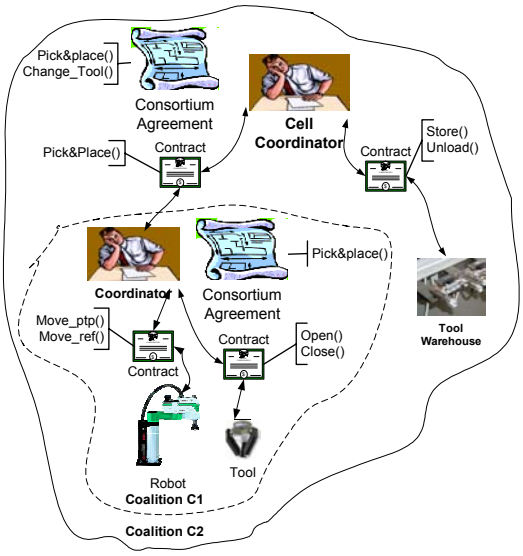

Figure 4-3 - Consortia example

As an illustration, let us consider an example of an assembly cell composed of a robot, a gripper, and a tool-warehouse, organised in a cluster from which coalitions can be formed. The type of cooperation envisaged here follows the Internal Coalition model presented in chapter 3. A first coalition can be established by involving a robot, a gripper, and a coordinator agent, resulting in an entity capable of performing *pick&place* operations (Figure 4-3). This operation is composed of the basic operations *move*, which is provided by the robot agent, followed by the *open*, which is provided by the gripper agent, followed again by another *move*, and finally an *open*. This means that new types of aggregated operations can be built into a coalition.

The behaviour of this type of coalition is regulated by an agreement and individual contracts between the coordinator and the other members. The agreement, among other things, defines the new operations that the coalition can implement. Individual contracts define what (and how) each member is obliged to bring in to the coalition. The individual operations offered by the members are also operations supported by the coalition, i.e., they belong to the coalition core capabilities. The complete competencies for the coalition C1 are *pic&place()*, *open()*, *close()*, *move_ref*(), and *move_ptp()*. In this example, the robot does not bring all its skills (core capabilities) to the coalition. This means, for instance, that this specific coalition C1 is not capable of changing the speed of its *pick&place()* operation. If this was the case the contract established between the coordinator and the robot should include the *set_speed()* operation. A new coalition (C2) can be established composed of C1, a coordinator agent, and another agent (tool-warehouse). The core competencies of the tool-warehouse are the operations *load()* and *unload()*. The new capability of this coalition is the operation

change_tool(), which is composed of basic commands *pick&place()* (from C1), and *store()* and *unload()* (from tool-warehouse). The competencies of C2 are the operations *pick&place()* and *change_tool()*. It should be noted that C2 cannot provide any *move_ref()* competency, because C1 does not agree to give this competency to C2. In the example under analysis, there are two coordinator agents, which can be seen as similar. In fact, they have the same type, because the general role is similar (coordination), but their specific tasks are different, since they depend on the agreements and contracts they have established.

It must be noted that a robot could itself support a *pick&place()* operation, which would occur in the case where the robot was attached to only one gripper that could not be removed automatically. In this case the robot would include this skill as its own and, therefore, could bring it into the coalition in which it might participate.

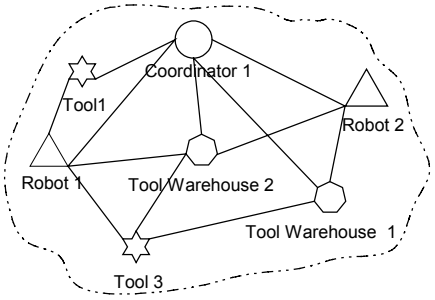

Figure 4-4 - Global coordination

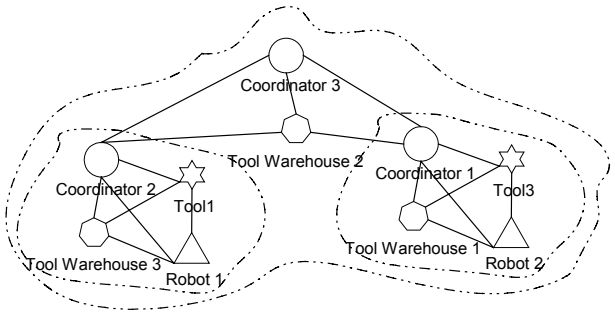

Figure 4-5 - Hierarchical coordination

A coalition must be analysed according to both its cooperation form (discussed in chapter 3) and type of coordination. It is possible to organise coalitions with different types of coordination. The first type of coordination to be analysed is the so-called global coordination (Figure 4-4). In this case, all agents that are required to perform some task or tasks are organised under a unique coalition. To join the coalition an agent must accept the global agreement, and "sign" the contract between itself and the coordinator. The coalition capabilities are composed of all the operations that each individual agent brings to the coalition (defined in the individual contract) and all new operations that are possible to build by compositions of the basic ones. The coalition of Figure 4-4 can perform complex operations, like parallel moves, involving the two robots. One big drawback of this type of structure is the requirements imposed on the coordinator. Large coalitions imply highly complex co-ordinators, and this complexity can be unmanageable. Nevertheless, it can be an adequate structure for small coalitions.

The next coordination type is the hierarchical coordination (Figure 4-5). In the example shown, two coalitions and an individual agent cooperate to compose a third coalition. It should be noted that the number of elements is the same as in the previous case. The main advantage of this structure is that the complexity of the coordination process is divided among the other coordinators. It is interesting to note that the *Robot 2*, for instance, might not be able to store its gripper in the *Tool Warehouse 3*,

which belongs to another coalition. This happens if the contract agreed between *coordinator 3* and the coalition of the *Tool Warehouse 3* does not consider the *store()* and *unload()* operations of this component.

Another type of organisation that can be established is the independent coordination (Figure 4-6). Here the physical system is explored by two different structures that are not related at all. This situation is adequate when it is necessary to explore only parts of the system.

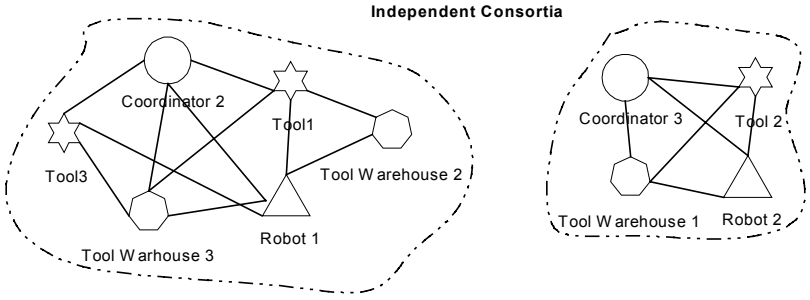

Figure 4-6 - Independent coordination

Although the last type of coalition to be presented is not really a new one, it shows that an agent can be a member of different coalitions (under different coordination). This example could happen also with the hierarchical coalition. Figure 4-7 shows that agents *Tool Warehouse 1*, *Tool Warehouse 2*, and *Tool Warehouse 3* belong simultaneously to the coalition led by *Coordinator 2* and to the one led by *Coordinator 3*. This means that each of these agents has established contracts and agreements with two entities that can ask for services concurrently.

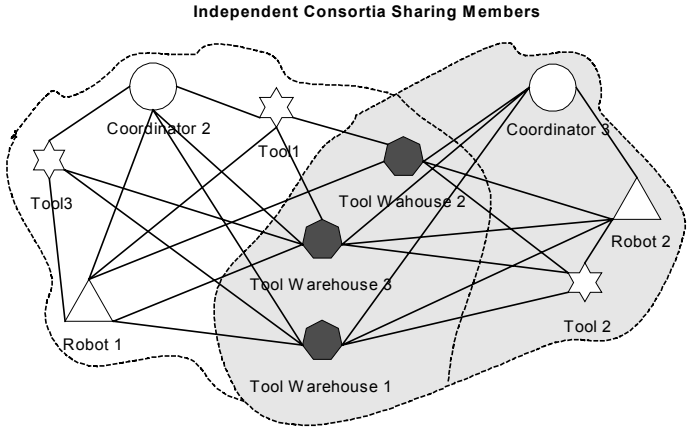

Figure 4-7 - Independent coordination with shared members

Each agent must handle these concurrent requests. An open question related to this case is the priority of the requests. One strategy would be to assign different priorities to each coalition, which should be defined in the individual contract that each agent has to establish. A practical situation where such a structure would be important is when a physical cell has two different areas to be operated separately, but which need to share the three tool-warehouses for gripper exchange purposes.

4.2 THE CoBASA ARCHITECTURE

Shop floor reengineering needs a system based on an appropriate architecture to guide the user in the process of adapting/changing the shop floor. Because this architecture is based on the concept of coalitions of manufacturing components it is called **Co**alition **B**ased **A**pproach for **S**hop Floor **A**gility – **CoBASA**. The base for the agility is provided by the way coalitions can be created, changed, and terminated. CoBASA is a contract based multi-agent architecture designed to support an agile shop floor evolution. It is a multiagent system because its components are agents, as defined in the Distributed Artificial Inteligence (DAI) / Multiagent community (Ferber, 1999; Franklin & Graesser, 1997; Weiss, 1999; Wooldridge & Jennings, 1995; Wooldridge, 2000, 2002). In addition, it is contract based because the behaviour of coalitions is determined by contractual arrangements. The coordination and cooperation of the coalitions and individual agents is inspired by the works of *social order* in multiagent systems (Conte & Dellarocas, 2001b). In the specific case of CoBASA its norms are the contracts that regulate the cooperation and behaviour of the involved agents.

Since a CoBASA system is a community of interacting agents some sort of knowledge sharing is needed to guarantee effective communication and coordination. The various concepts needed by CoBASA (contracts, skills, credits, among others) are supported by ontologies, which can be seen as global knowledge engraved in CoBASA agents.

Finally, CoBASA, can be considered a complex adaptive system that displays emergent behaviour (Johnson, 2001) mainly because this is essentially a bottom up system, in which complex structures (coalitions) are composed out of simpler manufacturing components. This "movement" from lower level structures to higher-level complexity is called emergence.

4.2.1 The components

Before indicating the basic agents that compose the architecture it is convenient to point out the most important concepts behind the suggested approach: **Cluster**, **Coalition**, **Broker, Manufacturing Component, Manufacturing Resource Agents, Coordinating Agent,** and **Contract**. The concepts of cluster, coalitions, broker, and manufacturing component have already been defined in section 4.1. It is now important to understand which of these concepts can be modelled as agents.

An agent called Manufacturing Resource Agent (MRA) models manufacturing components. This agent represents the behaviour of a manufacturing component. In addition it has a social ability (interaction and cooperation with the other agents) to allow its participation in the agent community.

Definition 4.5 – Manufacturing Resource Agent (MRA)
The MRA is an agentified manufacturing component extended with agent like skills such as negotiation, contracting, and servicing, which makes it able to participate in coalitions.

An agent called broker agent (BA) supports the brokering activity, which is relevant in order to create coalitions. The notion of brokers, also known as middle agents, match makers, facilitators, and mediators is a subject of intense research in the multiagents field (Giampapa, Paolucci, & Sycara, 2000; Klusch & Sycara, 2001; Payne, Singh, & Sycara, 2002; Sycara, Decker, & Williamson, 1997; Wiederhold, 1992; Wong & Sycara, 2000).

Definition 4.6 – Broker Agent (BA)
A broker is an agent that is responsible for the creation of coalitions. It gathers information from the cluster and, based on user preferences, supervises/assists the process of creating the coalition.

A cluster by itself is not an agent but rather an organisation of agents. However, an agent might model the activities that support cluster management, such as joining the cluster, leaving the cluster, changing skills, etc. An agent called Cluster Manager (CMgA) models the management activities of the cluster.

Definition 4.7 – Cluster Manager Agent (CMgA)
A cluster manager agent is an agent that supports the activities required by the cluster it represents. This agent stores information about all the MRAs that compose its cluster.

Although a coalition is not an agent, it is one of the main concepts that stand in the background of the architecture being presented. A basic coalition, besides being composed of MRAs, includes an agent that leads the coalition – Coordinating Agent (CA). In addition it can include as members other coalitions. The coordinator of a coalition is able to execute complex operations that are composed of simpler operations offered by coalition members.

Definition 4.8 – Coordinating Agent (CA)
A CA is a pure software agent (not directly connected to any manufacturing component) specialised in coordinating the activities of a coalition, i.e. that represents the coalition.

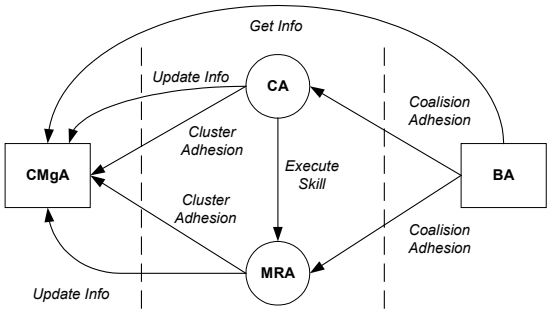

Figure 4-8 – Interactions among the main components

Figure 4-8 shows the main interactions that happen among the mentioned main components of the architecture.

4.2.1.1 The Manufacturing Resource Agent – MRA

Several types of MRAs, one type for each manufacturing component type, can be conceived. Therefore it is expectable to find robot MRAs, gripper MRAs, tool warehouse MRAs, etc. From a control perspective, each MRA is individualised by its basic skills, which represent the functionality offered by the represented manufacturing component.

Each MRA possesses the following basic abilities:

- Adhere to/ withdraw from a cluster
- Participate in coalitions
- Perform the manufacturing operations associated with its skills.

Each MRA that belongs to a given manufacturing cell can participate in the cluster that represents that cell. Therefore, every agent, independently of its skills, can join a cluster as long as it is compatible with the other cluster's elements. Nevertheless, this adhesion is not always guaranteed because the cluster, before accepting a candidate, evaluates its "values". The candidate's value is given by a concept called *credits*, which represents a kind of curriculum vitae. If the curriculum does not reach a certain level the agent is not accepted. Further details about the credit system are given in the clustering section. A negotiation is held between the MRA and the cluster whenever the agent wants to join the cluster. A MRA can join or leave different clusters when the manufacturing component it represents is installed or removed from different manufacturing cells.

All negotiations related to the creation, changing, and termination of coalitions are performed by the MRA. The agent does not automatically choose the skills the MRA brings in to a coalition, which are instead chosen by a user. The MRA participation in a coalition may terminate either because the coalition successfully reached its end or because of an abnormal condition. Performing the manufacturing operations associated with the represented skills is the kernel activity of the MRA. While the other two activities are more related to its social activity, this one represents real manufacturing work. Whenever a robot MRA, for instance, receives a request to execute a *move* command it reacts by sending the appropriate command to the real robot controller that in turn causes the movement of the physical robot.

4.2.1.2 The Coordinating Agent – CA

The CA is, in many aspects, very similar to the MRA. Because it must also be able to join a cluster as well as participating in coalitions, its basic social activity is quite the same. However, there are two differences. First, a CA does not directly support manufacturing operations (skills) but is instead able to create complex skills based on some rules of composition of skills brought in by the members (e.g.

MRAs) of the coalition it coordinates. Second, a CA does not offer manufacturing skills to a coalition except when leading a coalition participating in other coalitions.

The CA has two different statuses: 1) free to coordinate, and 2) coalition leader. When free to coordinate it is just waiting to be a coalition leader. When the CA is eventually chosen to coordinate a coalition its status is changed as well as its situation in the cluster. A CA with a coalition leader status represents a coalition in the cluster.

4.2.1.3 The Cluster Manager Agent – CMgA

The CMgA must support the following basic activities:

- Attend requests for cluster adhesion
- Update cluster-related information
- Provide information to the broker.

Whenever the CMgA receives a request from a MRA or CA to join the cluster it starts the negotiation process that ends either with a refusal or acceptance. Based on the credits of the requester the CMgA decides if the requester is accepted or not. A registry of all agents that constitute the cluster is maintained by the CMgA and, whenever necessary, this information is updated by cluster members. The CMgA also provides all the information needed by the broker agent when creating coalitions.

4.2.1.4 The Broker Agent – BA

The BA creates, changes, and extinguishes coalitions. It imposes the rules that regulate the formation of the coalition through user interaction. As coalitions just represent different logical organisations of manufacturing components, the BA is thus the agent that creates and changes different control solutions. The CoBASA architecture does not assume these tasks are done automatically. On the contrary, it is assumed that an user (systems integrator) is always supervising the process. Even if in future it is foreseen that some of the tasks will be automatised, for now the whole process is manual. The BA gets the information about the set of agents representing manufacturing components (MRAs) from the cluster. A user can then choose the most appropriate ones, and configures the coalition.

4.2.2 Clustering

4.2.2.1 Motivation

The reason to introduce a shop floor cluster in CoBASA is because there is a need for a kind of directory where the agents willing to participate in coalitions can register. This directory acts as the place where those agents become known and where they publish their skills. Every agent belonging to a cluster uses the cluster manager to become known and publish its skills. The cluster structure is also important as a facilitator of the adaptation of its members to participate in coalitions. For instance, the

cluster manager will inform candidate agents about which ontologies need to be followed in order to be able to participate in a coalition. By doing this, the CMgA improves the process of creating coalitions.

In summary, the main advantages of using a shop floor cluster concept are:

- It supplies a directory that allows agents to become known.
- It restricts the set of candidate agents when a coalition has to be formed.
- It speeds up the adaptation process of agents chosen to participate in coalitions.

A MRA cannot participate in more than one cluster simultaneously because the manufacturing component it represents can only be attached simultaneously to one manufacturing cell.

4.2.2.2 A Value System – Credits

The selection of cluster members to participate in a coalition is based on the following priorities:

- The agent type (robot, gripper, …) or the fitness of the agent to the intended goal.
- The agent's specific attributes (in the case of a gripper, for instance, the maximum aperture, the number of fingers, the maximum force, …).
- The past experience or performance record of the agent.

The agent type or its fitness to the intended goal and the agent's own attributes are inherent characteristics of the agent that cannot be changed. However, the same does not occur with the agent's past experience or performance, which varies throughout its life.

To be able to distinguish agents by their performance or experience it is mandatory to create a value system containing several indicators that can be used to measure the agent's quality in terms of its performance or experience. Since this value system qualifies agents in terms of the **credits** each one possess, it is called the CoBASA credit system. If the credits possessed by each agent are regarded as a collection of achievements, then they can be considered the agent's curriculum. The number of participations in coalitions as well as in clusters are an example of good indicators mainly because they quantify experience. In addition, indicators to measure how well the agent has performed (agent's performance) its duty as a coalition or cluster member are needed. Therefore, a **mark** expressing how well an agent has performed its task is given to it after its participation in a cluster or coalition. Nevertheless, the overall quality measure of the agent can be increased or decreased if other indicators are considered, as is the case with indicators related to the operation of the physical manufacturing components associated with each MRA. Number of breakdowns, number of working hours, number of hours on standby, average time to get repaired, consumption of energy, etc. are just examples of such indicators.

The agent's individual curriculum is analysed by the CMgA during the adhesion process. In this way the CMgA can decide to accept only those agents with a given quality level. This attitude avoids low quality physical manufacturing components being installed on a manufacturing cell without any

warning. As it is the CMgA that establishes the threshold level of acceptance for the manufacturing cell, certain manufacturing cells with fewer requirements might allow lower quality level agents.

Please note that this warning mechanism is useful in practice as different human operators interact with the process of change the cell during its life cycle. CoBASA supports, therefore, the maintenance of accumulated knowledge on the cell.

Agent's credits are also used when a coalition is created. The user can choose the members to participate in a coalition based on the credits each agent possesses. The practical effect of this situation is that more qualified agents have better chances to participate in coalitions. It is easy to find a direct relation between the credit system and the maintenance department. Fewer fitted agents indicates that the maintenance department should intervene in the equipment represented by that agent.

Looking back to the claim that CoBASA can be seen as a complex adaptive system, the credits represent a kind of value to express the fitness of the members (the manufacturing components) to participate in coalitions or, in other words, express how well MRAs contribute to a successful society (coalition).

4.2.2.3 Cluster Adhesion Contract - CAC

Contracts, as explained before, provide a framework to rule the agents' behaviours. As far as the cluster is concerned the relationship between the agents MRA and CA on one side and the CMgA on the other side is governed by a **cluster adhesion contract (CAC)**. In this type of contract the CMgA imposes all terms and conditions on the others (CAs or MRAs). An adhesion contract is used because it is the CMgA that knows what basic requirements the candidate members must follow to be accepted (the conditions). The only choice candidate members have is accepting or refusing it.

Modelling

Since contracts should contain more information than just the duties and obligations the cluster adhesion contract is composed of the following parts:

(i) General Conditions.

This part describes the purpose, definitions, and the ontology that validates the definitions and concepts of the contract. The parties, the contract identification as well as the starting date and the duration of the contract are also included in this part.

(ii) Consideration.

Both considerations from the offeror (CMgA) and from the offeree (MRA and CA) are described in this part. The agent's consideration is the set of skills that it is interested in bringing in to the cluster, and that might be used by future coalitions. Although the consideration from the cluster side could be just to let the agent belong to the cluster (implicit consideration), it was decided to define an explicit consideration – cluster skills are represented by some clustering functionalities or services. An example of a service made available by the CMgA to the

contracting MRAs is skills updating. If for some reason this service is not in the contract, the MRA is not allowed to update its skills in the CMgA.

(iii) Contract Logistics.

An important parameter defined here is the *clusterWorkingPlace* that defines the physical area covered by the cluster. An agent candidate to join the cluster must check the adhesion contract to see if it is installed in the area covered by the cluster it is trying to join. The addresses of the parties are also defined here. An address in this architecture is the Agent Identification Descriptor (AID) within the multiagent-supporting infrastructure, which is needed in order to send/receive messages. The maximum number of repetitions for each message (tries) before the sender party has the right to breach the contract claiming that the receiver party is not performing well is also defined. Related to the liveliness of agents is the parameter *maximumTime*, which describes how long the sender party should wait before considering repeating the message.

(iv) Exceptions & Terminations.

This part describes the behaviour that must be followed when the contract is not completely performed or when the contract reaches its end. The contract can be terminated for three reasons: 1) discharged by frustration, 2) discharged by performance (the contract reaches its end), and 3) breached (interrupted) by bad behaviour. The breach by frustration occurs whenever a party cannot perform its obligations because of an unexpected event, for which it is not responsible. In this case the other party can only claim for light or even no remedies. The worst situation occurs when breaching by bad behaviour; in this case the remedies to be claimed by the offended party can be very high. It must be pointed out that the non-existence of a breach or discharge from the offeror (the CMgA) occurs because it does not make sense to penalise the cluster. From a law point of view the dominant partner can impose an adhesion contract where it cannot be liable for its own bad behaviour.

A representation for the cluster adhesion contract is shown in Figure 4-9 using Unified Modelling Language (UML) class diagrams. This formalism was chosen because UML is adequate to model structures following the object-oriented paradigm, which, on the other hand, is adequate to represent an abstraction of a contract. The focus of the contract model is purely static because what is important to discuss now is the attributes that compose it. The functionality associated with the CAC is discussed when the life cycle is discussed as well as when the agents that deal with it (CMgAs/CAs/MRAs) were described.

By focusing only on the attributes, the UML class diagram can also be used to represent data structures based on the frame paradigm developed by Marvin Minsky (Fikes & Kehler, 1985; Hayes, 1980; Minsky, 1975), which is not very different from the object-oriented paradigm. However, the frame paradigm provides much better support for relations than the object-oriented paradigm. In this paradigm the representations of the world (concepts) are modelled by structures called *frames*, instead

of classes. Nevertheless, in the aspects discussed here, the reader will not notice any difference from the object-oriented paradigm.

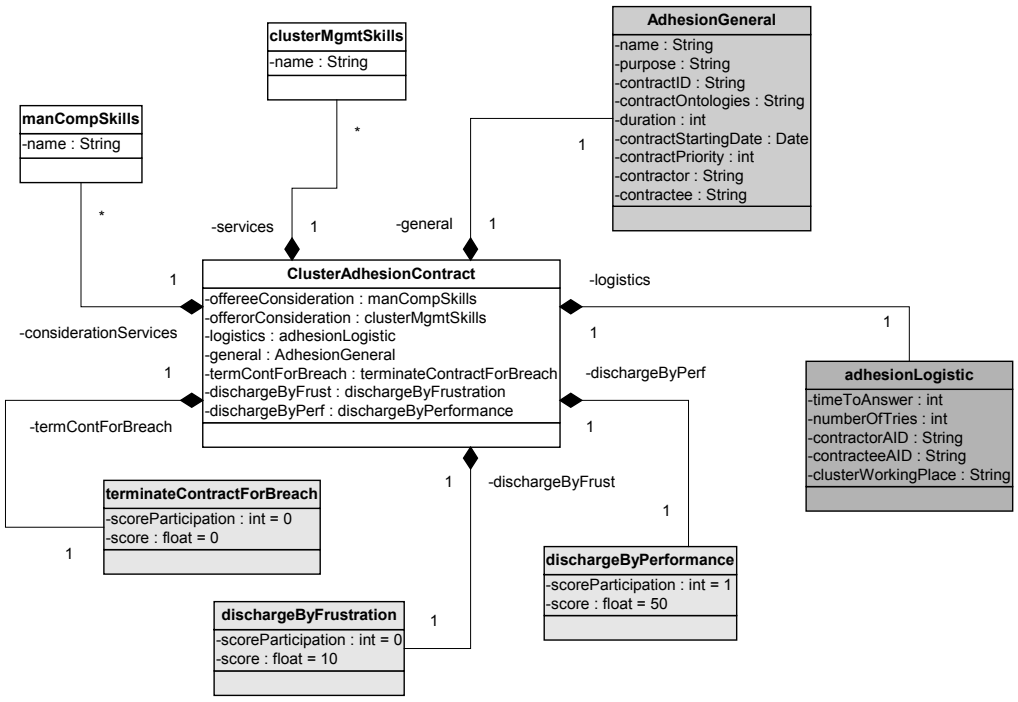

Figure 4-9 – The Cluster Adhesion Contract (CAC) model

The attributes (slots) that model the contract are:

- ***general*** – contains the general conditions of the contract, which are represented by an instance of the concept *adhesionGeneral*, which is composed of the following slots:
  - *purpose* – the purpose of this contract
  - *contractor* – the name of the CMgA
  - *contractee* – the name of the candidate (MRA or CA)
  - *contractID* – an identifier for the contract
  - *contractOntologies* – this is the set of ontologies that candidates must be able to deal with. This slot guarantees that the agents that belong to the cluster can understand all the important concepts used within the cluster.
  - *duration* – indicates the contract life. This attribute will be checked by the CMgA to verify if the contract is still valid. Contracts that reach their end will be terminated.
  - *contractStartingDate* – another slot related to the validity of the contract. The date when the contract was established is stamped on this slot.
  - *contractPriority* – indicates the priority level of this contract. This is used by the CMgA to decide which requests should be performed first when several cluster members ask for cluster services.

- ***offerorConsideration*** – The set of services (skills) offered to the adhering agents by the CMgA. Each of these skills is an instance of the *clusterMgmtSkills* concept. Further details of skills will be given in the subsection related to coalitions/consortia. A member can only request from the CMgA the services expressed in this slot.
- ***offereeConsideration*** – The set of skills brought to the cluster by candidate agents. The slot contains an instance of the *manCompSkills* concept. Because this concept or its subtypes represent all the skills that can be found on manufacturing components the contract can be used to regulate the behaviour of all kinds of MRAs. The candidate agent decides what skills will be included in this slot during the adhesion process (negotiation). Therefore, these are the only skills that can be offered by the agent when participating in coalitions. The BA uses this information (the skills brought in) when coalitions are created.
- ***logistics*** – Contains the aspects related to the communication and the cluster work space. Its content is an instance of the concept *adhesionLogistic* that is composed of the following attributes:
    - *timeToAnswer* – An imposition of the CMgA stating the maximum time a cluster member has to reply after receiving an enquiry from the CMgA. If this time limit is not complied with the CMgA can eventually terminate the contract.
    - *numberOfTries* – Another imposition of the CMgA, which is the maximum number of times the CMgA tries a request before breaching the contract for bad behaviour. A new try is only made after *timeToAnswer* has elapsed.
    - *contractorAID* – The communication address of the CMgA. This is used whenever a cluster member wants to send a message to the CMgA.
    - *contracteeAID* – The communication address of the CA or MRA that is used whenever the CMgA wants to send a message to the cluster member.
    - *clusterWorkingPlace* – This slot indicates the spatial area where the candidate MRA is installed. It is filled in by the candidate and it is important for the CMgA to determine if the MRA "lives" in the area covered by the cluster, i.e., if the manufacturing component belongs to the physical manufacturing cell covered by it.

The next three slots are related to contract termination. They are used during the contract termination phase. Further details about them are given in the next section.

- ***dischargeByPerf*** – This slot is used when a normal termination takes place (the contract duration is over). It contains an instance of a reward concept to be added to the CA/MRA's curriculum at the end of the contract.
- ***DischargeByFrust*** – This slot is used when an abnormal termination occurs, or more precisely when the CA/MRA requests the CMgA to terminate its contract by frustration. It contains an instance of a penalisation concept to be added to the MRA's curriculum.

- ***termContForBreach*** – This slot is also for an abnormal termination but, in this case, the termination is a consequence of a contract breach from the CMgA. It contains an instance of a penalisation concept to be added to the MRA's curriculum.

Contract Life-Cycle

The cluster adhesion contract's life-cycle is composed of the following phases: creation, performance, and termination. Since cluster members have their membership regulated by contracts, these phases are tightly connected to those phases of the CA/MRA's life cycle that interact with the cluster. The contract creation phase occurs during the CA/MRA cluster adhesion phase. They are so connected that this phase is discussed when the CA/MRA adhesion interaction is described. The contract performance phase runs while CAs/MRAs are full members of the cluster. Contract performance corresponds to the execution of cluster services by the CMgA. Consequently, this phase is also discussed by the time the CAs/MRAs membership cluster interaction is described. Finally, the contract termination phase occurs when CAs/MRAs leave the cluster. There is an evident connection between contract termination and the interaction between CAs/MRAs and the CMgA when the former leaves the cluster. However, there are some aspects of contract termination not related to cluster interaction that need to be discussed.

In the CoBASA architecture the CAC terminates for three reasons: performance, frustration, and breach. Termination by performance is a normal termination and takes place when the contract duration is over. When the CMgA detects that the contract is obsolete it sends a request to the other part (contractee) to terminate the contract. Termination by frustration is an abnormal termination because it happens before the agreed contract finishing date. Frustration in the context of the CAC happens when the agent (MRA) is forced to leave the cluster by reasons it cannot control. When a gripper, for instance, is moved from a manufacturing cell to another cell due to external reasons, the MRA that represents that gripper is forced to terminate its cluster contract because it no longer belongs physically to it (the cell). In this situation the MRA sends a request to the CMgA to terminate its contract invoking frustration. Termination by breach is a serious abnormal condition that happens for two main reasons:

1. When a MRA is found to have a curriculum inferior to the minimum required in that cluster. Please note that the curriculum of a MRA may become worse if its participation in coalitions is poor. The agents becoming poorer in performance should not be members of the cluster because this leads to poor coalitions. The CMgA is responsible for detecting the agents whose curriculum has been scaled down below the threshold level. This can be done because agents are obliged to inform the CMgA whenever there are changes to their credits.
2. When the BA informs the CMgA that one member refused to participate in a coalition without a valid reason. Cluster members have signed a contract to participate in coalitions (CAC), therefore, if they refuse, the CMgA has the right to breach its CAC.

There is a close connection between the CAC and credits (agent's curriculum). If credits are regarded as a kind of measure of performance it is quite natural that, at the end of a contract, credits are

updated, corresponding to a sort of curriculum updating. This happens independently of the termination type, either normal or abnormal. A contract terminated by performance might be regarded as a successful one because this means that the contractee agent (MRA/CA) has accomplished all its promises. Therefore it is natural that this agent could add some positive points to its curriculum. On the other hand, under an abnormal termination it is natural that penalisation takes place. This rewarding/penalisation step at the end of every contract guarantees that the agent's curriculum is a mirror of its performance. When the members of the cluster adhere to a cluster by accepting the CAC they "know" exactly what the penalisations or rewards they receive are because this is registered in the slots *dischargeByPerf*, *DischargeByFrust*, and *termContForBreach* of the contract. When the contract is terminated by performance, the contractee (MRA) updates its credits based on the information available in *dischargeByPerf,* which is a prize. When the termination is by frustration the penalisation updating is made using *DischargeByFrust*. Finally *termContForBreach* is used as a severe penalisation when the contract is breached.

Discussing the representation type for the prize or penalisation is not important while discussing the architecture. A suitable representation for these concepts will be made in the chapter related to implementation issues. However, this representation should be closely related to the representation of the credit system. This means that some of the concepts used in the credit system should also be used in the prize/penalisation.

Discussing penalisations associated with abnormal termination raises the question of making sure that the penalties are effectively self-inflicted by the agents (added to the agent's curriculum). The experience with human beings suggests that it is not sufficient to include penalisations in contracts for them to be consummated by the offender. The same type of question can be raised for rewards because it is not possible to guarantee that the reward is not inflated to enrich the curriculum. Since human beings are not perfect, and sometimes resort to cheating, human societies resort to independent institutions to guarantee the proper execution of rewards and penalisations. An example of such an institution is the court of justice. In the CoBASA society of agents it was decided not to use an additional agent to perform the role of court of justice (to simplify the architecture). The integrity of the society is guaranteed by the honesty of the involved agents that update their credits, based exactly on what is established in the contract, an assumption that is reasonable in this context. This requirement is feasible with software agents as long as the programmer codifies a completely honest behaviour for the agents. If this code is not honest, without external institutions (agents) it is impossible to guarantee the integrity. However, it was decided to take this risk for the sake of simplicity and because the main theme of this thesis is not security.

4.2.2.4 Cluster Interactions

The cluster interactions can be grouped by the life-cycle phases of MRAs/CAs in relation to the cluster. The phases that are being considered are: 1) adhesion to the cluster, 2) membership, which

happens when the agent is a member of the cluster, and 3) terminating, which happens when the MRA/CA leaves the cluster (terminating). The interactions are shown below using UML Sequence Diagrams.

Adhesion. This phase starts when the candidate sends the CMgA a REQUEST message asking permission to join the cluster. If the CMgA determines that this agent is not a valid agent because it does not accomplish an eligibility criterion (it can be a list of agents that have behaved badly in the past – blacklist), a REFUSED message is replied stating that the agent is not welcome (Figure 4-10). If the agent accomplishes the criterion, the CMgA queries the candidate for its credits. After the MRA/CA replies informing of its credits the CMgA evaluates the credits and decides if they are above the minimum acceptance level. If this is not the case, a FAILURE message is sent to the candidate stating that the cluster did not accept its REQUEST to join the cluster (Figure 4-12). When the credits of the candidate agent are accepted, the CMgA fills in the cluster adhesion contract (CAC) terms but the term *offereeConsideration* that refers to the skills that are brought in by the candidate, which should be filled in by the MRA/CA. Next the CMgA sends a REQUEST message to the candidate asking it to accept the contract. This corresponds to an offer in legal contractual terms. The MRA/CA evaluates the contract offer and decides if it can accomplish all its terms. If not, the candidate sends a FAILURE message to the CMgA stating that it does not accept the offer. In that case a FAILURE message is also sent to the candidate stating that the cluster manager did not accept its REQUEST to join the cluster (Figure 4-11). If, on the other hand, the candidate MRA/CA decides to accept the CMgA offer, it sends an INFORM message stating its acceptance. The CMgA then sends a final INFORM message to the candidate stating that its initial REQUEST to join the cluster has been accepted (Figure 4-13).

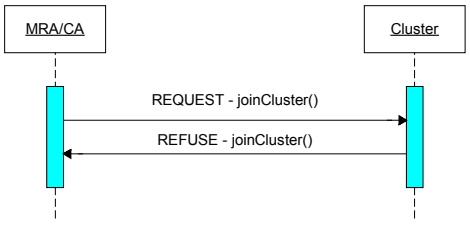

Figure 4-10 – Adhesion refused by eligibility criteria

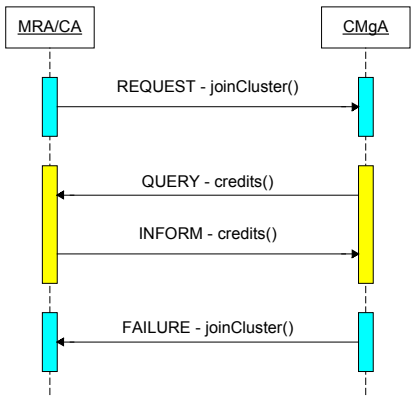

Figure 4-12 – Adhesion refused because of bad credits

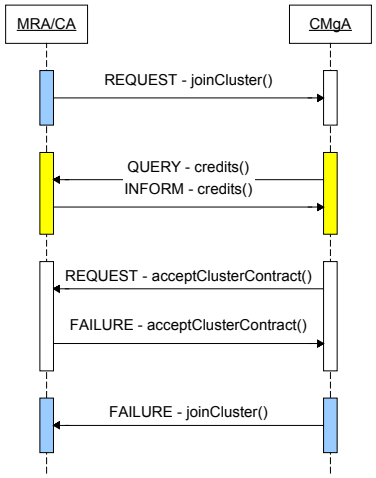

Figure 4-11 – Unsuccessful cluster adhesion

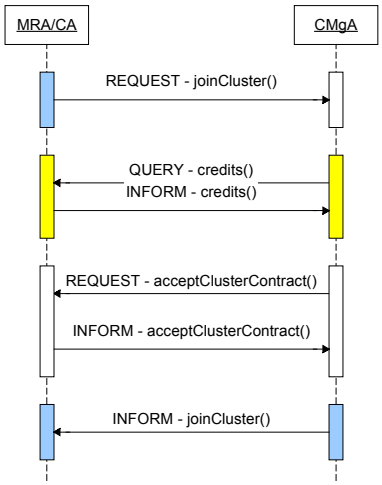

Figure 4-13 – Successful cluster adhesion

Membership. While a member belongs to a cluster it might request the CMgA for services promised in the CAC. The requests pattern can be seen in Figure 4-14. Although not shown in Figure 4-14, the requests are consequences of other actions that happen inside the CA/MRA. For instance, an update of credits is made whenever a coalition is finished.

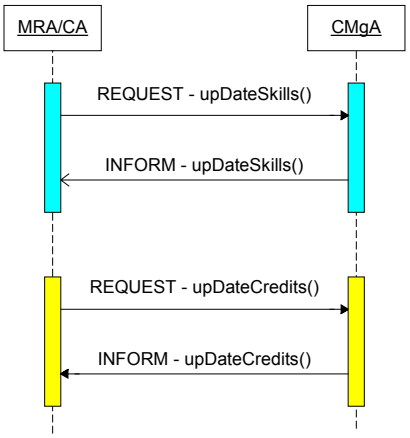

Figure 4-14 – Services requested by a MRA/CA

Terminating. This is related to CAC termination and occurs when the MRA/CA is leaving the cluster. As mentioned before, the agent can leave the cluster for three reasons:

1. The CAC terminates by due time (performance).
2. The agent is forced to leave because the CMgA breached the contract (bad behaviour from a CA/MRA).
3. The agent was physically removed and then it cannot assure any contractual relationship (frustration).

When an agent leaves a cluster it might affect existing coalitions. However, the affection level is not the same in the three situations. If a cluster member leaves because of due time it is not necessary to immediately abandon the coalitions it is participating in. The only consequence is that this agent is no longer available to be gathered in future coalitions.

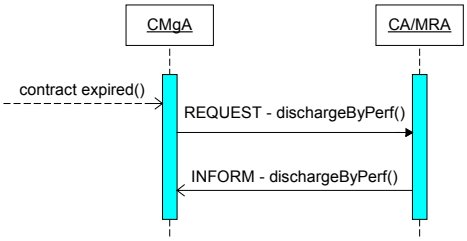

Figure 4-15 – Leaving because the CAC has expired

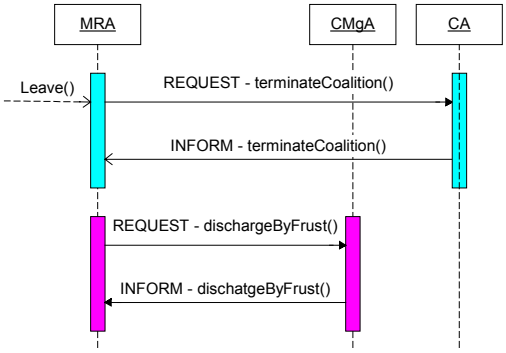

Figure 4-16 – Leaving because of frustration

To terminate the contract by performance the CMgA just sends a REQUEST message to the leaving member asking to terminate the contract. When it receives the INFORM message, all the information related to the agent is erased from it. An agent cannot refuse to terminate its CAC (Figure 4-15).

Leaving the cluster because of frustration is different because, in this case, it is the MRA that initiates the termination process rather than the CMgA, and any of its participations in coalitions must be aborted. When an user indicates to the MRA that its corresponding manufacturing component needs to be removed, the agent should first verify if it is participating in coalitions and, if yes the agent sends a REQUEST message to the CAs coordinating the coalitions in which it participates. After receiving from each CA an INFORM message, the agent knows it is no longer a member of any coalition and can then send a REQUEST message to the CMgA asking to terminate the contract by frustration. As before, when the CMgA receives the INFORM message it erases the information related to the leaving member (Figure 4-16).

Leaving by bad behaviour is slightly more complex because this kind of termination can be detected in two different ways. In the first one the CMgA detects that the quality level of the agent is below a threshold level, based on its credits, and so it decides to expel that agent from the cluster. In this situation it is not necessary for the agent to immediately terminate its participations in coalitions. Even though the agent is becoming "poorer", it can still contribute to the coalitions it is participating in. Nevertheless, the agent is no longer able to participate in new ones. It can be argued that the credits of the agent could be improved at the end of the coalitions the agent was participating in, which is clearly a situation that might happen. Although it might look strange as to why an agent that has improved its curriculum is still expelled from the cluster, the reason behind it is to make sure a user inspects the agent. Figure 4-17 shows that the REQUEST message sent to the MRA/CA is a

consequence of the credits updating REQUEST message. The second way to force the withdrawal of a CA/MRA is because of its refusal to participate in a coalition (Figure 4-18). When the BA detects an agent, member of a cluster, refusing to participate in a coalition[29] of that cluster, it informs the CMgA by sending an INFORM message. The BA detects an agent refusal by receiving a REFUSE message, after a REQUEST asking that agent to participate in the coalition being created. After being informed, the CMgA sends a REQUEST message to the faulty agent asking it to leave the cluster based on bad behaviour. Before replying to the CMgA, the faulty agent is obliged to leave the coalitions it is participating in, declaring bad behaviour. This is done by sending a REQUEST message to the CAs leading the coalition the faulty agent belongs to. After receiving the last INFORM message stating it has been removed from all coalitions, the faulty agent can now send back an INFORM message stating that he has left the cluster.

The reason why the agent is obliged to immediately terminate its participation in coalitions is due to the severity of refusing to participate in a coalition, which is the basis of the CoBASA operation. An agent that refuses to accept the most important rule, should be immediately banned.

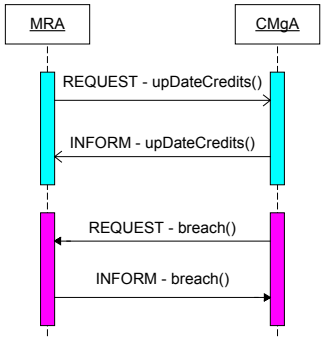

Figure 4-17 – Leaving by poor credits

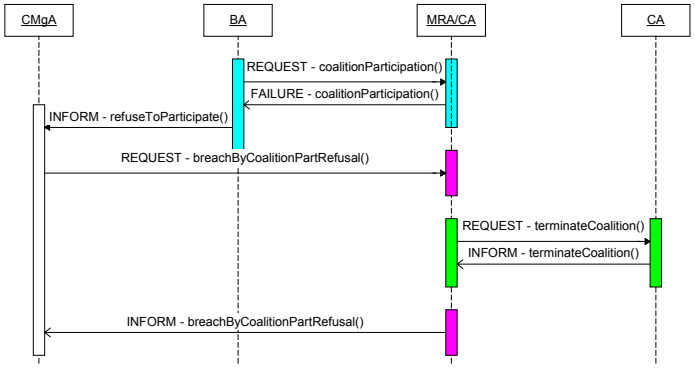

Figure 4-18 – Leaving by refusing to participate in a coalition

4.2.3 Coalitions / Consortia

4.2.3.1 Organisation

As members of coalitions, MRAs can only play the member role whilst CAs can play both the coordinator and member roles. A simple manufacturing coalition is composed of some MRAs and one CA. However, a coalition can be composed of other coalitions, creating, in this way, a hierarchy of coalitions. Therefore, a CA can simultaneously coordinate MRAs and others CAs (Figure 4-19). In this figure CA2 is simultaneously a member of *coalition 1*, and the coordinator of *coalition 2*, composed of MRA B and MRA C. Please note that *coalition 1* is composed of MRA A and CA2. CA1 does not have direct access to the members of *coalition 2*.

[29] A refusal is sensed by the BA either by the REFUSE message or by no response at all.

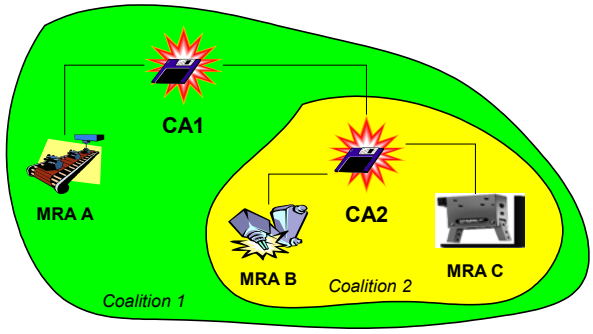

Figure 4-19 – Hierarchy of coalitions/consortia

A coalition needs a CA, instead of only MRAs to reduce the complexity of a MRA. If the coalition was only composed of MRAs, the complex task of coordinating a coalition would be added to the usual tasks such as controlling the manufacturing component, negotiating cluster adhesion and participating in coalitions, etc. Among other things, a coalition coordinator needs to generate new skills, and should be simultaneously member and coordinator. Please note that skill generation is not the only problem since the way skills are composed and represented in order to be executed properly is not a trivial task. Separating the functionality related to coordination from the one related to executing commands simplifies the architecture of the individual agents. MRAs become less complex at the expense of introducing another agent type, the CA.

Coalitions contracts represent simultaneously a coordination mechanism and a means to facilitate coalition dynamics. Since a coalition is created, changed, and terminated mainly through contract related operations, the task of grouping manufacturing components able to perform certain tasks (coalition) is facilitated because it involves only configuration (contract configuration). Agility is thus achieved since moving components from one organisational form to another involves only configuration instead of programming effort.

As CAs are able to generate new skills (composition of skills) from the set of skills brought in by its members, coalitions enable the creation of different control structures, whose global functionality comprehends the set of basic functionalities brought in by the coalition members plus the generated complex functionalities. This could not ever be achieved by a traditional control architecture because of its rigidity. Traditional control approaches need to know, in advance, the logical organisation of the components as well as the complete set of skills that need to be controlled.

Considering this agility at the coalition level and that coalitions can be composed of other coalitions, the next question is what impact a change on a coalition has on the others this one might be related to. After a change in a coalition (addition or removal of members), the skills its CA is able to perform are likely to change. They can be either increased, reduced, or in some situations, kept. The last situation occurs when a component that brings no value to the coalition is introduced or removed. If a coalition participating in another coalition looses skills, then it is necessary to verify if any of the

lost skills were offered to any other higher-level coalition or are used in the composition of complex skills also offered to other higher-level coalitions. If this happens, a renegotiation process must be started with the higher-level coalition to verify the impact of the alterations and, if necessary, renegotiate with its own higher-level coalition(s). This process is expanded through all the levels until reaching the upper one. The removal (or addition) of a manufacturing component (MRA) (its skills) provokes the automatic updating of the higher-level skills that could be direct or indirect depending on the ones that were removed (added). If instead of loosing a component the coalition acquired a new one a similar behaviour happens. However, in this situation, the higher-level nodes might be enriched with new skills. The aspects related to skill updating are discussed below.

Considering:

S - The set of skills that compose the manufacturing world under consideration

$S_{generated}$ - The set off skills that it is possible to generate

$S_{MRAallskills_i}$ - The set of skills the MRA i has.

$S_{MRA_{i,j}}$ - The set of skills the MRA i contributes to coalition j.

S_{CA_j} - The set of skills of the CA in consortium j.

$S_{CAmembers_j}$ - The set of Skills brought to the consortium j, led by CA, by its members

$S_{CAgenerated_j}$ - The set of Skills generated by the CA to consortium j

$S_{CAoffered_{i,j}}$ - The set of Skills the consortium i, led by CA, offers to the consortium j

then:

$S_{MRAallskills_i} \subset S$ $\quad S_{MRA_{i,j}} \in \wp S_{MRAallskills_i}$ $\quad \wp$ = Power set

$S_{CAgenerated_j} \in \wp S_{generated}$ $\quad \bigcup_j^n S_{MRA_{i,j}} \subseteq S_{MRAallskills_i}$

$S_{CAmembers_j} = \bigcup_i^n S_{MRA_{i,j}} \cup \bigcup_l^n S_{CAoffered_{k,l}}$

$S_{CA_j} = S_{CAmembers_j} \cup S_{CAgenerated_j}$

$S_{CAgenerated_j} = generateSkills : S_{CA_j} \rightarrow S_{generated}$

$S_{CAoffered_{i,j}} \subseteq S_{coalition_j}$

It is important to remember that the skills offered by a CA to the coalitions at a higher-level are a subset of the skills possessed by the CA. Another relevant statement is that the set of skills offered by a MRA to a coalition j belongs to the Power Set of the skills possessed by that MRA, which means that a MRA might offer part of its skills. The skills brought into a coalition j led by CAj are the union of the skills brought in by all MRAs that belong to the coalition j plus the skills offered by the various

coalitions that might be participating in coalition j. This means that a complex skill can depend on another complex one. With this notation it is possible to see what happens to the set of skills when there exists the situation shown in Figure 4-20.

The figure shows that the skills offered by the *coalition 2*, when involved in *coalition 1*, are a subset of the skills the coalition possesses, which is perfectly valid. The skills to be offered are chosen by the broker during the coalition creation phase. The generation of skills is based on a set of rules that are part of the CoBASA knowledge base. For instance in *coalition 1*, according to the rules illustrated in Figure 4-20 only the rule "s8 = f(s7,s1)" can be fired and thus s8 is the only generated high level skill. All the other rules require input skills that are not present.

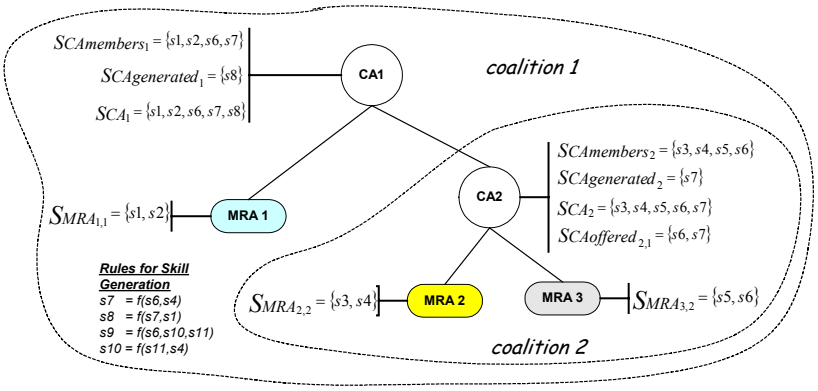

Figure 4-20 – A coalition in its initial situation

The effect of coalition dynamics in CoBASA can be verified by analysing what happens when a new component is added, for instance to *coalition 2* (Figure 4-21). The introduction of MRA 4, which brings in the new skill s11, causes an alteration to the set of skills handled by CA2. It can be seen that the set of skills for the *coalition 1* is increased. The update is automatic because it is only connected with the generation of complex skills and renegotiation between coalition leaders.

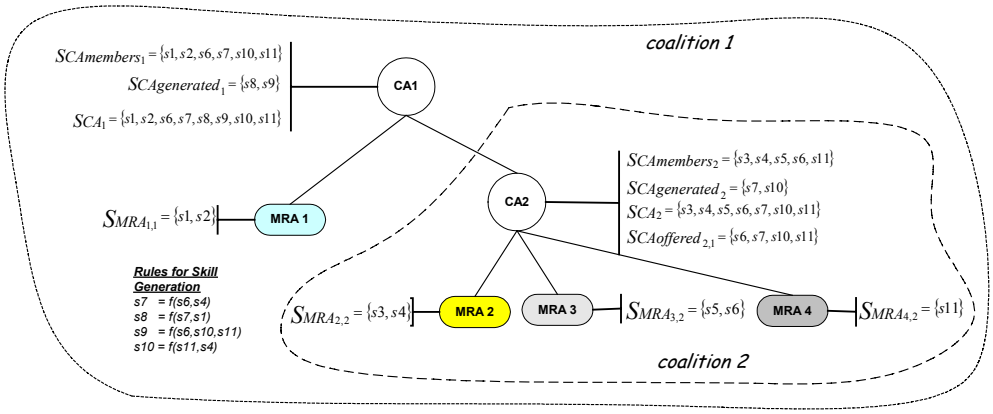

Figure 4-21 – The same coalition after introducing another element MRA 4

Considering now the removal of a component (MRA 3, for instance), it causes a reduction of skills both in *coalition 1* and *coalition 2* (Figure 4-22). From this discussion, it is now possible to better

understand why the CoBASA architecture can be considered a complex adaptive system. When a complex skill owned by the coalition leader is executed this can be regarded as a behaviour that results from the interaction of its members.

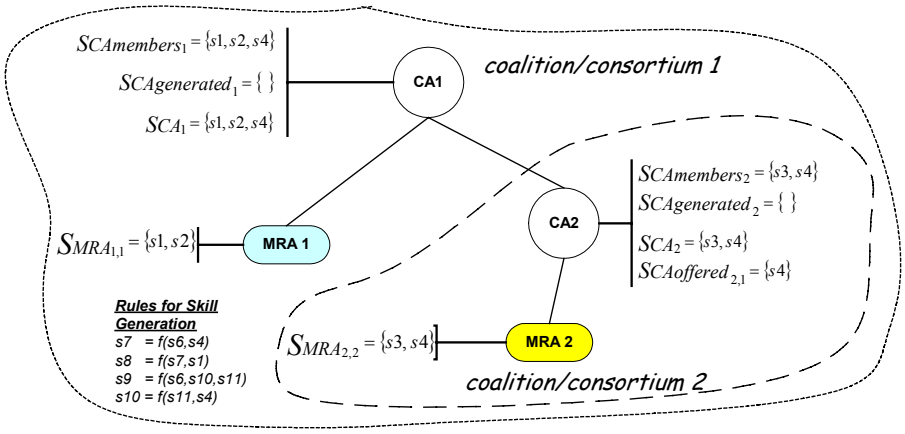

Figure 4-22 – The coalition now after removing MRA 3

4.2.3.2 Skills

Up to now skills have been generally identified as agents' abilities. This section further details what these skills are and shows ways to model and generate composite skills.

Identifying skills is not as easy as it would seem due to the different areas involved in manufacturing in which there are diversified processes that lead to different manufacturing cultures. Although CoBASA intends to be comprehensive enough to cover different manufacturing areas, this description is mainly focused on assembly since the testing platform, the NovaFlex cell (Barata & Camarinha-Matos, 1994), is essentially an assembly system.

Basic skills are in general easier to be identified since they are directly associated with the manufacturing components' capabilities or functionalities. It is advantageous to name skills after the name of the functionalities of manufacturing components, as shop floor people know them, because this facilitates their recognition. Complex skills, on the other hand, are more difficult to be identified because they are more independent from the manufacturing devices and are more dependent on the process. Therefore, a complete taxonomy of complex skills is only possible if the various manufacturing processes are structured, which is not the case. The assembly process, for instance, is far from being structured in spite of the many efforts that have been made (Onori, 2002; Vos, 2001). Although, these efforts do not provide a complete taxonomy of complex skills, it was possible to define a particular set of complex skills that are relevant in the context under which CoBASA has been tested. Please note that the structure of the assembly process is beyond the aim of this thesis.

In the assembly context, for instance, skills (either complex or simple) can be seen as a representation of an assembly operation. As an example, if a coalition includes all the required components to assemble a product (a robot, feeders, a fixture, tools, tool exchange mechanism, and tool warehouse) a possible set of complex and basic skills that it should include are indicated in Figure

4-23. This figure shows how complex skills are composed and which agents support each basic skill. For instance the highest-level skill is *assembleProdSk*, which enables the coalition to assemble a product. This skill is composed of the basic skill *fixSkill* (actuate the fixture) and the complex skill *assembleCompSk*, which is the skill that allows one component of the product to be assembled. Please note that *assembleProdSk* is composed of one or more *assembleCompSk*, and that Figure 4-23 does not indicate the structure of each skill but only those skills that participate in the composition of complex skills (relation *composed of*).

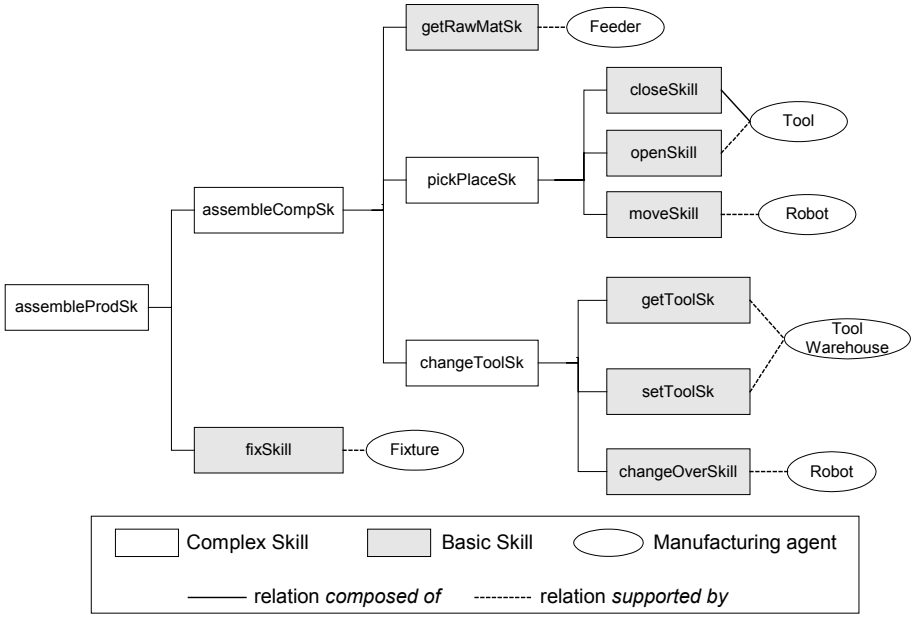

Figure 4-23 – Skill composition and relationship with agents

The concept name indicates the functionality (operation) associated with it while the representation (attributes) allows a distinction between two skills that belong to the same concept. This is very important because different instances of manufacturing components might share the same skill type but with different characteristics (attributes).

Figure 4-24 – Gripper DPZ 64

Figure 4-25 – Gripper PGH30

A very simple case that helps to illustrate this situation is, for instance, the case of the grippers DPZ64 and PGH30 from SCHUNK. In spite of sharing the same type of skills (open and close) these two grippers cannot handle the same type of objects for the simple reason that their maximum aperture, holding force, and number of fingers are different. Thus, skill attributes help to decide which is the component that has the skills that best fits the given requirements as well as to distinguish between manufacturing components of the same type.

Figure 4-26 shows an example of the taxonomy of skills that is needed in CoBASA. Complex skills can inherit the attributes of the skills that compose them. This is illustrated by the *pickPlace* skill, which is a composition of the *grabSkill* and *moveSkill*. The figure also shows an instance for each represented skill.

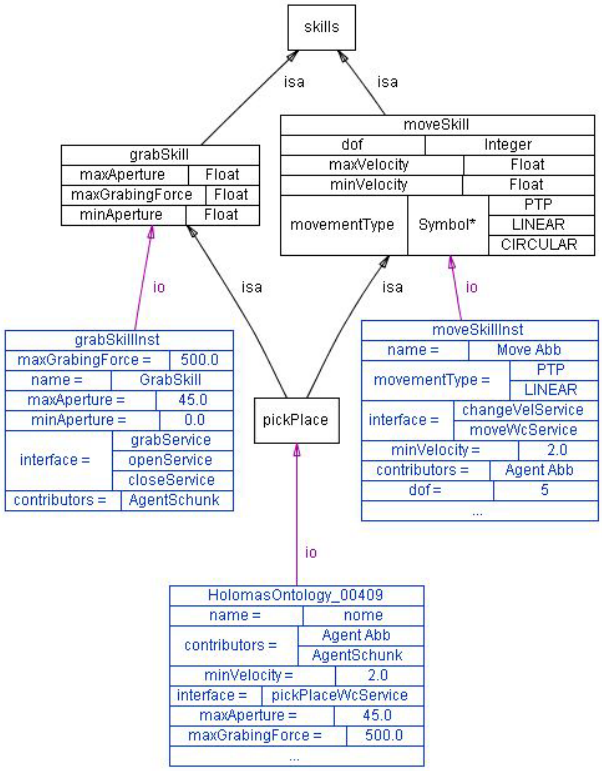

Figure 4-26 – Skill representation using frames

Skills descriptions help the creation of manufacturing coalitions as they are a major element when searching for adequate candidates. However, this is not their only role, since they are also useful when the coalition is being operated (operational phase). This is so because skills also represent the commands to be used within coalitions (services). The question here is how the CA reacts when it receives a request to perform a certain task according to the skills it offers. These requests correspond to tasks (commands) being asked by the higher-level CA. Therefore, a command can be regarded as an execution of an offered skill.

When a coalition led by a CA_n is requested to perform some task associated with one of its skills, the way it behaves depends on the type of skill that was requested. If the skill being requested is not generated by CA_n the action consists simply in redirecting the request to the member of the coalition that has offered it to the coalition. If, on the other hand, the skill is generated within CA_n (i.e. a complex skill) then the procedure is more complex because, instead of just a redirection, it must be decomposed into its basic skills that must be sent to the right destination (the agents that offered the basic skills) in the proper order. Consequently a representation model indicating which the basic skills are and how they interrelate is needed. This model should allow the representation of sequential and

parallel actions as in a graph, which can be regarded as a program model that controls the way the basic skills that compose the complex one are executed. Therefore, the CA needs to include an interpreter machine that is able to execute the mentioned graph model. In an analogy with a computer, the graph representation represents the program, while the interpreter represents the CPU. Changing the model (the skills) changes the actions of the executing machine (behaviour). If each CA embeds a generic execution machine, like a Petri Net executor (Zurawski & Zhou, 1994), or even a workflow engine (WFMC, 2002) able to execute Petri Nets or Workflow models, then the CA is transformed into a kind of generic machine that can work with different types of skills.

The model that represents the way the basic skills are composed in a complex skill is included in its model. Therefore, whenever the representation for a new skill is created, the representation of the composition is automatically attached. A skill needs only to be defined once, and from then on it can be used in different coalitions. For each complex skill that needs to be included in CoBASA two things are necessary: a rule that describes the way the skill can be generated, and the model that represents it. The knowledge base attached to each CA includes skill models, and the rules defined to generate them. When a coalition is created the CA automatically generates the complex skills based on the rules its knowledge base contains.

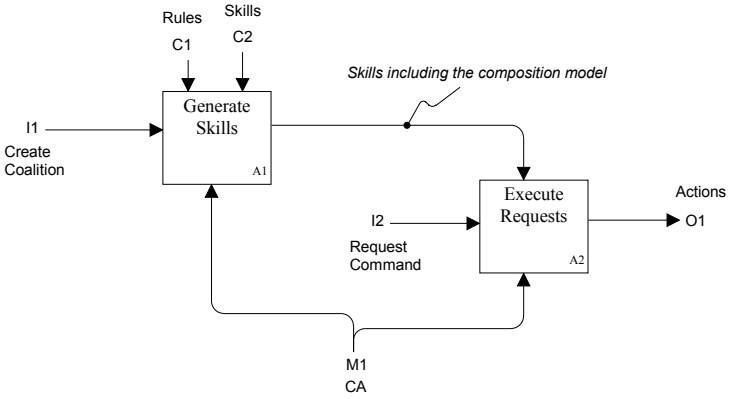

Figure 4-27 – Relationship between the generation of skills and their execution

Figure 4-27 illustrates the mechanism that has been described. The generate skills and execute requests activities are done in the CA. When a coalition is created the CA generates complex skills based on the rules and skills that have been created before. Whenever a request is received by the CA asking for one of its generated skills the model associated with the complex skill being requested is used to control the execution of the CA's interpreter. This is the main mechanism to avoid codification, since whenever a coalition is created it is not necessary to reprogram the CA.

To illustrate the ideas that have been introduced, lets consider that the executing machine is a Prolog engine. Consequently a Prolog program is defined for each complex skill.

Lets suppose a coalition consisting of one robot, one tool exchange mechanism, two grippers, and one tool storage system is created (Figure 4-28). Some of the skills of each of the coalition members are shown in Figure 4-28. The robot only shows a generic "move" skill (*moveSkill*) even if it usually

offers "moves" using world coordinates, or joint coordinates. Nevertheless, it was decided not to show them for better figure reading. The grippers show the normal open and close skills (*openSkill* & *closeSkill*), while the tool exchange system shows the skills *attachSkill* (used to attach the tool) and *releaseSkill* (used to release the tool). Finally, the tool storage shows the skills *getToolSkill* and *setToolSkill*. The first one is used to find out where a certain tool is. When activated it responds by giving the tool's position in coordinates. The second one is used to get a free available position in the tool storage. When activated it responds with the coordinates of the free position.

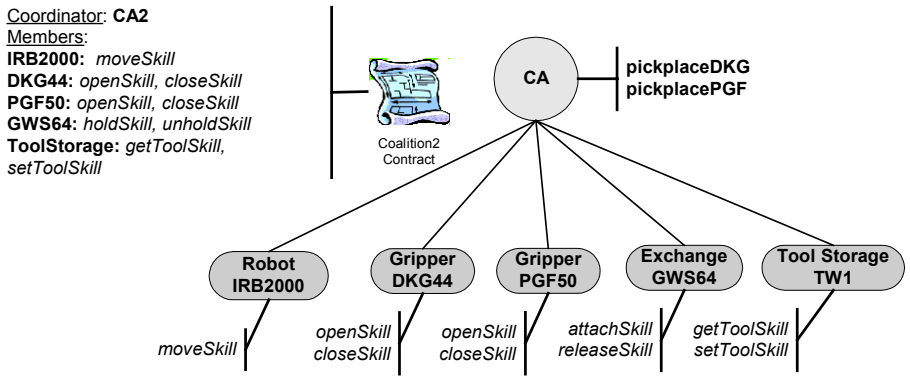

Figure 4-28 – A manufacturing coalition and its basic skills

As it was mentioned before, the CA executes the rules that generate complex skills. As an example the Prolog rules to generate the complex skill *pickplace* are shown below:

```
pickplace(Robot, Gripper)  :-  skill(moveSkill,Robot,_), skill(openSkill,Gripper,ToolStorage),
                               skill(closeSkill,Gripper,ToolStorage), skill(attachSkill,Exchanger,Robot),
                               skill(releaseSkilll,Exchanger,Robot), skill(getToolSkill,ToolStorage,_),
                               skill(setToolSkill,ToolStorage,_).
    pickplace(Robot, Gripper) :-  skill(moveSkill,Robot,_), skill(openSkill,Gripper,Robot), skill(closeSkill,Gripper,Robot).
```

These rules are based on the predicates **skill(<skillName>,<agent>,<location>)**. The CA, based on the coalition contract, creates these predicates. In the specific case of Figure 4-28 the following facts are created:

skill(moveSkill,"IRB2000",_)

skill(openSkill,"DKG44","TW1")

skill(closeSkill,"DKG44","TW1")

skill(attachSkill,"GWS64","IRB2000")

skill(getToolSkill,"TW1",_)

skill(openSkill,"PGF50","TW1")

skill(closeSkill,"PGF50","TW1")

skill(releaseSkill,"GWS64","IRB2000")

skill(setToolSkill,"TW1",_)

The location attribute is used to identify the places where grippers are stored. In the case of the tool exchange mechanism the location is the robot where the mechanism is mounted.

In the indicated coalition two *pickplace* skills are generated because two grippers exist in the CA local knowledge base after the execution of the generate skills task. In other words, the complex skills *pickplace("IRB2000","DKG44")* and *pickplace("IRB2000","GWS64")* are obtained because the CA of the illustrated coalition in Figure 4-28 contains all the necessary skills to satisfy the *pickplace* rule. The rules only show the conditions required to generate the complex skill. The process of creating the representation of the skill is not shown. However, it is during this process that the program (model) of the complex skill is generated and put in the representation of the new complex skill (Figure 4-27). As an example, the program generated by the rule for the *pickplace("IRB2000","DKG44")* skill is:

```
/* The robot is holding the tool it needs: DKG44*/
run :- executeSkill(holding,"IRB2000","DKG44", TRUE), executeSkill(openSkill,"DKG44",_,_),
       executeSkill(moveSkill,"IRB2000",OrigCoord,_), executeSkill(closeSkill,"DKG44",_,_),
       executeSkill(moveSkill,"IRB2000",DstCoord,_), executeSkill(openSkill,"DKG44",_,_).
/*The robot is not holding a tool*/
run :-  executeSkill(holding,"IRB2000",_, nil), executeSkill(getToolSkill,"TW1","DKG44", TCoord),
        executeSkill(moveSkill,"IRB2000", TCoord, _), executeSkill(attachSkill,"GWS64", _, _),
        executeSkill(setHolding,"IRB2000", "DKG44", _), executeSkill(openSkill,"DKG44",_,_),
        executeSkill(moveSkill,"IRB2000",OrigCoord,_), executeSkill(closeSkill,"DKG44",_,_),
        executeSkill(moveSkill,"IRB2000",DstCoord,_), executeSkill(openSkill,"DKG44",_,_).
/* The robot is holding another tool */
run :- executeSkill(holding,"IRB2000",_, HoldedTool),  executeSkill(getToolSkill,"TW1","DKG44", TCoord),
       executeSkill(setToolSkill,"TW1", HoldedTool, NTCoord), executeSkill(moveSkill,"IRB2000", NTCoord, _),
       executeSkill(releaseSkill,"GWS64", _, _), executeSkill(moveSkill,"IRB2000", TCoord, _),
       executeSkill(attachSkill,"GWS64", _, _), executeSkill(setHolding,"IRB2000", "DKG44", _),
       executeSkill(openSkill,"DKG44",_,_), executeSkill(moveSkill,"IRB2000",OrigCoord,_),
       executeSkill(closeSkill,"DKG44",_,_), executeSkill(moveSkill,"IRB2000",DstCoord,_),
       executeSkill(openSkill,"DKG44",_,_).
```

The template for the executeSkill predicate is **executeSkill(<skill name>,<executor agent>, <input arg>, <output arg>)**. This predicate is able to request the <*executor agent*> for the <*skill name*>, and the input argument of that request is <*input arg*> while the result is obtained in <*output arg*>.

The *pickplace* skill execution model is complex because the robot may exchange tools, and thus the execution model is not only a sequence of moves, opens and closes. This was chosen on purpose to emphasise that even with skills with complex execution models the approach is still valid.

The central point of this proposal is to have all rules that create skills defined at the time the coalition is being created (configuration phase) because then, at execution time (when the coalition is being operated), each offered skill that is requested has an execution representation that can be easily executed. Every time a new skill needs to be added to the CoBASA architecture new rules need to be created, which should be able to generate the model representing the structure of the complex skill. When a coalition is created the user works with a set of predefined composed skills, which are all the

ones known at that time for the specific manufacturing domain under consideration. The skills' representation, as well as the rules used to generate them, are included in the knowledge base of the CoBASA architecture. If, for some reason, a new skill is really needed then the knowledge base must be updated with the new skill representation as well as its execution model.

To conclude, it is possible to generate different structures of manufacturing components (coalitions), which can show different behaviour without programming effort since they were able to generate their own execution models from a set of basic operations (skills).

4.2.3.3 Multilateral Coalitions Contracts - MCC

The behaviour of the coalition is regulated by **a multilateral coalition contract (MCC)** that is "signed" by all members of the coalition. The important terms of this type of contract other than the usual ones like duration, names of the members, penalties, etc., are the consideration and the individual skills that each member brings to the coalition. Note that the skills involved in a specific consortium contract may be a subset of the skills offered by the involved agents when they join the cluster.

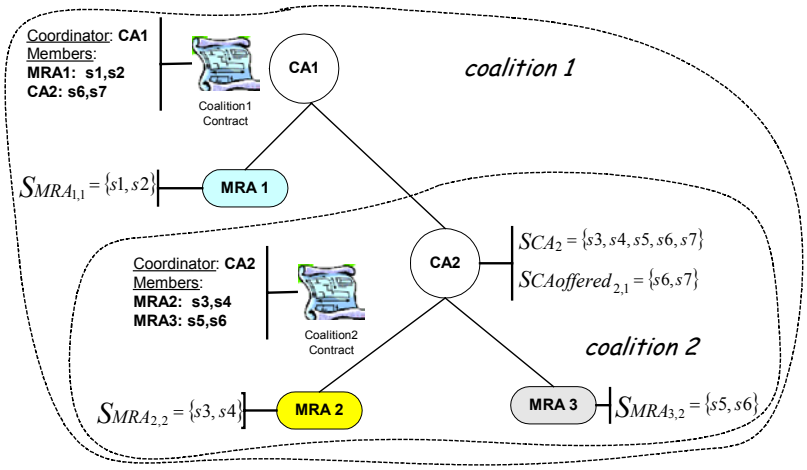

Figure 4-29 – Coalitions contracts

Figure 4-29 shows a hierarchy of two coalitions in which CA2 is simultaneously the coordinator of *coalition 2* and a member of *coalition 1* led by CA1. As it could be expected there are two multilateral consortium contracts, one for each coalition. However, each member of a coalition must have a copy of the contract that regulates the coalition's operations, since the members' behaviour is regulated by that contract. This means that in the case of Figure 4-29 the behaviour of CA2 is conditioned, in fact, by two contracts instead of one: 1) the contract of *coalition 1*, where CA2 is a member, and 2) the contract of *coalition 2*, where CA2 is the coordinator. To distinguish between these two types of roles, the MCC contracts each CA might be bound to are divided into **membership contracts** and **coordination contracts**. All MCC contracts in which the agent plays the member role are membership contracts while those in which it plays the coordinator role are coordination ones. Despite this

division, the structure of the contracts is the same, since both types are multilateral consortium contract - MCC.

Modelling. The MCC model is similar to the Cluster Adhesion Contract (CAC), described in section 4.2.2.3, and since the parts that compose it are the same they will not be described again. Nevertheless, there are some differences in the model that result mainly from the multilateral nature of the MCC, while CAC is a bilateral contract. A model for the MCC, using UML is shown in Figure 4-30.

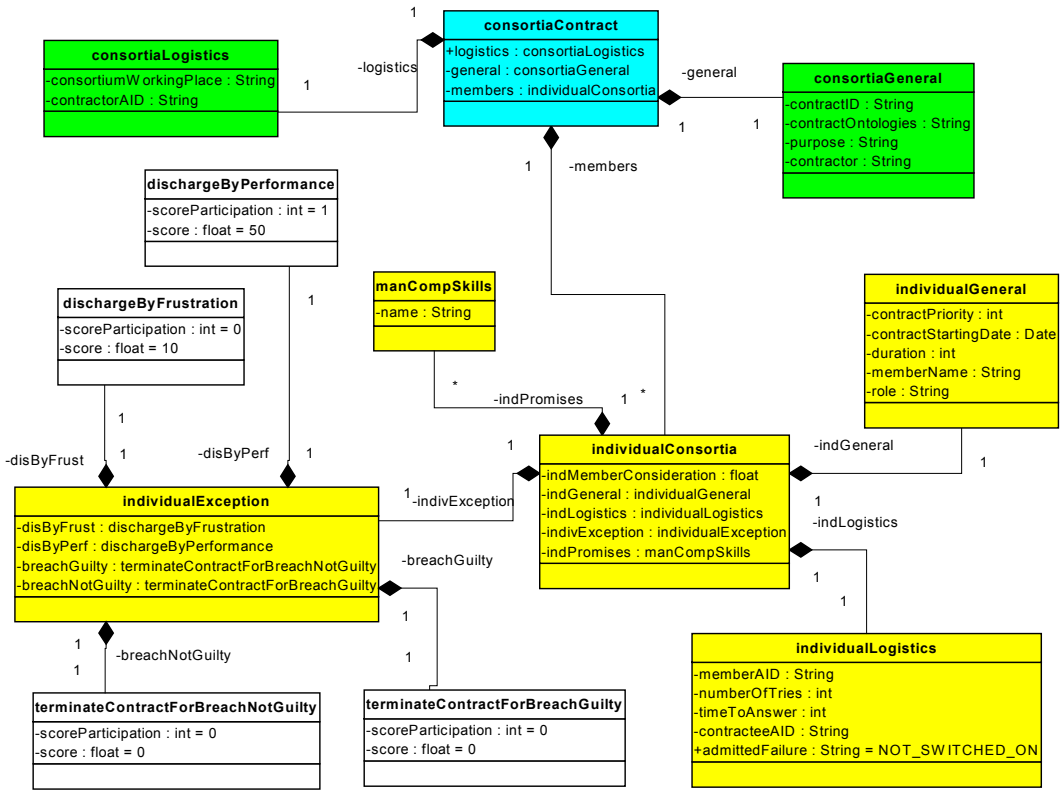

Figure 4-30 – The Multilateral Consortium Contract (MCC)

To support various members and one contractor, the MCC has one common part dedicated to the contractor (the agent playing the coordination role), and another part dedicated to each one of the other members. The *members'* attribute in *consortiaContract* is composed of several *individualConsortia* elements that, in turn, describe the individual contractual terms of each member of the coalition.

The *consortiaGeneral* and *consortiaLogistics* parts are similar to the adhesion contract, but applied to the coordinator. The *consortiumWorkingPlace* has the same role as before but now applied at the consortium level. The name of the CA leading the coalition is described in the attribute *contractor* and its agent address in *contractorAID*. The contract is uniquely identified by the attribute *contractID*.

The individual part of the contract is composed of the promise (declaration or manifestation of an intention in a contract) or set of skills each agent brings in to the coalition, logistics and general aspects related to the specific individual member, and the reward or penalisation that each member

receives at the end of the coalition. The *manCompSkills* type describes the set of skills each member brings in and the attribute that contains them is *indPromises*. The Logistic and general parts of the individual members are similar to the global part describing the coordinator as well as in the case of the cluster contract. However, the *admittedFailure* attribute is only used at the MCC. It indicates the set of failures that the member is allowed to report when asked to perform one of its promised skills. Whenever the member replies, giving a different reason for not being able to perform the operation, the CA can consider this as a violation of the contract. Further details will be provided when the MCC life cycle is discussed.

At the end of the contract the CA awards each member a number that represents the quality of the handed out service. This award or penalisation, if added to the agent's credits, can be used to improve (or reduce) its qualification, and is important for the future participation of the agent in coalitions. This mechanism is similar to the one discussed when CACs were discussed. However, there are some differences that are indicated in the next section. As before, a contract can be terminated by performance, breach, or frustration.

MCC life cycle. As in the case of the CAC, the MCC life cycle phases include creation, performance, and termination. The contract is created by the broker agent (BA) when a coalition is created. The user configures the different parts of the contract based on the requirements required by the coalition. For each member the individual part is fulfilled by choosing which skills the members bring into the coalition. Further details will be given when discussing the coalition life cycle.

The performance of the MCC includes the execution of the contract promises (skills). This is done while the contract is still valid and the coalition is operating. Only promised skills can be requested.

When a member leaves the coalition it does not mean that the coalition will be dissolved. Only the part referring to the leaving member is changed. However, from the point of view of the leaving member its contract terminated.

The "good" way of terminating a contract is by performance. In this situation, the CA (coordinator) verifies if the participation of the other members is still date valid. If not, the CA asks that member to terminate its participation. Based on the value stored in *dischargeByPerformance*, in the individual exception part of the MCC, the award for the participation in the coalition is collected.

Terminating the MCC by a frustration reason is abnormal and, consequently, the breaking agent may incur some penalisations. The request to break the contract by frustration is always initiated by the coalition member that detects the reason for frustration. When this happens, it collects the penalisation stored in *dischargeByFrustration*. Three reasons are considered to claim the termination of a contract for frustration reasons:

1. The user requests the agent (MRA/CA) to leave (physical move, for instance).
2. A CA participating in another coalition detects their members are not responding.
3. A CA/MRA of a lower level could not renegotiate a contract change with its higher level CA.

Reason 1 occurs when, for instance, the physical component needs to be moved from the place where it is currently placed (Figure 4-16). In this situation a user can exploit the MRA's GUI to demand that the agent break up all its coalitions contracts. The CA then sends a REQUEST command to the CA leader to terminate the contract for frustration reasons. After its response, the MRA asks the CMgA to terminate its cluster contract as was mentioned in section 4.2.2.4.

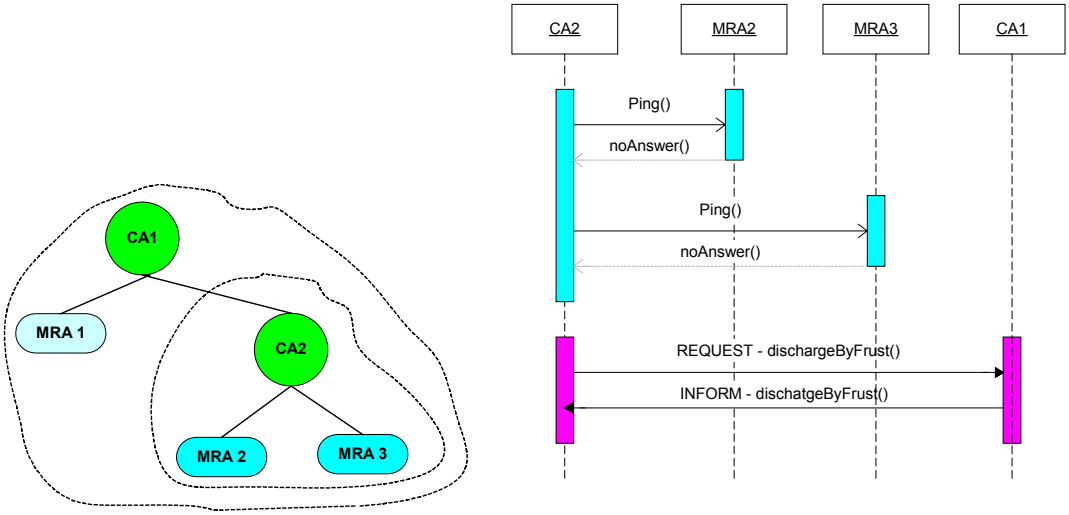

Figure 4-31 – MCC terminated by frustration triggered by lack of members

Reason 2 occurs when a CA suddenly finds out it does not have members in its coalition (Figure 4-31). To guarantee that the members of a coalition are alive, the CA of that coalition sends, from time to time, a ping command. If none of the members replies, this means the CA cannot accomplish its duty and so if it is participating as a member of another coalition it requests the leader of that coalition to terminate its contract for frustration reasons. Considering the hierarchy of coalitions depicted in Figure 4-31, whenever CA2 does not receive answers from MRA2 and MRA3 to its ping command, it sends a request to CA1 to terminate its participation in the coalition 1. In this case, for instance, only MRA2 did not reply, CA2 renegotiates new terms with CA1 for its participation. If CA1 does not accept the new terms then CA2 can terminate the contract claiming frustration.

Reason 3 occurs whenever a coalition member (MRA) that wants to renegotiate the contract promise (skills) with its coordinator receives a FAILURE reply, which means the coordinator has not accepted the renegotiation. Since the member cannot fulfil its promises anymore, it requests its coordinator to terminate the contract for frustration reasons. The reasons that lead the member to renegotiate the contract terms can be various. The most common is a change in its skills that can happen either due to a physical change, or removal from one of the coalitions it might be participating in. The termination of the contract by frustration when an agent cannot renegotiate its contract terms can be ambiguous because, at first sight, it seems that the coordinator should have the right to breach the contract (worst penalisation to the member) rather than the member. However, if it is considered that the member is doing that because it cannot achieve what was promised, not because of an

intentional fault, but because of something the agent could not control, then a termination by frustration is justifiable. Figure 4-32 shows this process of termination. If *MRA3* is asked to be removed using its GUI, it asks *CA2* to terminate its participation in *coalition2* claiming frustration. After the acceptance of *CA2* it generates the new skills and then verifies if that removal has an impact on the skills promised to *coalition1*. If yes, it asks *CA* to renegotiate by sending a REQUEST command. If *CA1* does not accept (FAILURE command), CA2 then asks to terminate the participation of *coalition2* in *coalition1*.

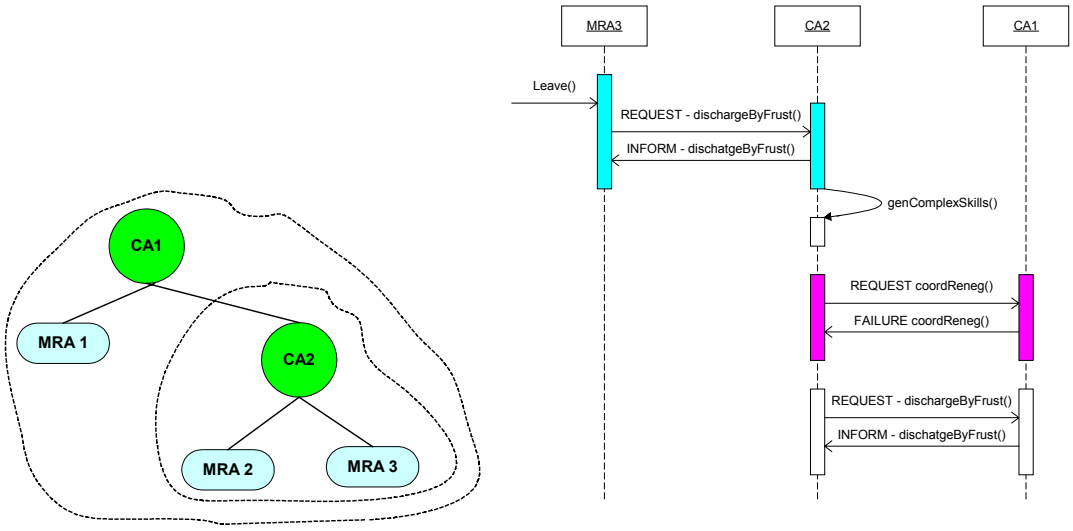

Figure 4-32 – MCC terminated by frustration triggered by lower level CA

Terminating by breach is the worst case of termination of a contract from the penalisations point of view. The request to breach the MCC can be started either by the coordinator or by one of the members. A breach started by the coordinator implies that one of the members misbehaved. On the other hand, a breach started by one of the members means coordinator misbehaviour. A member starting a breach does not incur penalties. In effect it takes its reward from the contract part *terminateContractForBreachNotGuilty*. However, when it is "guilty", i.e., the coordinator detected some misbehaviour, it gets its penalisation from *terminateContractForBreachGuilty*. A member shows bad behaviour whenever it does not answer a request from its coordinator to execute one of the promised skills (Figure 4-33 using the hierarchy of coalitions shown in Figure 4-32). The same happens if the member, in spite of replying to the request, is not able to perform it properly, i.e., the excuse for the failure is not included in the MCC (attribute *admittedFailure*). A coordinator, on the other hand, shows bad behaviour whenever it does not answer a request from the member, which can be, for instance, a call to renegotiate the contract terms.

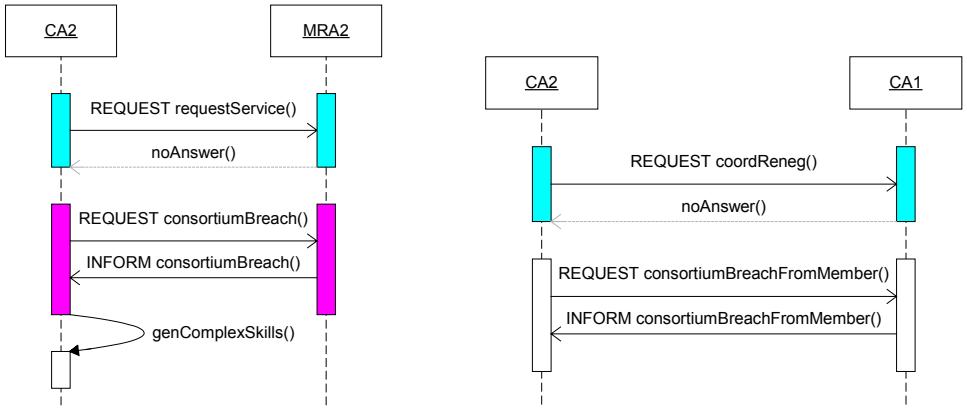

Figure 4-33 – Breaches in coalitions

4.2.4 Coalitions Life Cycle

The phases in the life cycle of a coalition are important because they are intrinsically related to the agility of the production system in the way new manufacturing components (MRAs) can be added and removed to build different control/supervision structures on top of the physical structure (the cell, which is represented by the cluster). Since CoBASA is an architecture that supports part of the life cycle of a production system, the relationship between the phases of this life cycle and the phases of the coalition life cycle are shown. This is done in order to facilitate the task of the reader in understanding how CoBASA, or more specifically, its coalitions support some of the phases of the production system life cycle.

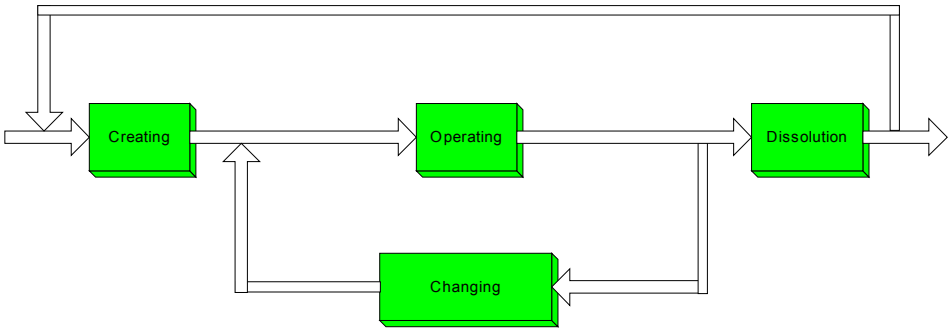

Figure 4-34 – Coalition life cycle phases

The life cycle phases of a coalition (Figure 4-34) consist of the phases creating, operating, changing, and dissolution. The creation phase occurs when a coalition is created for the first time. The changing phase occurs when some items from the cluster are added or removed from the coalition. The dissolution occurs when all the elements are removed, including the CA.

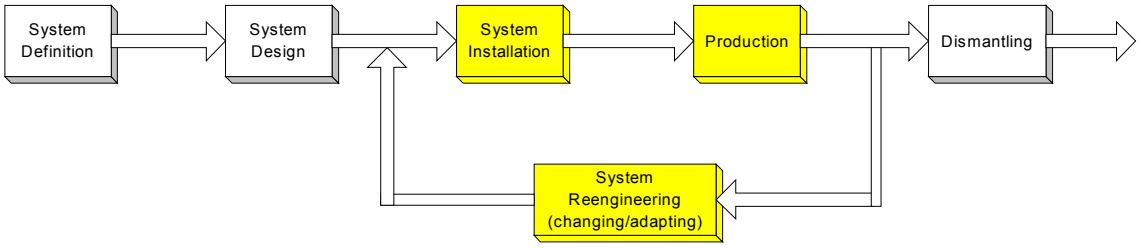

Figure 4-35 – Production system life cycle phases

Figure 4-35 shows one proposal for the life cycle phases of a production system. It should be noted, however, that other classifications exist. Although, this work does not intend to discuss thoroughly the structure or even the names of a production system life cycle, the introduction of a specific phase to characterise the period in which the system is changed/adapted is a direct result of the work developed in the framework of this thesis. The important thing is to note that a system is created, installed, operated, and changed/adapted (reengineered).

The relationship between the phases of the production system life cycle and the coalition life cycle is shown in Table 4-1. It shows that CoBASA coalitions support the phases *System Reengineering*, *System Installation*, and *Production*. Although the phases *System Design* and *Dismantling* are also supported, it needs the help of the cluster manager.

Coalition Life Cycle Phase	Supported Production System Life Cycle Phase
Creating	1. *System Design* 2. System Reengineering
Operating	1. Production
Changing	1. System Reengineering 2. System Installation
Dissolution	1. *Dismantling* 2. System Reengineering

Table 4-1 - Relationship between production system and coalition life cycle phases

Coalitions are created, changed, or dissolved whenever the system is modified or needs to be operated in a different way. During the phase in which the system is operating the active coalitions are in their operating phase. The dismantling of a production system involves the termination of all active coalitions as well as the termination of the cluster manager. Please note that the termination of the cluster manager does not involve the death of the MRA agents that composed it. Coalitions can be created both in the *System Design* and *System Reengineering* phases. When the system is created, the *System Design* phase is used to create the coalitions. Future changes to the system are made in the *System Reengineering* phase. Existing coalitions are changed in both *System Reengineering* and

System Installation phases. This happens because during the *System Installation* phase it might be necessary to change coalitions that have been created during the reengineering because of last minute changes.

4.2.4.1 Creation

The main actor in creating coalitions is the broker agent (BA). A human user chooses the coalitions based on the logical structure he/she wants to create. The other important actor is the cluster manager agent (CMgA) that provides information about available members. In addition to these two agents others are needed to create a coalition:

1. A CA not currently engaged in any consortium (available to lead a coalition).
2. MRAs, if the coalition will include manufacturing components.
3. CAs leading coalitions that might be included as members of the coalition being created.

Figure 4-36 shows the interactions that happen between the different actors involved in creating a coalition. The figure shows the BA agent, the CA agent that has been chosen to be the coordinator, an agent to represent the members of the coalition (*MRA/CA(member)*), and the cluster manager agent (CMgA). Independently of the type of MRAs or CAs that form the coalition, the behaviour is the one indicated in the figure. All information exchanged between the various actors is shown using the FIPA REQUEST protocol (FIPA, 2001).

The broker asks for information about candidate members in the cluster by sending a REQUEST command. After getting the information from the cluster manager (CMgA), the broker shows the available members to the user as well as their individual information and lets him/her compose the coalition and create the contract that regulates it. The broker then asks each member to verify if they accept the contract, what is done by sending a REQUEST *to be member* command. This step is done in order to make sure each individual agent evaluates the contract before accepting it. This corresponds to asking the agent if it is interested in participating in the coalition under those conditions.

After all candidate members, including the coordinator, have expressed their interest in participating in the coalition, the broker starts the process of signing the contract by sending a REQUEST *to sign* command. Signing does not involve a complex formalism because the objective is to indicate to coalition members that the contract is now effective. After the broker requests that the coordinator signs the contract, the coalition is now operating from its point of view. After signing the contract the CA must try to generate its complex skills (*genComplexSkills*) as it has just received a new set of skills from its members. This step is crucial for the agility of the system, because the coalition is now generating automatically its skills based on the skills brought in by the members that compose it. When the skills are generated the new coalition leader can then ask the CMgA to update

its skills and to change its status from free to coordinate to coalition leader. The coalition is now registered in the cluster manager through its leader.

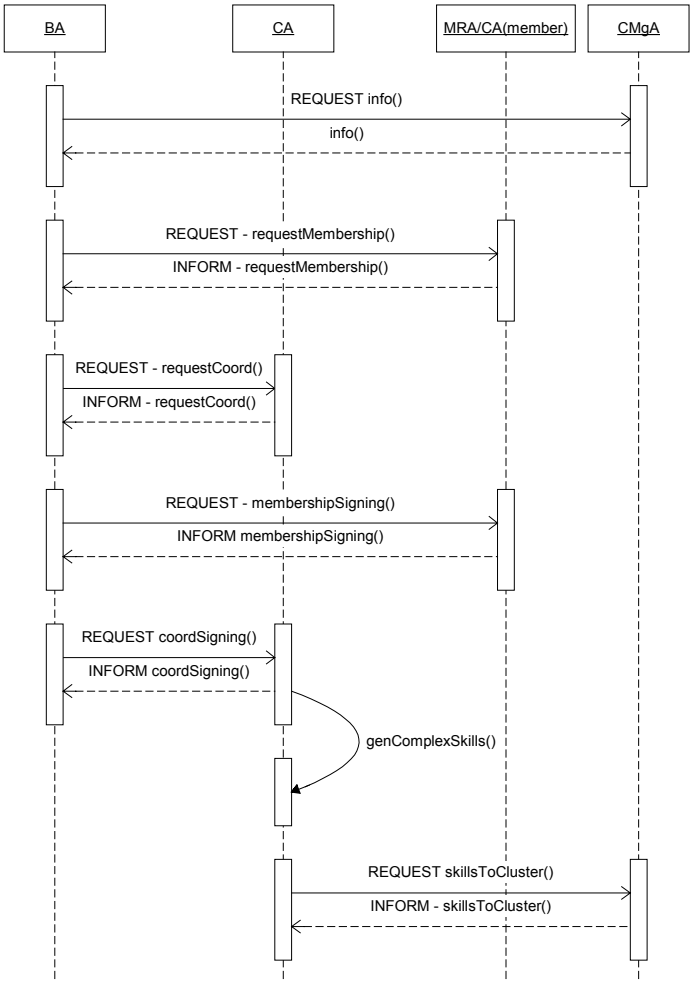

Figure 4-36 – Interactions when creating a coalition

4.2.4.2 Operation

This phase corresponds to the production phase of the production system life cycle, since operating a coalition is asking its members to execute skills (or commands) they have promised in the MCC that regulates that coalition. In addition, asking to perform a skill involves, ultimately, executing some commands in the manufacturing physical component connected to one of the MRAs that belongs to the hierarchy of coalitions. It must be recalled that MRAs are always the lower level participants of any hierarchy of coalitions.

While in section 4.2.3.2 it was discussed what skills are and how they are composed as well as a proposal for a generic executing machine, this section focuses on the interactions that happen among coalition members when they are executing skills. Figure 4-37 shows the execution of skills in the hierarchy of coalitions shown on the left part of the figure. It is considered that CA1 requests the

complex skill *s7*, which is offered to *coalition 1* by *coalition 2*. Furthermore, *s7* is composed of *s4* and *s6*, offered to *coalition 2* by *MRA 2* and *MRA 3* respectively. When CA1 needs to execute its skill *s7*, due to, for instance, a higher-level request, the agent finds out in the coalition's MCC that CA2 offered that skill. Then CA1 sends a REQUEST *service* command asking CA2 to execute skill *s7*, since it is offered by *coalition 2*. When CA2 receives the request it validates its origin by looking in the various contracts stored in **membership contracts** to whose coalition or coalitions this skill had been offered to. Next, the leaders in the set of membership contracts in which the skill is offered are checked to validate the request. After this validation, the CA1 decomposes the requested skill into its basic components (*s4* and *s6*), and, then, after verifying which agents offered them, starts sending the requests according to the complex skill structure. When MRA 2 finishes the execution of *s4* it replies to CA2 with an INFORM *service* command or a FAILURE *service* command (not shown in the figure), depending on, respectively, if the request was successfully accomplished, or not. After receiving the INFORM message for the first request (*s4*) CA2 sends the REQUEST service command to MRA 3 asking for *s6* in a way similar to *s4*. After CA2 receives the s6 INFORM message from MRA 3, it sends an INFORM service command to CA1 informing that its request for *s7* has been successfully achieved.

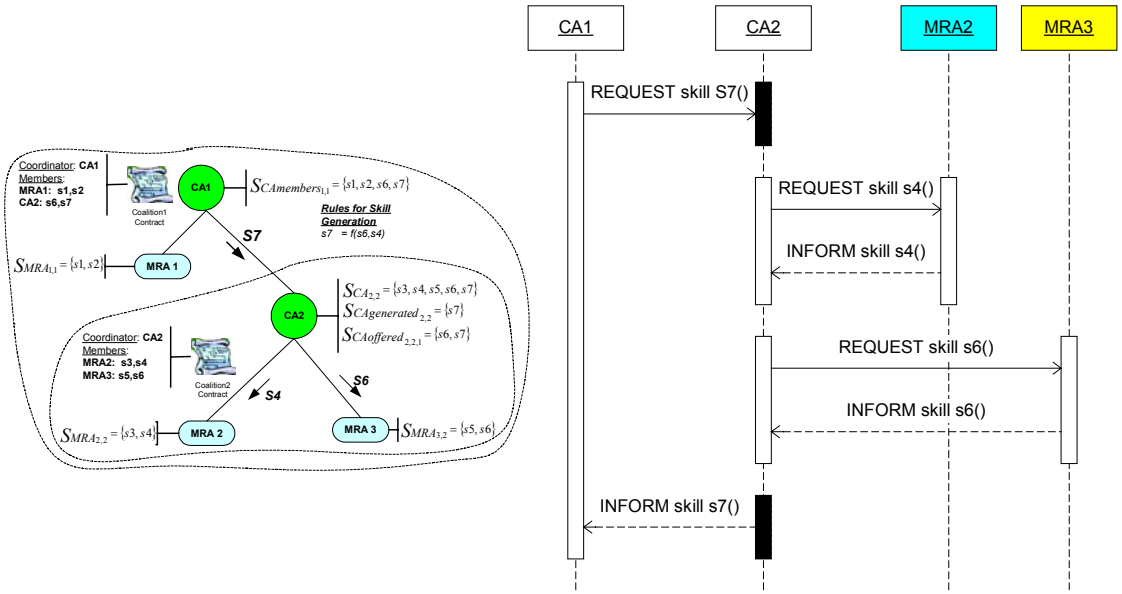

Figure 4-37 – Skills requests in a hierarchy of coalitions

Although abnormal execution situations are not shown, it is important to know when they happen:

1. An agent does not answer a valid request addressed to it from its coalition leader.
2. An agent refuses to execute a valid request from its coalition leader.
3. A request command was not successfully accomplished.

In the first and second situations the agent is immediately expelled from the coalition. The coordinator does this by asking the faulty agent to breach its coalition contract. Although this extreme situation rarely happens, it is considered to be showing agents that the act of refusing something promised on a contract has serious consequences. Eventually the faulty agent asks for user attention after such a situation happens. The third abnormal situation is when the agent who was asked to execute an offered skill replies with a FAILURE message, which denotes that for some reason the agent could not successfully execute the command. The reason is indicated in the message content. Whenever the coalition leader (CA) receives such a message, it first verifies the reason and then decides accordingly. If the reason is acceptable, the CA tries to find an alternative solution using an error recovery strategy. If the reason is not acceptable the error is so serious that it needs the attention of a user.

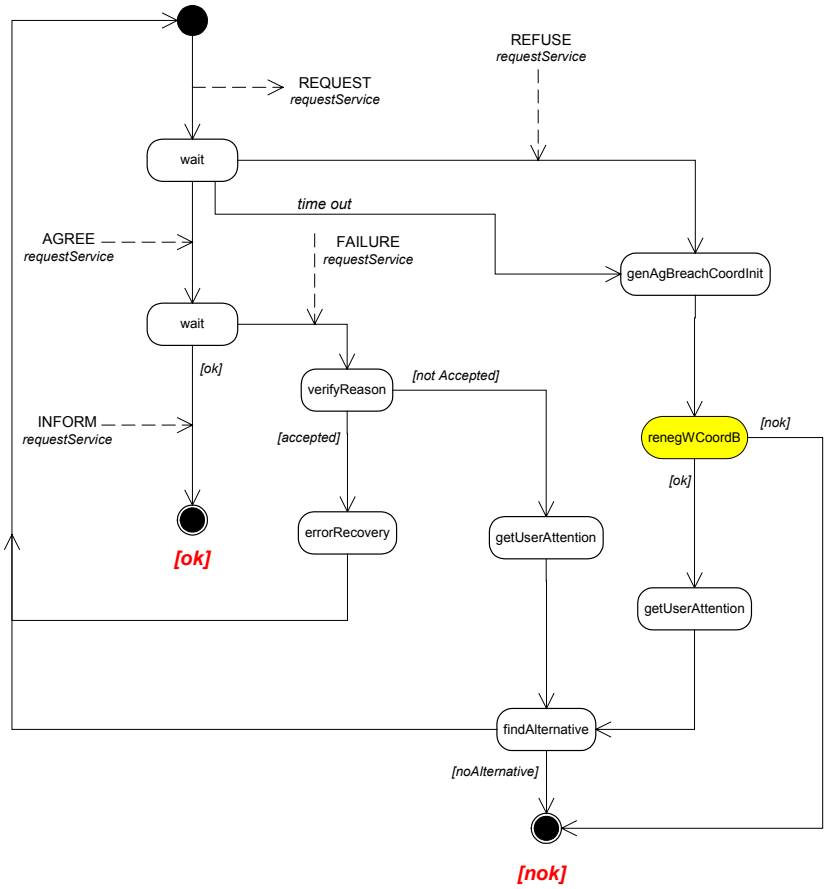

Figure 4-38 – The activity diagram for the execution of the CA agent

Figure 4-38 shows the process that has been described, using AUML (AUML, 2002), which is no more than an adapted UML activity diagram. The arrows leaving the diagram denote messages sent by the CA, while the arrows entering the diagram are the answers to the sent messages. It can be seen that CA starts by sending a REQUEST *requestService*, and then waits for the AGREE or REFUSE message in accordance with the FIPA REQUEST protocol. After the AGREE message, which indicates that the agent is committed to perform the request, the CA receives an INFORM

requestService to denote that the command has been successfully performed. Contract renegotiation (*renegWCoordB*) is done with the coalitions in which the CA participates because after a breach (*genAgBreachCoordInit*) the coalition coordinator (CA) might have its set of available skills reduced.

4.2.4.3 Changing

Changing a coalition corresponds to changing the way the manufacturing components are organised, i.e. changing the system's logical control structure, making this phase directly connected to the reengineering phase of the production system. This phase is divided into two different parts: the first one discusses the addition of one member to an existing coalition, and the other discusses the removal of one element. Although the description is made for one element to simplify the diagrams, the addition/removal of several elements is straightforward.

The interactions involved when a new member is added to an existing coalition are shown in Figure 4-39. As in the previous case, the broker and the cluster manager are important players because it is through the broker that the coalition is altered while the CMgA provides the necessary information. Furthermore, the coalition coordinator (CA) and its members (consMemb), the member to be added (newMember), and the coordinators of the coalitions (CA+1, CA+2), where hypothetically the coalition being changed is participating in, are the other actors.

The process starts with the BA asking the CMgA to provide information about its members. Hence, the user, via the broker, selects the coalition to be changed which provokes the BA to ask the coordinator of that coalition to send it its MCC (REQUEST *getContract*). This contract is needed because the user needs to configure its individual part with data from the new member as well as possibly changing other parts. After changing the contract, the new member is asked to accept the contract and to sign it. These operations are similar to the ones introduced in the creation phase. The broker now needs to renegotiate the new terms of the contract with the other coalition members to let these members discuss it (REQUEST *membershipReneg*). Under normal circumstances these agents accept the changed contract. What happens if one or more members refuses to participate is not shown to keep the figure simpler. In any case, when in this situation, the user through the broker or through the member's GUI has the authority to overcome this situation. The broker then proceeds to the renegotiation phase with the coalition leader (CA). The goal of this phase is to get the new contract version accepted by the CA. This is why this process is called a renegotiation (REQUEST *coordReneg*). When the broker receives the INFORM stating that the contract was accepted the process is finished from the broker point of view. However, the CA has some other tasks to do before the whole process is concluded. First, it needs to check if the addition of the new element has generated new skills, which is done by activating *genComplexSkills*. Next, the CA checks if it is currently engaged in any other coalition as well as if it has got new skills. If yes in both cases, it renegotiates with the leader (CA+1) of that coalition to change the skills it is bringing in (REQUEST

coordReneg). Finally, after the successful renegotiation, the CA updates the skills of the coalition in the cluster manager (REQUEST *updateSkills*).

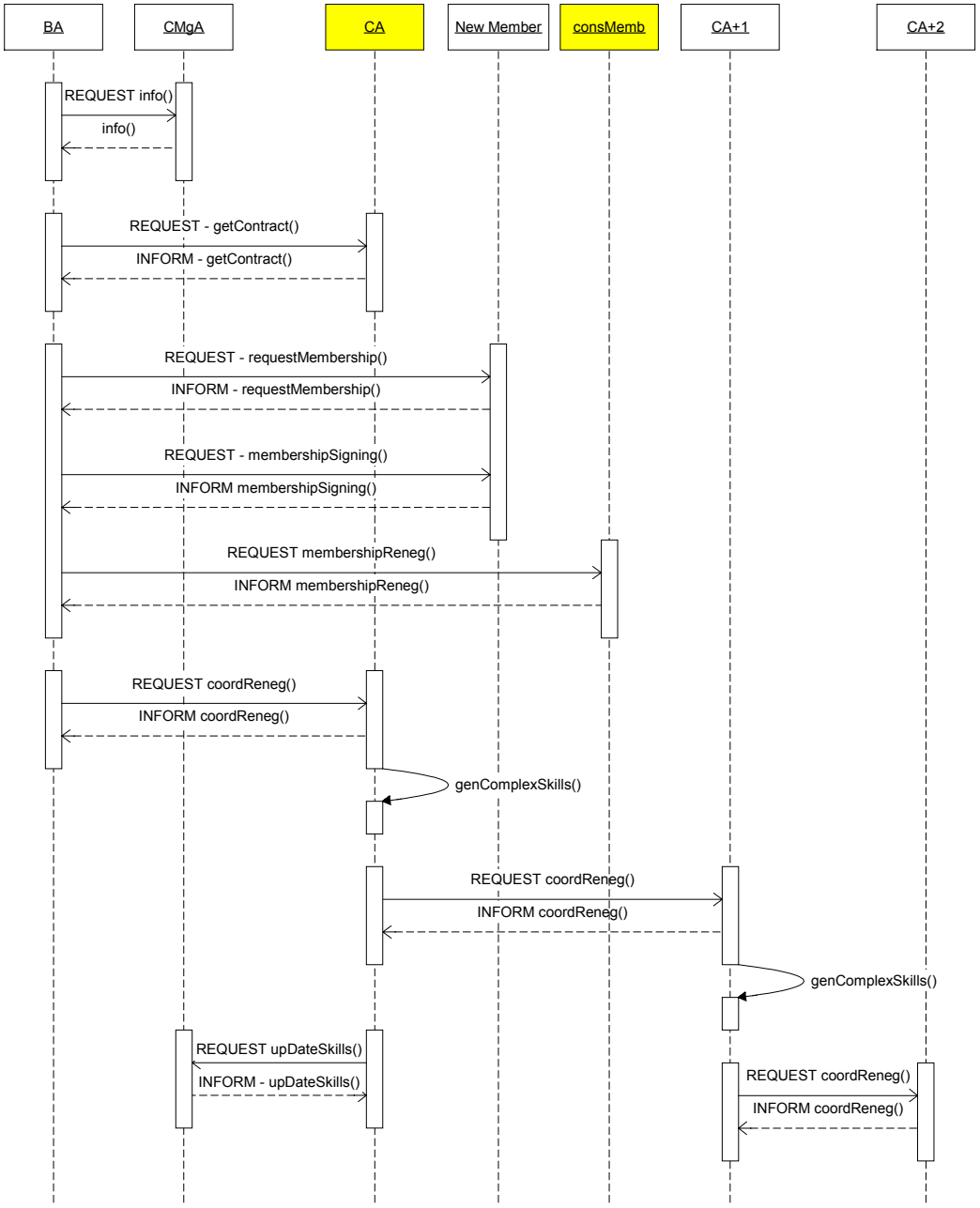

Figure 4-39 – Adding an element to an existing coalition

Figure 4-39 also shows that if the renegotiation between the CA and CA+1 has impact on CA+1's skills, and if CA+1 is also participating in another coalition led by CA+2, then it will request CA+2 to renegotiate the terms of its participation in that coalition contract. The process is repeated until it reaches the highest-level coordinator in the hierarchy of coalitions. This is a very important

mechanism because whenever a coalition is changed, the impact of this change is automatically propagated to all the coalitions that are directly and indirectly related to it (transitivity).

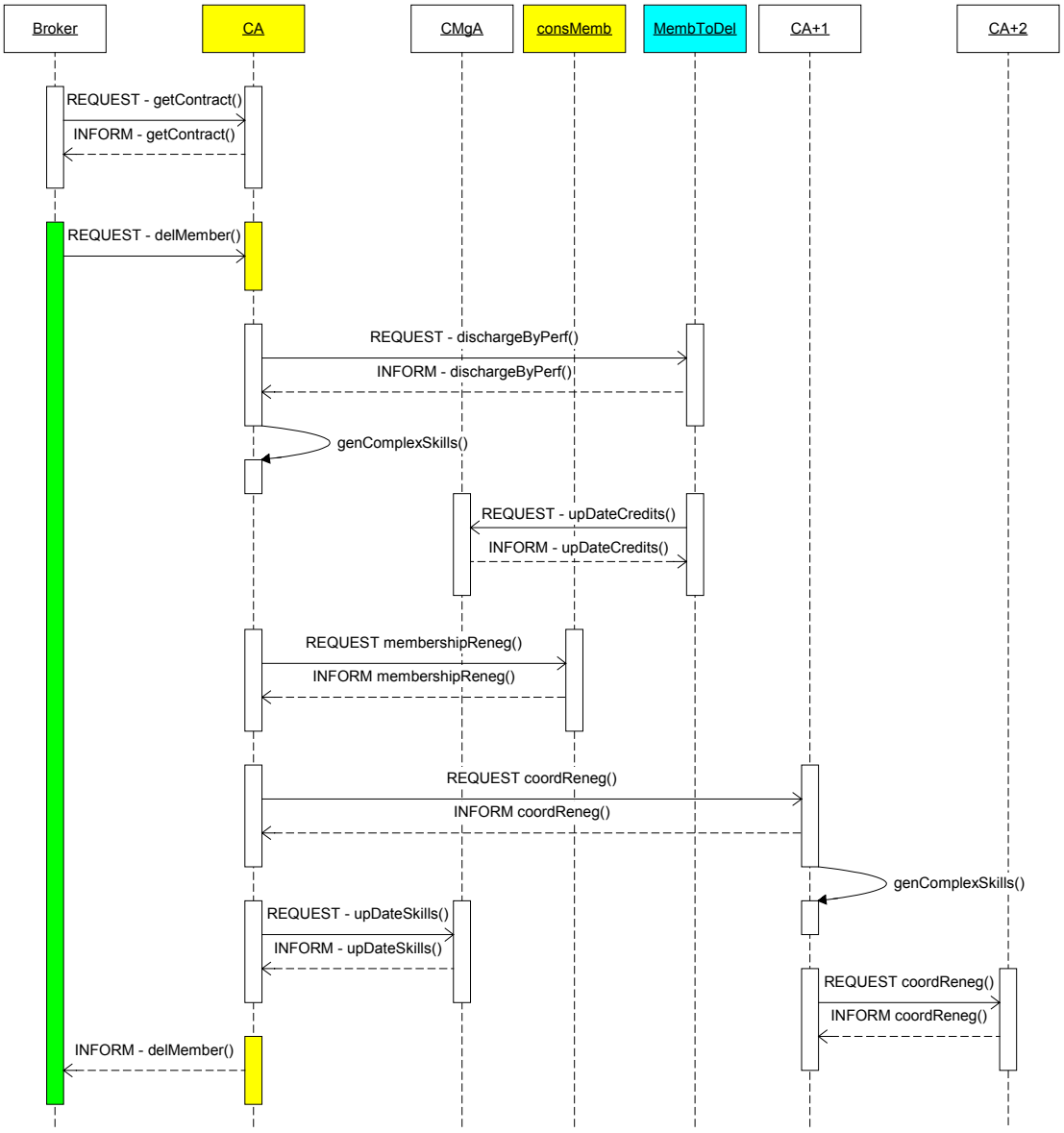

Figure 4-40 – Removing an element from an existing coalition

The interactions involved when one of the members is removed from its coalition are shown in Figure 4-40. The players in this situation are the same ones as when adding an element. Figure 4-40 shows the method of removing one coalition member. In this situation, the contract is terminated by performance, which allows the member to collect the reward for its participation. A not recommended termination occurs when the agent is forced by a user (through its GUI) to terminate its participation in the coalition. In this case the leaving member might be penalised since the contract is terminated by

frustration. Since CoBASA encourages termination by performance, when a coalition member needs to be removed it should preferably be removed using the broker agent.

The process starts when the user identifies the agent he/she wants to be removed (*MembToDel*). The BA then requests the coordinator of the coalition (CA) to delete that member (REQUEST *delmember*).

Based on that request, the CA terminates the participation of the selected member by terminating its contract by performance (REQUEST *dischargeByPerf*). After this, the CA generates its new set of skills since the leaving of one member might have affected its set of skills, and informs the remaining coalition members that a change in the contract has been made by sending them a new contract version (REQUEST *membershipReneg)*. Next, the CA follows the same process as when a new member is added, that is to say, renegotiating with the leader (CA+1) of the coalition it might be involved and updating the skills in the CMgA. Finally, if the whole process is successful, the CA replies to the initial request to delete the member (INFORM *delMember*) informing the broker that the member has been successfully removed. In the meantime, after the selected member to leave the coalition (*MembToDel*) finishes its contract termination (INFORM *dischargeByPerf*) it updates its new set of credits in the cluster which is done by issuing the REQUEST *updateCredits* command.

4.2.4.4 Dissolution

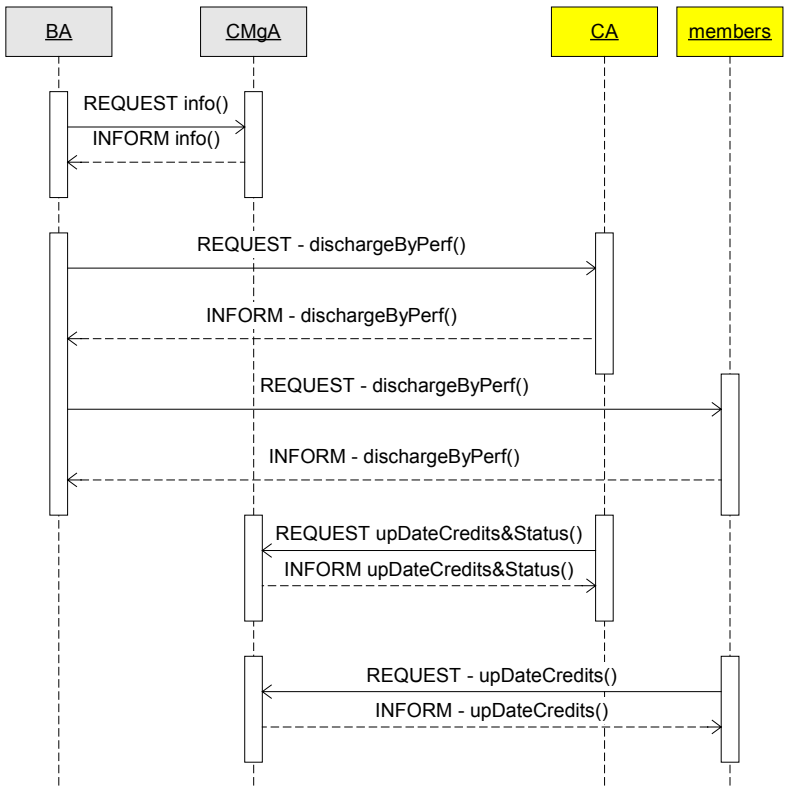

Figure 4-41 – Coalition dissolution

A coalition can be dissolved either when the system is being dismantled or when it is being reengineered. In the first case, all coalitions need to be terminated and then all cluster contracts must also be terminated. In the second case, the system is suffering such a radical change that it is not worth keeping any of the existing coalitions. Therefore all coalitions are dissolved in order to create completely new ones. Dissolving a coalition is different from changing it (removal of elements) in the way that the coalition coordinator also terminates its activity and changes its status in the cluster from coalition leader to free to coordinate.

Figure 4-41 illustrates the whole process for a coalition composed of one coordinator and one member. Since this is a convenient way of terminating, the BA discharges the MCC by performance. It first discharges the CA and then all coalition members (REQUEST *dischargeByPerf*).

After accepting the discharge, the CA updates its credits in the cluster, which have just been increased by the reward it has received, as well as its status, since the CA is now free to coordinate. Note that now the CA does not generate complex skills because it does not have any member to give it any skill. After discharging the MCC, coalition members collect their rewards and add them to their credits, and then update their credits in the CMgA (REQUEST *upDateCredits*).

4.2.5 CoBASA society

Reaching agreements. This topic is closely related to the issue of cooperation, within societies of self interested agents. CoBASA agents need to reach agreement in the following situations: 1) creating a coalition (MCC contract negotiation) (one to many), 2) adhering to a cluster (CAC contract negotiation) (one to one), 3) renegotiating MCC contracts within the coalition (one to one), 4) terminating a CAC, and 5) terminating a MCC. CoBASA agents reach agreement using different negotiation protocols, which although not very complex follow the properties defined in (Sandholm, 1999). They guarantee success in the agreement; they maximize the agent social welfare using the CoBASA credit system; they are individually rational because agents can only improve their credits by participating in coalitions and clusters; finally, they are simple.

Complex negotiation is not required because the interactions can be kept simple while maintaining the CoBASA objectives. However, these processes can be made more complex if the requirements are made more complex. An example of this situation occurs if the formation of coalitions turns out to be an automatic process, which is now done with user help. In an automatic situation the coalition could be formed using auction negotiation protocols (Sandholm, 1999), since each agent is competing against the other to participate in the coalition being created. In this situation the user would only need to specify the type of coalition that is needed.

CoBASA protocols are based on communication messages exchanged by the agents, or more specifically using communicative acts from the speech acts theory (Searle, 1969), which eventually lead to the various agent communication languages (Labrou et al., 1999). As mentioned before, FIPA

Agent Communication Language (ACL) (FIPA, 2002) is used to support the CoBASA's negotiation protocols.

Coordination. One thing is negotiating or renegotiating a coalition participation or leadership, joining a cluster, or terminating contracts, another completely different thing is how the agents should be designed to work together, after negotiations, in order to achieve CoBASA's goals. Some coordination mechanism is essential if the activities of the agents interact in any way, as is the case in CoBASA. The use of contracts in the CoBASA society imposes a kind of social law, since social laws represent simply an expected pattern of behaviour carrying with them some authority. Therefore CoBASA follows coordination by norms and social laws (Conte & Dellarocas, 2001a; Conte & Dignum, 2001). Contracts are the mechanism that imposes some behavioural constraints on agents, striking a balance between individual freedom on the one hand, and the goal of CoBASA on the other. The most important conventions or social laws that can be found in CoBASA are:

1. MRA and CA agents join clusters
2. MRA and CA agents update their credits truthfully when a coalition terminates
3. MRA and CA agents breach contracts when required
4. MRA and CA agents execute, if possible, the services that they have promised
5. MRA and CA agents accept to participate in a coalition whenever they are required to.

These conventions are enforced by the presence of MCC and CAC contracts. Some of the social laws are "hardwired", as is the case of 1, 3, and 5, while others depend on the contract terms (2,4).

Another approach to classifying coordination is described in (Papadopoulos, 2001), in which three categories are considered: 1) basic coordination infrastructure, 2) coordination frameworks, and 3) logical coordination. The basic coordination is the lower level and corresponds to the infrastructure that will be used by higher-level coordination mechanisms. Agent Communication Language (ACL) is the basic mechanism used in CoBASA. CoBASA itself does not need the middle level coordination frameworks because the multiagent development environment usually supports it. Finally, logical coordination is the level where the higher-level coordination aspects take place. The use of brokers or mediators, yellow-pages (in CoBASA the cluster manager supplies this service), local area coordinators (CAs in CoBASA) are just examples of middle-agents developed to implement coordination. The use of norms and social laws are examples of coordination mechanisms used at this level.

4.3 INDIVIDUAL COMPONENTS ARCHITECTURE

As Wooldridge (Wooldridge, 2002) and others have pointed out, the multiagent field consists of two interrelated areas. The first deals with *individual agents* (as the individual agents that compose CoBASA), while the second deals with *collections* of these agents (Omicini, Zambonelli, Klusch, &

Tolksdorf, 2001; Wooldridge, 2002), which are also known as societies. CoBASA is a society of agents, and therefore their interactions, which were discussed in the previous sections, are interactions at society level.

Architecture of the individual agents. CoBASA agents can be considered intelligent agents because they are proactive, reactive, and have social ability (Wooldridge & Jennings, 1995; Wooldridge, 2002). They are proactive since they have goal directed behaviour that is seen when MRAs and CAs participate in coalitions with the best possible performance. Moreover, they can keep working even under environmental changes. A robot type MRA, for instance, can participate in various clusters as well as in different coalitions. They are reactive in the sense that they react to changes in the external environment, which are "sensed" through messages. Although CoBASA agents are not purely reactive, the importance of messages in their behaviour is so relevant that their architecture is more reactive than proactive. This favouring of a more reactive approach should not be considered a handicap, since the interaction that arises from reactive behaviour (simpler) can still create complex behaviours. Finally, they have social ability because they are able to negotiate and cooperate with the other CoBASA agents.

From the most important agent architectures described in chapter 3, it was decided that CoBASA agents would be designed following a **hybrid or layered architecture**. The main reason for this choice is the suitability of this architecture to deal with hybrid situations in which the agents are simultaneously deliberative and reactive, and simplicity. Each layer represents a behaviour composed of a simple behaviour or a set of simple ones. Each agent is decomposed into separate subsystems (behaviours), which are arranged into a hierarchy of interacting layers. Some of the behaviours are typically reactive while the others are proactive. As the behaviours that compose the layers are competing against each other, coordination mechanisms are used.

The agents to be described are the ones identified in 4.2.1, i.e. broker agent (BA), cluster manager agent (CMgA), coordinating agent (CA), and manufacturing resource agent (MRA). These agents are described by indicating their basic behaviours and the agents with whom they are interacting. In its simplest form in a software agent, a behaviour is the result of executing a sequence of programming instructions. This sequence of actions can be represented in a simple way by a graph. Complex behaviour can be achieved using hierarchical composition of simpler behaviours. Complex behaviours are the aggregation of actions and interactions with other behaviours in response to events to achieve some task. The most important behaviours of each of the agents form an agent-layered architecture.

Communication and interaction among individuals and about a domain can only take place if some conceptualisation of that domain exists. To guarantee a common semantic understanding, agents must use an appropriate ontology (Chandrasekaran, Josephson, & Benjamins, 1999; Gómez-Pérez, 2001; Gruber, 1995; Guarino, 1995; Noy & Hafner, 1997; Uschold & Gruninger, 1996) to communicate with their partners. In the case of CoBASA, all the agents need to share some basic concepts, such as *skills*,

contracts, *credits*, *arguments*, *requests*, *services*, and *agent*. Therefore all agents of the proposed architecture share a basic global ontology that models the basic referred concepts.

4.3.1 The Broker

The most important functionalities that the broker agent should be able to perform are: 1) interface with the user, 2) interact with coalitions, and 3) interact with the cluster manager.

Figure 4-42 shows the most important behaviours of the broker agent that are able to implement these functionalities.

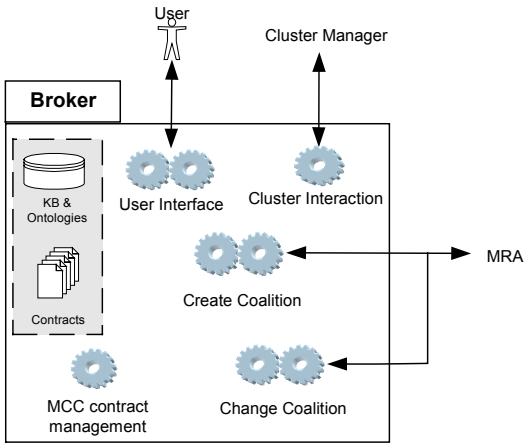

Figure 4-42 – Behaviours of the Broker Agent

The goal of this agent is mainly directed by the behaviours associated with the user interface because it is the user that guides the broker agent towards its main goal, i.e. creating, changing, and dissolving coalitions. The interactions of the broker agent with the cluster manager are supported by the behaviour that is able to ask the cluster for information about candidate agents. This behaviour is executed under the control of the user interface behaviour. The BA, when interacting with coalitions, needs the following behaviours: 1) creating a coalition, 2) changing a coalition, and 3) dissolving a coalition. These behaviours interact with the behaviours from MRAs and CAs and, as in the case of cluster related behaviours, they are also executed under the control of the user interface behaviour. Finally, since the broker agent has to negotiate coalition contracts when it is forming or changing a coalition, there is a behaviour specifically related to the process of creating or changing a MCC contract.

Figure 4-43 shows in a detailed way the most important behaviours of the broker agent using an AUML diagram. The BA gets information from the cluster manager through the behaviour *requestInfo*. The BA uses two behaviours to create a coalition. The first one is used to create the coalition contract (*generateMCC*) while the second behaviour (*negotiateCoalition*) implements the protocol that is followed when the BA negotiates with the candidate agents and coordinator. The BA requires the contract of a coalition whenever that coalition needs to be changed. The contract is

obtained by requesting it from the coordinator agent using the behaviour *getContract,* which implements the protocol used to ask the CA. If a new member is added, the behaviour *renegotiateMCC* is executed. If, on the other hand, a member is deleted, the behaviour *deleteMember* is executed. The behaviour *renegotiateMCC* implements the protocol that is followed by the BA to negotiate the terms of the coalition contract with the new candidate, and renegotiates the new contract with the other coalition members. The dissolution of a coalition is achieved by executing the behaviour *dischargeByPerf*. The BA negotiates with each member of the coalition a discharge by performance through this behaviour.

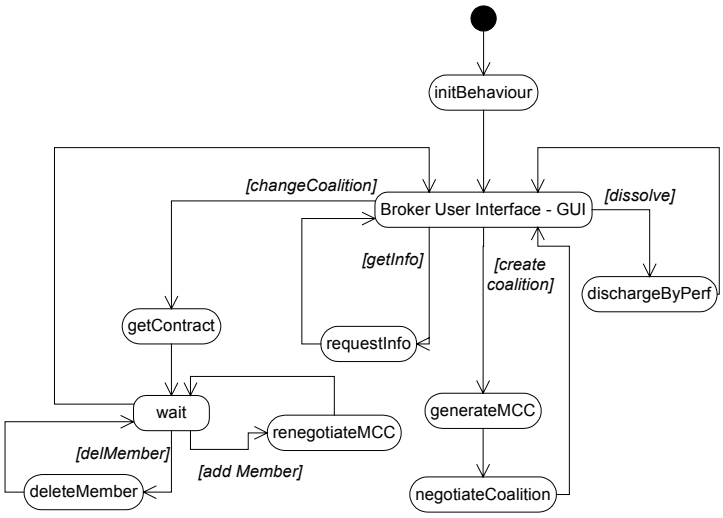

Figure 4-43 – Broker behaviours expressed in AUML

The protocols have different levels of complexity. The simpler ones are just implementations of the FIPA REQUEST protocol (FIPA, 2001), while the complex ones are compositions of FIPA REQUEST.

4.3.2 The Cluster Manager – CMgA

Figure 4-44 shows the most important behaviours of the cluster manager agent as well as with whom this agent interacts. The interaction between the CMgA and the members of the cluster (CAs/MRAs) is supported by the *executeServicesB* behaviour. Only the services that are included in the CAC contract can be requested. This behaviour implements the receiver part of the FIPA REQUEST protocol. The content of the request is the service being requested. The request to update skills and/or credits are examples of used services. This behaviour is purely reactive since it is activated only when a REQUEST command is received. The *user interface* behaviour exists mainly to show cluster activity and for configuration operations. The *RequestInfo* behaviour supports the interaction with the broker agent (BA). This behaviour implements the receiver part of the FIPA REQUEST protocol, and the content of the request message asks for information about cluster members. This is a reactive behaviour. The most complex behaviour is the *ClusterContractManagement*, since it is responsible for

CAC contract negotiation, CAC follow-up, and CAC contract termination. This complex behaviour supports the interaction with MRAs/CAs, either as candidates or members, and is composed of several other behaviours. The protocol that must be followed when a candidate wants to join the cluster (Figure 4-13), and the various protocols to terminate the CAC contract are implemented here (Figure 4-15, Figure 4-16, Figure 4-17, Figure 4-18). These behaviours are reactive and proactive.

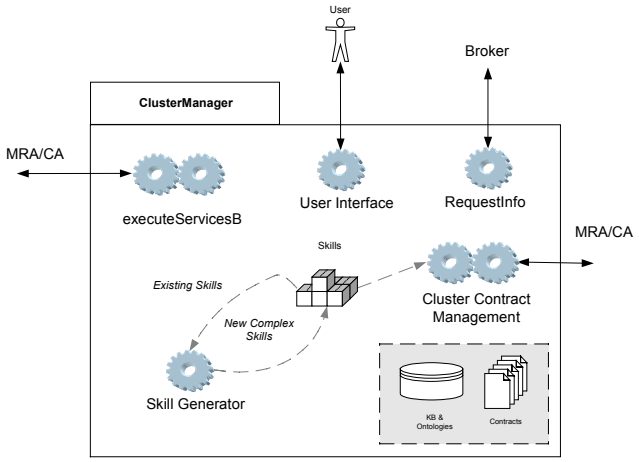

Figure 4-44 – Behaviours of the Cluster Manager Agent (CMgA)

The *SkillGenerator* behaviour is responsible for generating new skills. The behaviour departs from a set of existing skills and a set of rules in the CMgA knowledge base to generate the skills that can be created. These composed skills represent the potential skills that can be created out of the cluster.

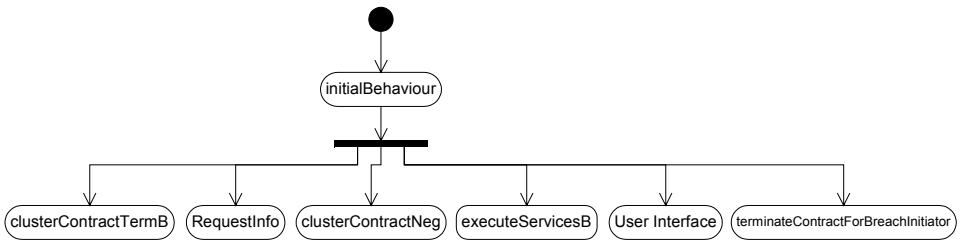

Figure 4-45 – Active CMgA behaviours

The behaviours that are always active are represented in Figure 4-45 using an AUML activity diagram. For instance, the behaviour *SkillGenerator* is not shown because this behaviour only runs after an adhesion process terminates. The behaviour *clusterContractNeg* implements the protocol, from the CMgA side, followed when MRA/CA agents adhere to the cluster. This behaviour is detailed in Figure 4-46.

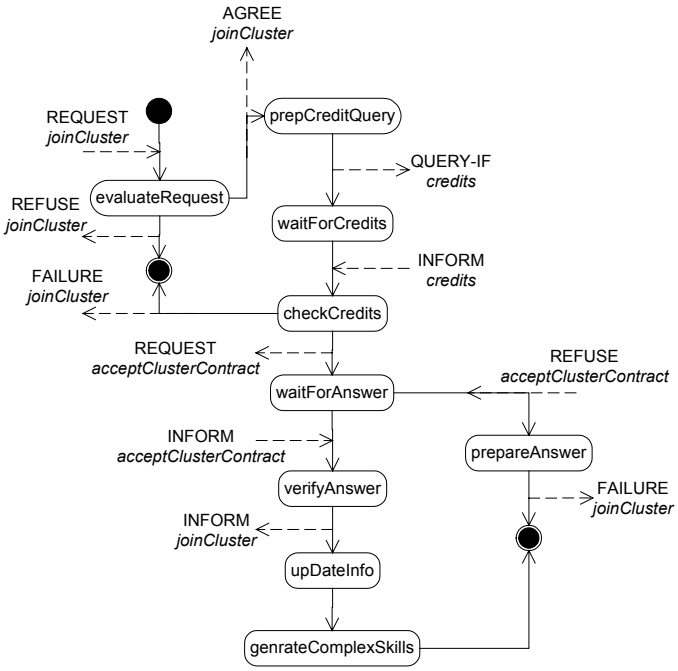

Figure 4-46 – Behaviour *clusterContractNeg*

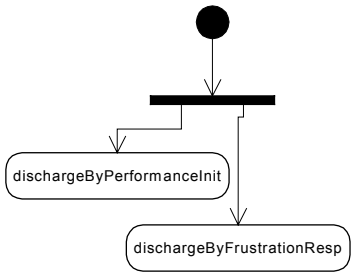

Figure 4-47 – Behaviour *clusterContractTermB*

The behaviour *clusterContractTermB* is also a composition of behaviours and is shown in Figure 4-47. Please note that *dischargeByFrustrationResp* is a behaviour that is waiting for the REQUEST message from one of the cluster members that wants to terminate the CAC by frustration. The behaviour *dischargebyPerformanceInit*, is a timed behaviour that runs from time to time to verify if any of the CAC contracts have terminated.

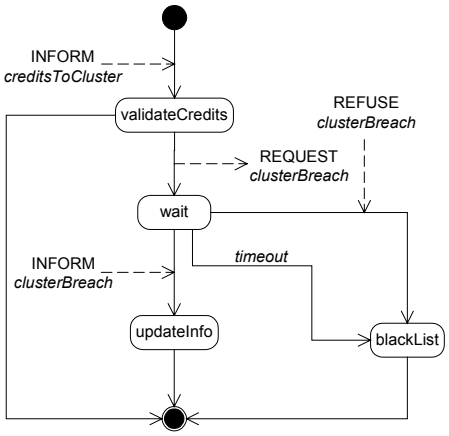

Figure 4-48 – Behaviour *terminateContractForBreachInitiator*

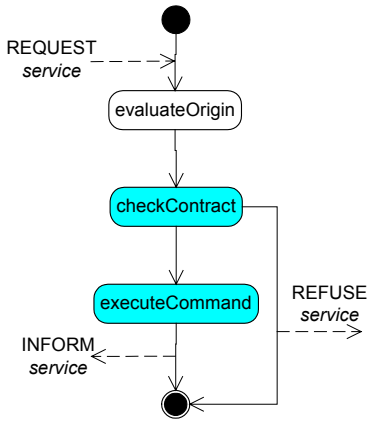

Figure 4-49 – Behaviour *executeServicesB*

The statechart of the behaviour *terminateContractForBreachInitiator* is illustrated in Figure 4-48. If after serving the request for an update of credits, the CMgA finds out that the agent's credits are not sufficient to stay in the cluster it starts the negotiation process to breach the CAC contract with that

agent. Please note that a cluster member informs the CMgA to update its credits when it finishes its participation in a coalition. The statechart that describes the *executeServicesB* behaviour is shown in Figure 4-49.

4.3.3 The Manufacturing Resource Agent - MRA

The manufacturing resource agent (MRA) was previously defined as being a manufacturing component extended with agent skills, which corresponds to its agentification in order to be able to participate in the CoBASA society. The agentification could have been achieved by developing an agent that could simultaneously interact with the manufacturing equipment controller, and manage its CoBASA social activities (cluster joining and coalition participation) that includes, among others, the contract negotiation and composition of skills tasks. In addition, the requirements imposed by the need for interaction to be maintained with the controller (short response time, interaction protocol specificities, ...) suggest the use of a dedicated agent to interact with the controller. Therefore, as the two activities are both very demanding to be accomplished by one agent only, the adopted approach was to separate the functionalities and to have one dedicated agent to interact with the controller – Agent Machine Interface (AMI) and a generic agent specialised in the CoBASA social activities, namely, contract negotiation and skills composition – Generic Agent (GA).

The Agent Machine Interface (**AMI**) is the agent that is directly connected to the physical controller. It acts as a kind of device driver to the agent specialised in negotiation and skills composition (GA). For each different controller there should be one AMI. Nevertheless, only one type of GA is required to build different MRA types. The AMI is the agent wrapper of a manufacturing component that exports the functionalities existing in its physical controller.

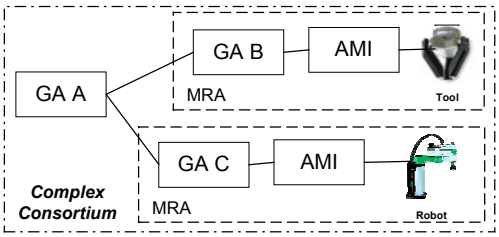

Figure 4-50 – Hierarchy of consortia

Since the functionality of the CA is similar to the requirements of the generic agent (GA), the CA is effectively implemented by the same GA code (Figure 4-50).

The set of potential skills (operations) of a coalition led by a GA is composed of the basic skills brought in by coalition members plus the complex skills that are locally generated by composition of the basic ones. When the GA is leading an AMI the offered skills (operations) are those included in the AMI. In this situation, the coalition is a MRA, and therefore its basic skills are those offered by the AMI. When leading a complex consortium or coalition, the basic operations come from the set of

skills brought in to the coalition by all its members. In this case, the high-level GA coordinates each of the other lower-level GAs that are leading other coalitions. In this situation the GA is nothing other than a coordinating agent (CA). This strategy facilitates the implementation because GAs are used either as CAs or builders of MRAs, which improves the flexibility of CoBASA, since it is possible to create MRAs as well as CAs using only configuration instead of additional programming effort.

4.3.4 The Generic Agent – GA

Figure 4-51 depicts the building blocks of the generic agent architecture. It clearly indicates the two types of contracts to which the agent can be bound according to the two possible roles: *membership contracts* and *coordination contracts*.

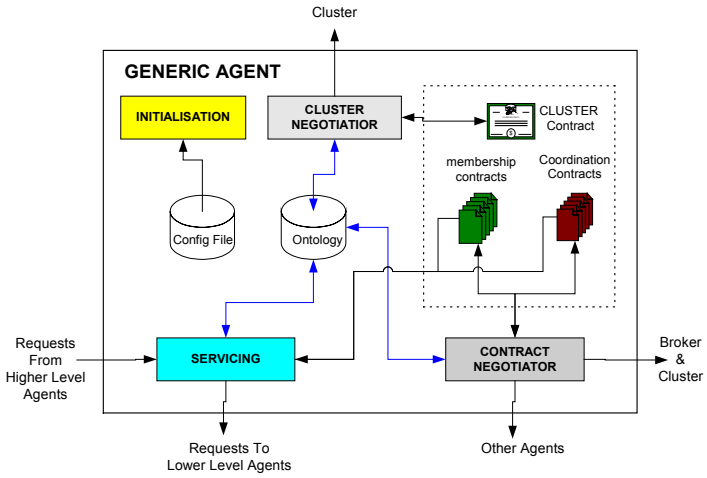

Figure 4-51 – Generic agent building blocks architecture

The configuration file establishes the base for the individualisation of a generic agent. This file contains the relevant information, such as its skills and credits, required to help the agent joining the cluster and eventually joining coalitions.

The most important behaviours of a generic agent are (Barata & Camarinha-Matos, 2002a):

- The **initialisation**, which is used to read the configuration file, to create the instances of the concepts of the ontology, and other data structures initialisation.
- The **cluster negotiator**, which is used when the agent joins a cluster.
- The **contract negotiator**, which supports the negotiation of cluster contracts and the two types of consortium contracts. The negotiated contracts are maintained in two different data structures depending on its type: *membership contracts* and *coordination contracts*.
- **Servicing** represents the operative part of the agent where the complex requests from a higher-level coordinator agent (CA), to whom the agent has a membership contract, are received and split into basic commands to be executed, either internally or by the lower level agents co-ordinated by this one.

A detailed illustration of the most important behaviours for the GA agent is shown in Figure 4-52, using an AUML diagram (Barata & Camarinha-Matos, 2002b).

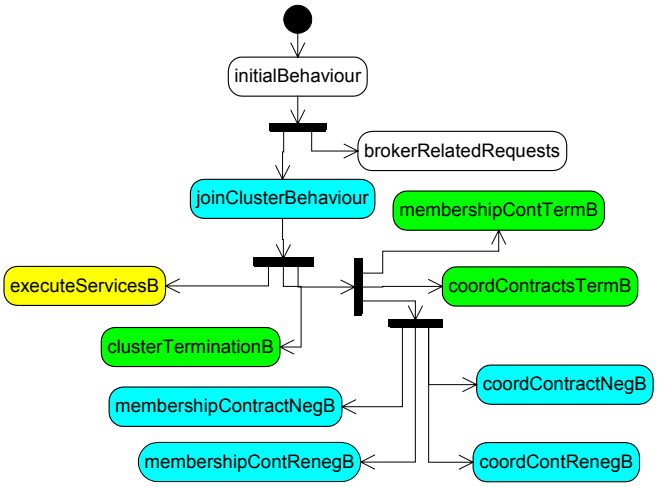

Figure 4-52 – Active behaviours for the Generic Agent (GA)

The initialBehaviour runs at the beginning of the execution of the agent, and initialises and configures the agent data structures. The agent uses the behaviour ***joinClusterBehaviour*** to join the cluster. One thing it does, in this phase, is negotiating the cluster contract. Therefore, this behaviour implements the protocol required to negotiate this contract with the cluster manager (initiator role). The statechart diagram for this behaviour is not shown because it is quite similar to the behaviour *clusterContractNeg* (Figure 4-46) that belongs to the cluster manager. The behaviours ***membershipContTermB***, ***clusterTerminationB***, and ***coordContractsTermB*** deal with the termination of the three types of contract each GA has. An example of the complexity associated with the termination occurs when a coalition member reaches the end of its contract. In this situation, this member negotiates the termination of the contract through specific behaviours with its coordinator. Since the departure of the member changes the coalition contract, the coordinator of this coalition needs to renegotiate the contract alteration with the other members. Furthermore, as the coalition skills might also have been reduced the coordinator might need to renegotiate the contracts in which the affected skills have been promised. This happens because of the hierarchical nature of the consortia organisation.

The *clusterTerminationB* behaviour is detailed in Figure 4-53. The behaviours that implement the receiver part of the protocols to terminate a CAC contract by performance (*clusterDischargeByPerformanceResp*) and breach (*clusterTermContractForBreachResp*) are indicated as well as the behaviour responsible for terminating the CAC contract by frustration. However, in the latter case, the GA plays the initiator role (*clusterDischargeByFrustInit*). This behaviour is executed whenever the user asks the agent to leave the cluster (section 4.2.2.4).

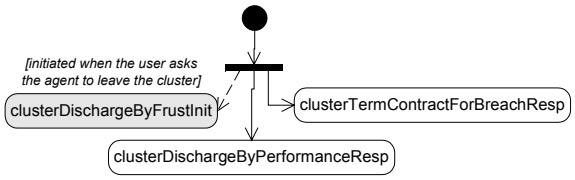

Figure 4-53 – Behaviour *clusterTerminationB*

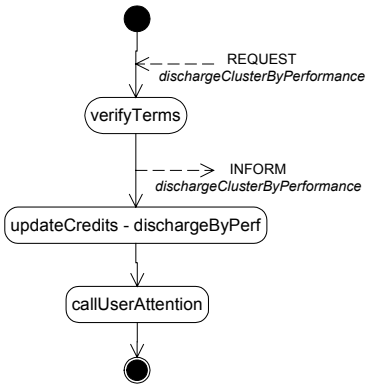

Figure 4-54 – Behaviour *clusterDischargeByPerformanceResp*

The statechart diagrams for the *clusterDischargeByPerformanceResp* is illustrated in Figure 4-54, and *clusterTermContractForBreachResp* in Figure 4-55 to detail the protocol and to show when GA agents update their credits.

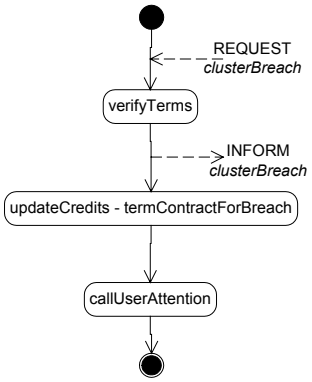

Figure 4-55 - Behaviour *clusterTermContractForBreachResp*

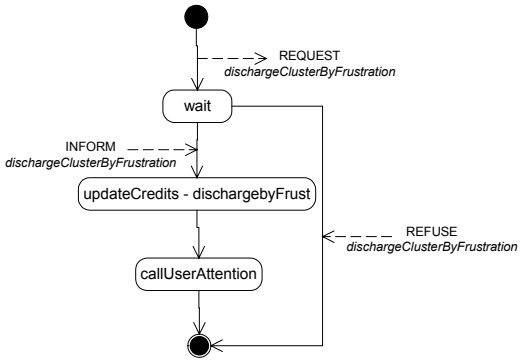

Figure 4-56 – Behaviour *clusterDischargeByFrustInit*

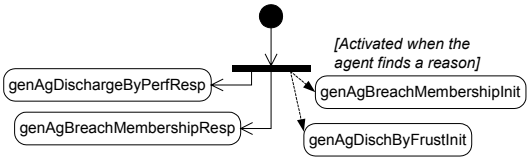

Figure 4-57 – Behaviour *membershipContTermB*

The behaviour ***membershipContTermB*** deals with the termination of MCC contracts in which the agent is participating as a member, which is composed of 4 main behaviours (Figure 4-57): *genAgDischargeByPerfResp*, *genAgBreachMembershipResp*, *genAgDischByFrustInit*, and *genAgBreachMembershipInit*. The first two (*genAgDischargeByPerfResp* and *genAgBreachMembershipResp)* are behaviours that are always activated to implement the receiver part of the protocol followed to terminate the membership MCC contract. When a coordinator wants to terminate the contract of one of its members by performance its behaviour *genAgDischargeByPerfInit*

sends a REQUEST command to the agent, which is received by the behaviour *genAgDischargeByPerfResp* (Figure 4-58) of the agent being requested. In this behaviour, the message is validated by checking if it is coming from the coordinator of any of the agents' membership MCC contracts, and if the reason for contract discharging is valid. In the case where the request is valid, the terminating agent replies with an INFORM message and, after that, it removes the contract from the entity that stores its membership contracts, updates its credits, and informs the cluster about them. When the termination is due to a bad behaviour from an agent participating in a coalition, its coordinator, through the behaviour *genAgBreachCoordInit*, sends a REQUEST to breach the agent participation, which is received by the behaviour *genAgBreachMembershipResp* (Figure 4-59). The actions of this behaviour are similar to the previous one, with the exception that, in this case, the terminating agent is guilty and therefore is penalised when updating its credits, which is done by collecting the score for its participation in the coalition, in the *terminateContractForBreachGuilty* part of the MCC contract being terminated.

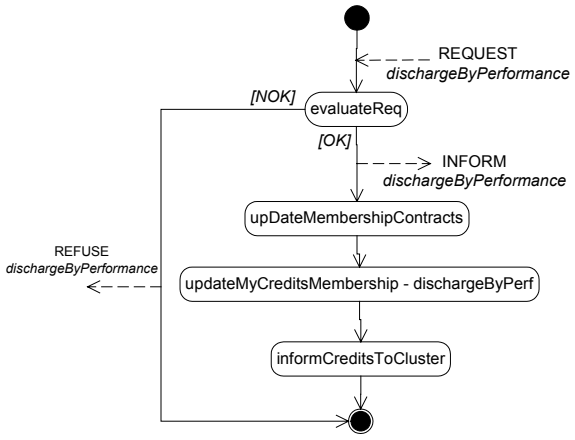

Figure 4-58 – Behaviour *genAgDischargeByPerfResp*

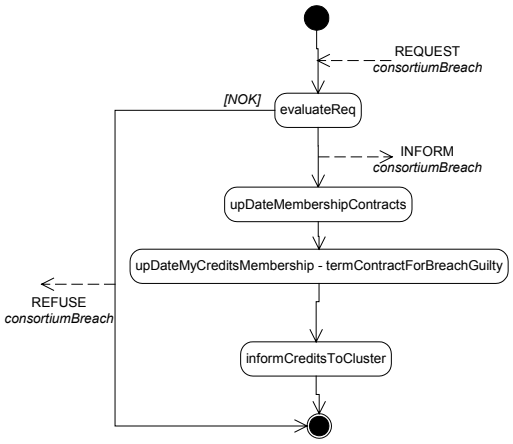

Figure 4-59 - Behaviour *genAgBreachMembershipResp*

The behaviour *genAgDischByFrustInit* is activated by the behaviour *verifyIfFrustExists*, which verifies if a frustration condition exists. In this situation the agent tries to terminate its participation claiming frustration. On the other hand, the *genAgBreachMembershipInit* is called when a coalition member finds out it has a reason to breach its membership MCC contract with its coordinator (coordinator bad behaviour), for instance, when the coordinator does not answer a request from it.

The ***coordContractsTermB*** deals with the termination of MCC contracts in which the agent is a coordinator, which is composed of five main behaviours: *verifyIfFrustExists*, *genAgDischByFrustResp, genAgBreachCoordResp, genAgDischByPerfInit,* and *genAgBreachCoordInit* (Figure 4-60). The contracts that are considered are the ones the agent is coordinating (coordinated contracts).

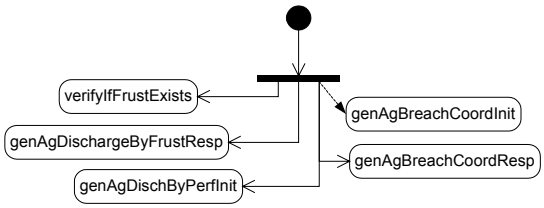

Figure 4-60 – Behaviour *coordContractsTermB*

The behaviour *genAgDischByFrustResp* (Figure 4-61) implements the receiver part of the protocol to finalise the participation of one of the coordinated members that claimed termination by frustration. Please note that the requester uses the behaviour *genAgDischByFrustInit* to interact with this one. The requester behaviour deals with a membership contract while this one deals with a coordinating contract. Note also that the MCC contract that regulates a coalition is a coordinating contract in the coordinator, while the same contract in the members of that coalition is a membership contract. Since the exit of a coalition member involves loosing its skills from the coalition, and changes to the coalition contract, the agent must negotiate the changes that occurred in the coalition contract (shadowed area) with the other members. After this, the coordinator might need to renegotiate with higher-level coalitions, in case the leaving skills affect the ones that were promised to higher-level coalitions, using the behaviour *renegWCoordB*.

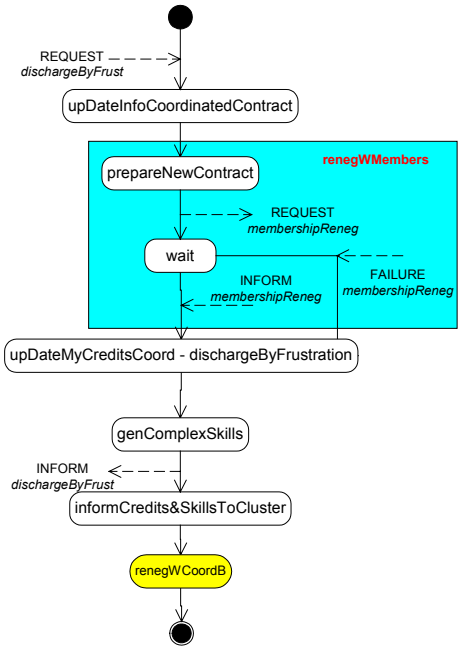

Figure 4-61 – Behaviour *genAgDischByFrustResp*

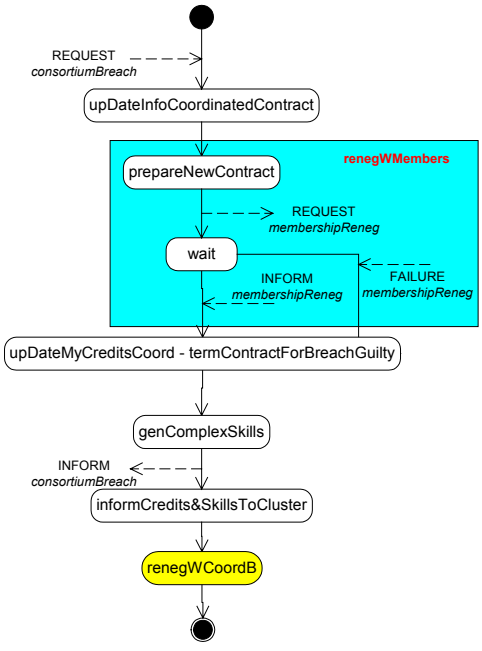

Figure 4-62 – Behaviour *genAgBreachCoordResp*

The behaviour *genAgBreachCoordInit* is only activated when the coordinator has the right to breach a contract (for instance when one of its members does not reply to a request), and it implements

the sender part of the protocol used to terminate the coalition member participation. The behaviour *genAgBreachMembershiResp* is the one used by the member being requested to reply to this one.

The behaviour *genAgBreachCoordResp* (Figure 4-62) implements the receiver part of the protocol to finalise the participation of a coalition member due to bad behaviour of the coordinator itself. It is similar to the behaviour *genAgDischByFrustResp*. As in previous cases, this behaviour deals with coordinating contracts while the requester behaviour *genAgBreachMembershipInit* deals with membership contracts.

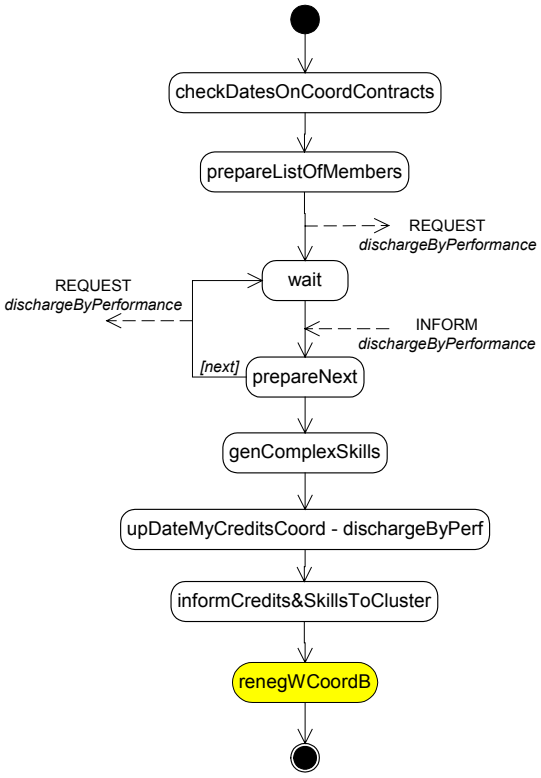

Figure 4-63 – Behaviour *genAgDischByPerfInit*

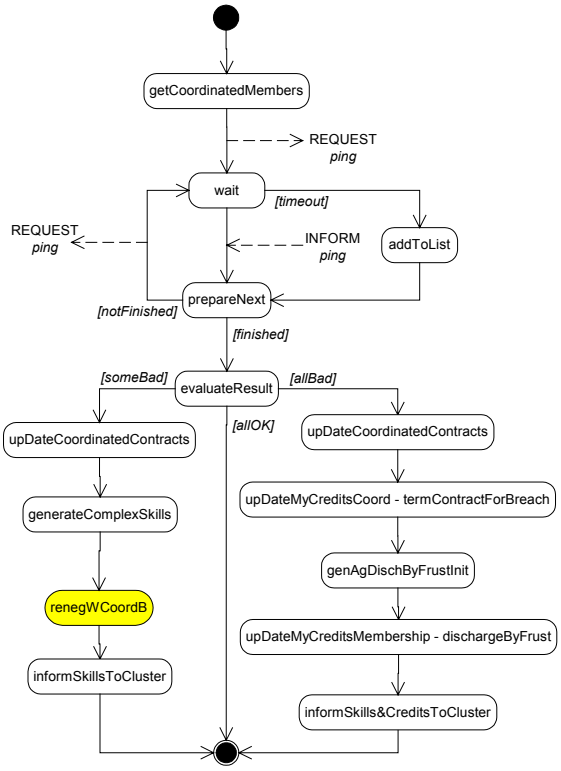

Figure 4-64 – Behaviour *verifyFrustExists*

The behaviour *genAgDischByPerfInit* (Figure 4-63) is always executing because it is verifying if any of the coordinated contracts has reached the end. This behaviour interacts with the *genAgDischargeByPerfResp* behaviour of a member agent. Finally, the behaviour *verifyFrustExists* (Figure 4-64) is also always running to detect if a situation of frustration exists. The members of coordinated MCC contracts are checked if they are alive (pinged). If none of them reply, then the coordinator cannot keep its participation in higher-level coalitions. Therefore, the coordinator executes the behaviour *genAgDischByFrustInit* to terminate its participation, claiming frustration reasons. Please note that *verifyFrustExists* is in the coordinated contracts parts because the verification is made extracting information from the coordinated contracts.

Figure 4-65 illustrates how the behaviours *coordContractsTermB* and *membershipContTermB* interact. The *AGENT A* coordinates the *AGENT B* which coordinates the *AGENT C*. The interactions of the

behaviours that compose membership contract termination (*membershipContTermB*) and those associated with coordinated contract termination (*coordContractsTermB*) are indicated and, for each pair of interacting behaviours, the arrows indicate the behaviour performing the receiving role.

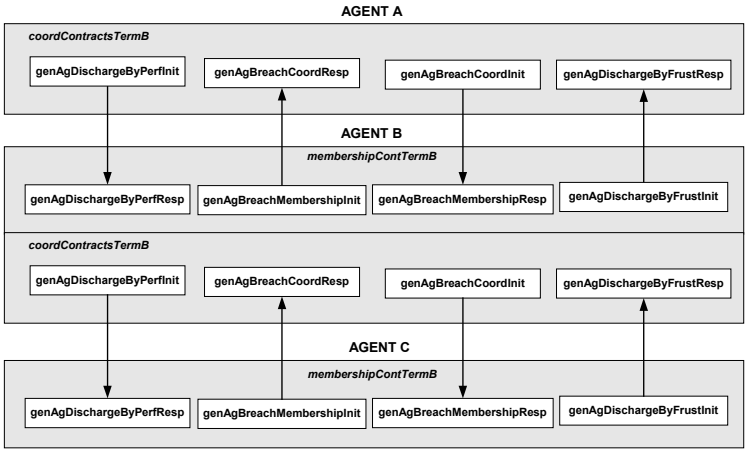

Figure 4-65 – Interaction between *coordContractsTermB* and *membershipContTermB*

The GA uses the behaviour ***membershipContractNegB*** (Figure 4-66) to negotiate with the broker the participation of an MRA in a coalition. This negotiation protocol also includes the contract signature process. The full protocol consists of evaluating the proposal, after the REQUEST to participate in a coalition is received and then, if the GA accepts the MCC contract terms, the signing process takes place when the agent receives the request to sign the contract. The complete MCC contract of the coalition is received with the request to sign it.

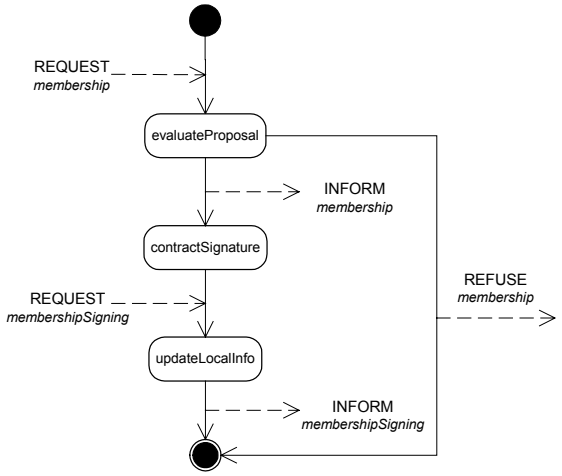

Figure 4-66 – Activity diagram for *membershipContractNegB*

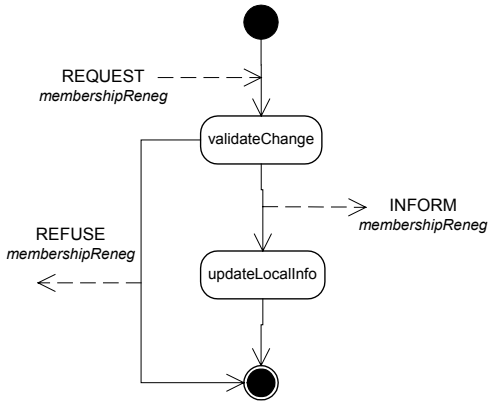

Figure 4-67 – Activity diagram for *membershipContRenegB*

The behaviour ***membershipContRenegB*** (Figure 4-67) is used when the coalition contract of a member needs to be renegotiated. This occurs whenever a coalition is changed, which implies alterations to the coalition contract and, consequently, the remaining members must negotiate the new contract version.

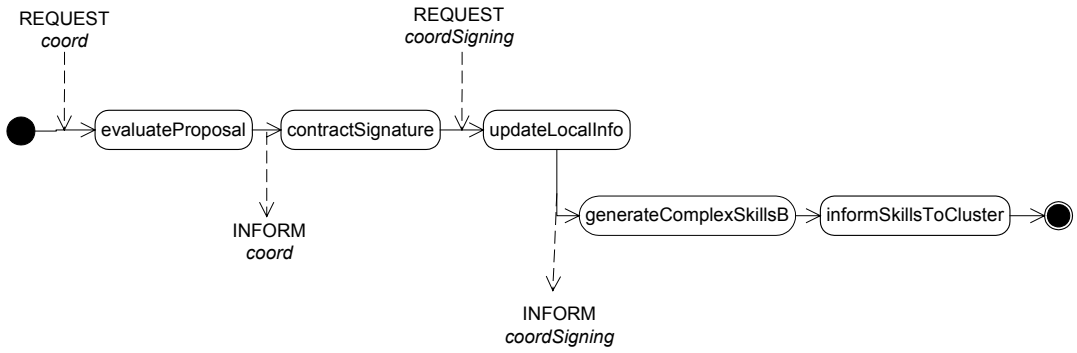

Figure 4-68 – Activity diagram for *coordContractNegB*

The behaviour ***coordContractNegB*** implements the negotiation protocol with the broker, when the GA participates as coordinator of a new coalition (Figure 4-68). After the negotiation, the contract signature is done and, after that, the *generateComplexSkillsB* is activated to generate the complex skills. At the end of the behaviour, the GA updates the skills in the cluster, which in fact correspond to the skills of the coalition being formed.

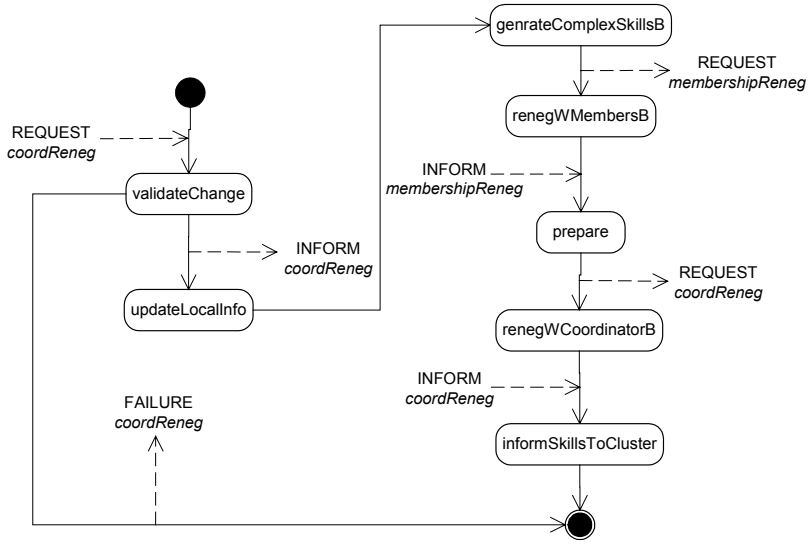

Figure 4-69 – Activity diagram for *coordContRenegB*

The ***coordContRenegB*** (Figure 4-69) is the most complex contract related behaviour, which starts execution when a coordinator (GA) is requested to renegotiate a coordinated coalition contract. This happens either because of a natural reason (the broker needs to change a coalition) or because one of the coalition members wants to change its participation by increasing or reducing the skills it brings in to the coalition. The complexity arises from the need to generate complex skills after contract updating and, after that, the need to renegotiate with the remaining coalition members and the possible renegotiation with higher-level coordinators, in case the coalition participates in others coalitions. The renegotiation starts when one of the coalition members issues a REQUEST *coordReneg* command. If this request is accepted, the GA updates the corresponding coordinated MCC contract, and then

updates its skills using the *generateComplexSkillsB* because a change in the contract might have changed the set of skills brought into the coalition. The behaviour *renegWMembersB* is used in the renegotiation of the new contract version with the other coalition members, which, in turn, use the behaviour ***membershipContRenegB***. Since the coordinator might be participating in other coalitions it might need to renegotiate the terms of its participation in higher-level coalitions. This is only necessary if the renegotiation process changed the skills brought in by the GA to coalitions where it participates as a member. The GA verifies this by checking in its membership MCC contract if the skills being promised are still valid. This renegotiation with higher-level coalitions is achieved through the behaviour *renegWCoordinatorB*, which interacts directly with the behaviour ***coordContRenegB*** that belongs to the coordinators of the higher-level coalitions. At the end of this process the GA informs the cluster about its new skills.

The behaviour ***executeServicesB*** is responsible for the execution of requests and is the one that gives execution ability to MRAs or coalitions. To better understand the operation of this behaviour it is important to consider what was mentioned in 4.2.3.2 about skills and their execution. When the GA receives a REQUEST message to execute a skill (command), it is first validated by checking if it was promised in any of the *membership contracts*.

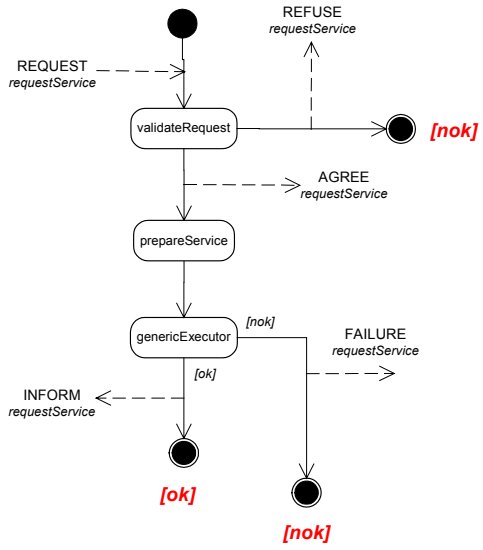

Figure 4-70 – Activity diagram for *executeServicesB*

The behaviour *prepareService* gets the code to be executed from the model of the skill that was requested, and delivers it to the *genericExecutor* behaviour. This behaviour then executes the code according to the model. If the skill being executed is complex, the behaviour sends various requests for services to the lower levels. If the service is well executed then *executeServicesB* replies to the higher-level requester using an INFORM message, which states that the command has been successfully executed. The generic machine associated with the behaviour *genericExecutor* needs another behaviour (*executeSimple*) that is called anytime a simple skill needs to be activated. Please recall that simple skills are those skills that were brought into the coalition by its members. For

instance, if the generic executor machine is a Prolog engine as defined in section 4.2.3.2, the predicate *executeSkill* is responsible for activating the behaviour *executeSimple*.

4.3.5 The Agent Machine Interface – AMI

An AMI is always connected to a physical controller (Robot, PLC, …) and implements the services (skills) supported by the physical component. The generic AMI agent is a simple agent composed of two simple behaviours. The first behaviour accepts requests for services from the MRA and then calls the wrapper to execute the requested service. An example of a request for a robot "move" using world coordinates is:

((requestService serviceName moveWcService
arg1 (5DofType alfaArg 45.0 vel 80.0 xArg 34.0 yArg 12.0 zArg 10.0)))

The service asked is *moveWcService* and the argument is a frame from the class *5DofType*. This message is coded using the FIPA SL0 language. When the command is executed, the enquirer MRA is informed. The second behaviour exists only for user interface purposes (Figure 4-71).

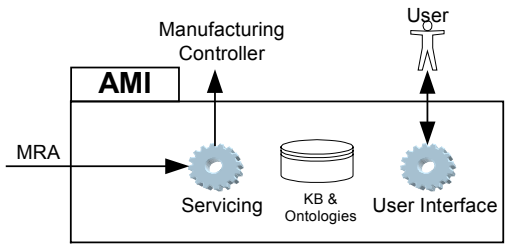

Figure 4-71 – Behaviours of the AMI

To integrate these legacy components in the agents' framework it is necessary to develop a software wrapper to hide the details of each component. The wrapper acts as an abstract machine to the agent supplying primitives that represent the functionality of the physical component and its local controller. The agent machine interface (AMI) accesses the wrapper using a local software interface (proxy), where all services of the wrapper are defined. Figure 4-72 shows a high level representation of an operative agent indicating how the wrapper integrates a manufacturing component (robot).

The Agent Machine Interface implementation is generic, i.e. an AMI can be connected to different distributed components (proxy) just by configuring what the services of that proxy are and the name/address of the component.

The generic agent tied to the AMI behaves slightly differently to other agents, at the initialisation phase. In this situation, the GA reads an instance of a coalition contract between itself and the AMI from a contract representation file, and establishes a coalition. The member promises part (AMI) of the contract contains the services supplied by the AMI. The agents not connected to an AMI, on the

other hand, are configured not to read any contract representation file at initialisation time. This approach is very flexible because it permits the creation (generate) of any type of manufacturing agent just by configuring an AMI and the coalition contract between the agent and the AMI. The only part of the system that is dependent on the physical component is of course the wrapper.

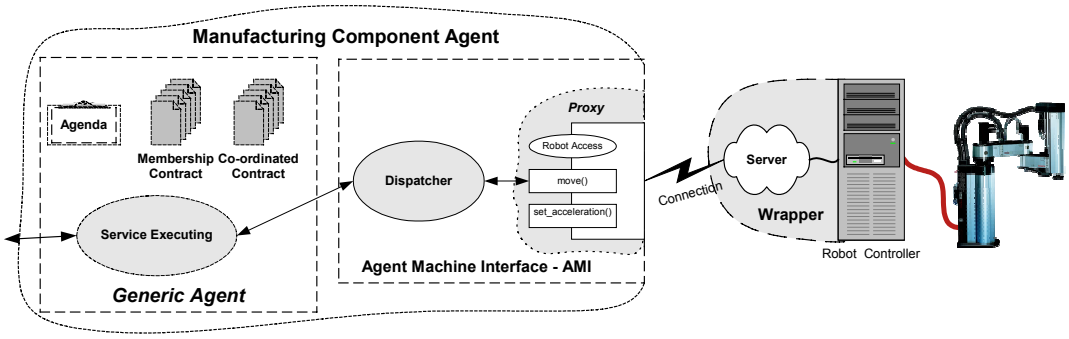

Figure 4-72 - Physical component integration

5 Prototype and Validation

This chapter discusses the aspects related to the validation of the work described in this thesis according to three different dimensions. The first dimension of validation (feasibility) consists in showing that a prototype was implemented. Therefore the tools used to develop the prototype and its implementations details are described as well as the steps of the methodology required to operate the prototype. The second dimension (adequateness) consists in describing that the system was applied to a real manufacturing system and external experts were asked to test the prototype. The manufacturing scenario is described and the experts' opinion is included. Finally, the third dimension (theoretical) consists in answering several questions adequate to prove the validity of applied research.

5.1 INTRODUCTION

The approach followed to validate the concepts and ideas developed in the framework of this thesis considers three dimensions of validation (Table 5-1). This table shows the three validation dimensions and, for each one, it indicates how the dimension is considered, and the sections in this chapter that describe aspects related to the validation in that dimension.

The first dimension, feasibility, is validated by proving that the concepts can be implemented (feasible). By showing that a prototype was implemented the feasibility of the system is proved. Only after the implementation it is possible to obtain results about the effectiveness of the proposed architecture. This is true even in a situation where the methods and the technologies required to implement the finest solution are not yet well mature. The prototype that was developed implements the CoBASA architecture that was introduced in the last chapter. It is composed of several agents that were implemented using a number of supporting tools, of which the most important are the agent

development tool and the ontology development tool. The key elements of the architecture were defined and programmed: the agents, the ontologies, the data structures, negotiation protocols, definition of the messages to be exchanged, the contracts, etc. In addition, a number of legacy controllers of manufacturing components were also integrated in the prototype.

Dimension	How to validate	Section in chapter
Feasibility	Showing that CoBASA was implemented.	Development tools; Prototype description; The steps of the methodology
Adequateness	Applied to a real manufacturing cell.	Experimental validation
	Asking external experts.	External validation
Theoretical	Answering Olesen points.	Theoretical validation

Table 5-1 – Dimensions of validation

The choice of the agent development environment is much more important than it seems at first sight. In fact, an agent development tool with an adequate programming model can considerably reduce the programming effort as well as increasing the reliability and maintainability of the prototype. Therefore it is worth to carefully choose the agent development tool. The same situation happens with the supporting tool for the development and management of ontologies, which helps reduce the overall complexity of dealing with the semantics associated with messages, and the creation and management of the ontology. Because these two tools played a so important role in the implementation of the prototype, a special subsection is devoted to describing them.

The second dimension of validation, adequateness, is accomplished by proving that the CoBASA prototype serves its purpose. This is proved using two approaches: the first one by applying the prototype to a real manufacturing system and the second one by asking the opinion of external experts about the adequateness of the system. The first approach is described in the section "experimental validation" that shows how the CoBASA prototype was used in the NovaFlex Cell (Barata & Camarinha-Matos, 1994), which is a flexible assembly system (FAS) composed of industrial equipment. The second approach is described in the section "external validation", in which two external experts with wide experience in managing projects dealing with shop floor modifications were invited to give their expert opinions of the CoBASA prototype.

The third dimension of validation is called theoretical because it consists of answering several questions that were defined by Olesen in (Olesen, 1992), which seems adequate for applied research. The Olesen's questions are described and answered in the section theoretical validation.

5.2 DEVELOPMENT TOOLS

Agent based technologies, although simple to understand, cannot realise their full potential without adequate environments for the development of agents. Several tools were searched and analysed that led to the choice of Java Agent DEvelopment framework – JADE (JADE, 2001), which is a FIPA compliant Java based development environment.

Another aspect that was also considered was modelling. In CoBASA, the entire set of agents needs to share some basic concepts such as skills, contracts, credits, arguments, requests, services, and agent, which, in general, are better modelled with knowledge-based models. Since most of the available multiagent development environments are Java based, which is the case with JADE, and the Java language does not directly support knowledge models, it was necessary to find a tool that could support these models within a Java environment. The choice was Protégé-2000 (Protégé-2000, 2000), which is a Java based open-source modelling platform.

5.2.1 Multiagent development environment

The development framework needed to implement the CoBASA community of agents requires support for agent programming (development phase), and an infrastructure to support the community (deployment and execution phases). This supporting infrastructure should support the following activities:

- Agent management including supervisory control, authentication, and registration/deregistration of agents. This activity should support a white-pages mechanism in which a list of agent identifiers of the entire set of registered agents is maintained.
- A mechanism that can be used by registered agents to advertise their services (also called yellow pages).
- A message transport system responsible for the communications between the agents. This software layer hides the communication details from the programmer whenever the agents need to exchange messages. This is the software layer that supports distributed agents over remote and heterogeneous computers. Do not confuse it with the requirements at the Agent Communication Language.
- Support for heterogeneous agents running in different and possible distributed computer platforms.

Figure 5-1 gives an idea about the required multiagent architecture, indicating the services to be included in the supporting environment and how the agents to be developed interact with the infrastructure.

An important characteristic of the supporting infrastructure is that it should be FIPA compliant to allow interoperability between agent systems. CoBASA is foreseen to be used in different industrial

environments that might be composed of different kinds of manufacturing agents, which might be developed using different programming environments and executing hardware. Therefore, it is important to be FIPA compliant in order not to be restricted to a specific kind of manufacturing agent.

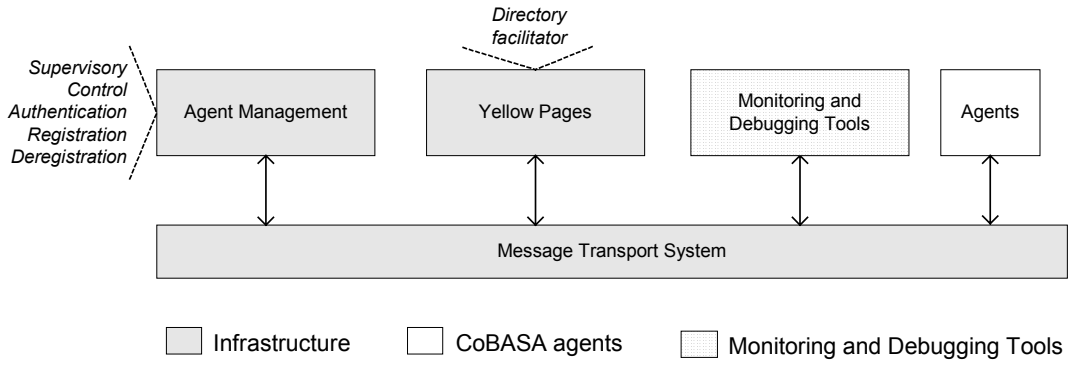

Figure 5-1 - Requirements of the multiagent architecture

Agent programming support can follow two extreme approaches: hiding completely the lower level implementation primitives from or exposing them entirely to the programmer. Hiding these primitives is convenient to speed up the programming process and the learning curve; however, it can be inconvenient for flexibility reasons. In the specific case of developing a prototype such as CoBASA, in which different programming environments need to be integrated, it is important that the programmer has access to some of these primitives, which control lower level implementation details.

For agent-programming support the following aspects are required:

1. A library (API) of reusable software components that closely match an agent model. In general APIs provide a good trade off between hiding some implementation details while keeping enough flexibility to allow the programmer to be able to integrate other applications.
2. Tools to manage the running community of agents as well as to monitor and debug them.

This library (API) can be regarded as a middleware facility that hides from the programmer implementation details such as the agent abstract model itself, the agent communication language, the communication between agents via the supporting infrastructure, ontologies, communication with the supporting infrastructure, and interaction protocols. This API should support an agent model that implements the distinguishable agent properties such as autonomy, reactivity, and social behaviour. Autonomy can be achieved if the agent model supports different paradigms such as logic programming, rule based systems, or behaviours, which are logical threads of software to carry on agent duties. If the agent model does not directly support logic programming or rule based systems it should still be possible to integrate these programming paradigms with the agent model. Independently of the used programming paradigm, what is effectively required to support autonomy is that the agent is an independent process or thread that is executed independently from other agents. Furthermore, this autonomous process should ensure that the agent actions are not based only on the type of messages

received but also on the internal state represented by the execution of its own process or thread. Reactivity can be achieved as long as the agent model supports a mechanism to receive messages. Actions can be executed after a message is received. Social activity is ensured by the ability of the agent model to support conversations. Examples of mechanisms to support conversations are: agent communication languages, message send and receive mechanisms, communication protocols, and ontologies.

An important final requirement for CoBASA development environment is that it should be open source mainly for financial reasons.

Tool selection. A large number of development platforms for multiagent systems have been proposed in the last few years. The most appropriate one should be selected using the requirements introduced in the last subsection. After a preliminary analysis, the set of candidates to be considered was composed of the following environments JADE (JADE, 2001), JATLite (Jeon, Petrie, & Cutkosky, 2000), Infosleuth (Nodine et al., 2000), RETSINA (Sycara, Paolucci, Van Velsen, & Giampapa, 2001), IBMAglets (IBM-Aglets, 2001), Open Agent Architecture - OAA (Cohen, Cheyer, Wang, & Baeg, 1998; Martin, Cheyer, & Moran, 1999), Jack (*JACK*, 2000), FIPA-OS (FIPA-OS, 2001), Zeus (Nwana, Ndumu, Lee, & Collis, 1999), and Agentbuilder (AgentBuilder, 2001). This set was considered based on suggestions from other researchers with whom the author has worked, and by Internet searching. A remarkable number of agent construction tools can be found in http://www.agentbuilder.com/AgentTools/index.html.

The selection process was based on the Analytic Hierarchy Process – AHP (Anderson, Sweeney, & Williams, 1986), which is adequate for decision analysis with multiple criteria. The process requires the decision maker to provide judgements about the relative importance of each of the criteria and then specify a preference for each decision alternative relative to each criterion. The output of the AHP is a prioritised ranking indicating the overall preference. The decision alternatives are the set of candidate tools. The criteria set for this process was defined based on the requirements presented when the CoBASA requirements were introduced. Accordingly, the following set of criteria was defined:

- ***FIPA compliant (FIPA).***
- ***Open-source (OS).***
- ***Agent model (AM).*** The agent model should support the distinguishable agent properties (autonomy, reactivity, and social ability).
- ***Lower level control (LLC).*** The programming environment should include an API that lets the programmer access and control some lower level mechanisms of the agent implementation to increase flexibility and integration capability. In the CoBASA case some low level control is required.

- ***Easy agent creation (EAC).*** This criterion evaluates the effort that is needed to create an agent. Usually, the easier the agent creation, the more the implementation details are hidden from the programmer. This means that this point is usually associated with the previous one. In CoBASA a situation of lower level ***control over better agent creation*** is favoured***.***
- ***Agent Communication Languages (ACL)*** primitives should be included.
- ***Community infrastructure (CI)***. This criterion evaluates how well the development environment provides middleware for the support of the community of agents.
- ***Monitoring and debugging (M&D)*** tools.
- ***Conversation protocols (CP)***. Evaluates if conversation protocols are already implemented and ready to be used by the developer.
- ***Integration capability (IC)***. This criterion evaluates how far it is possible to integrate with the agents other programming paradigms such as: different languages (Prolog, Lisp, etc), middleware such as CORBA and DCOM, and rule-based systems (e.g. Jess).
- ***Documentation (D).***
- ***Support (S).***
- ***Number of users (NU).***
- ***User-friendliness of the system (UF).***
- ***Learning curve (LC).*** This criterion evaluates how easy the learning process is.

Mobility was not considered as a criterion because CoBASA does not need mobile agents.

As an example the priorities obtained for each of the criteria defined above are shown, using the AHP process:

OS	AM	ACL	LLC	CP	FIPA	CI	M&D	IC	UF	EAC	S	NU	D	LC
0.11	0.10	0.10	0.09	0.08	0.08	0.07	0.06	0.06	0.05	0.05	0.04	0.04	0.04	0.04

The vectors used to compare each decision alternative against each criterion are not shown due to lack of space. After weighting these results with the ones obtained from the criteria priorities defined above the decision alternatives are then prioritised. The decision alternative that scored the highest priority was JADE. This choice proved to be very effective, and later on, it was also noticed that a large number of researchers with whom the author has been working, are using JADE.

JADE. JADE is an open source FIPA compliant Java based software framework for the implementation of multiagent systems. It simplifies the implementation of agent communities by offering runtime and agent programming libraries, as well as tools to manage platform execution and monitoring and debugging activities. These supporting tools are themselves FIPA agents. JADE is distributed under the terms of the LGPL (Lesser General Public License – Version 2).

A description of the JADE main functionalities can be found in (F. Bellifemine, A. Poggi, & G. Rimassa, 2001; Fabio Bellifemine, Agostino Poggi, & Giovanni Rimassa, 2001; Poggi, Rimassa, &

Turci, 2002). JADE offers simultaneously middleware for FIPA compliant multiagent systems, supporting application agents whenever they need to exploit some feature covered by the FIPA standard (message passing, agent life cycle, etc), and a Java framework for developing FIPA compliant agent applications, making FIPA standard assets available to the programmer through Java object-oriented abstractions. Related to these considerations the following aspects of JADE that will be considered:

1. The Middleware[30] required to support the community of agents. This includes the agent management tool.
2. The agent model.
3. Agent communication.
4. Ontology support.
5. Debugging tools.

The supporting infrastructure middleware offered by JADE, which is known as the JADE agent platform, is composed of the following items:

1. The communication channel used for messaging service (ACC – Agent Communication Channel). The ACC provides the path for basic contact between agents inside and outside the platform.
2. The Agent Management system – AMS, which is an agent that exerts supervisory control over access to and use of the platform, and is responsible for authentication of resident agents and control of registrations.
3. The Directory Facilitator – DF, which is itself an agent that provides a yellow pages service to the agent platform.
4. The Remote Monitoring Agent – RMA (Figure 5-2), which is the graphical user interface for the remote management, monitoring and controlling of the status of agents.

The general management console for a JADE agent platform (RMA) (Figure 5-2) acquires the information about the platform and executes the GUI commands to modify the status of the platform (creating new agents, shutting down containers, etc) through the AMS.

The agent platform can be split between several hosts (provided that there is no firewall between them). Agents are implemented as one Java thread and Java events are used for effective and light-weight communication between agents on the same host. Parallel tasks can be still executed by one agent, and JADE schedules these tasks in a more efficient (and even simpler for the skilled programmer) way than the Java Virtual Machine does for threads. Several Java Virtual Machines (VM), called containers in JADE, can coexist in the same agent platform even though they are not

[30] The term middleware is also used by some researchers (Poggi et al., 2002) to indicate that a software layer must be made available to simplify agent creation.

running in the same host as the RMA agent. This means that a RMA can be used to manage a set of VMs distributed across various hosts. Each container provides a complete run time environment for agent execution and allows several agents to concurrently execute on the same host. The DF, AMS, and RMA agents coexist under the same container (main-container), as it is shown in Figure 5-2.

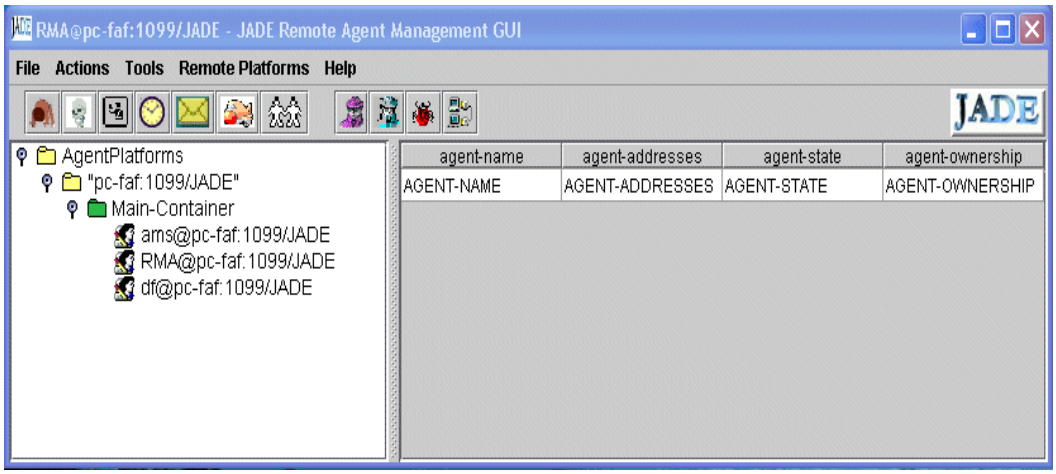

Figure 5-2 - JADE RMA agent

Since JADE platforms can be connected to others, either JADE or non-JADE, it is possible to create a complex network of agent communities, using different agent development frameworks. Each platform houses and controls its local community of distributed agents, which can communicate with other agents belonging to remote platforms through their own host platforms (Figure 5-3). As it could be expected, the communication between agents housed in the same platform is much less time consuming than between agents housed in different platforms. Consequently, it is possible to structure the agents according to how much they need to communicate with each other. Agents that need to interact more can be grouped under the same platform for efficiency reasons (if that is consistent with the physical distribution of the system).

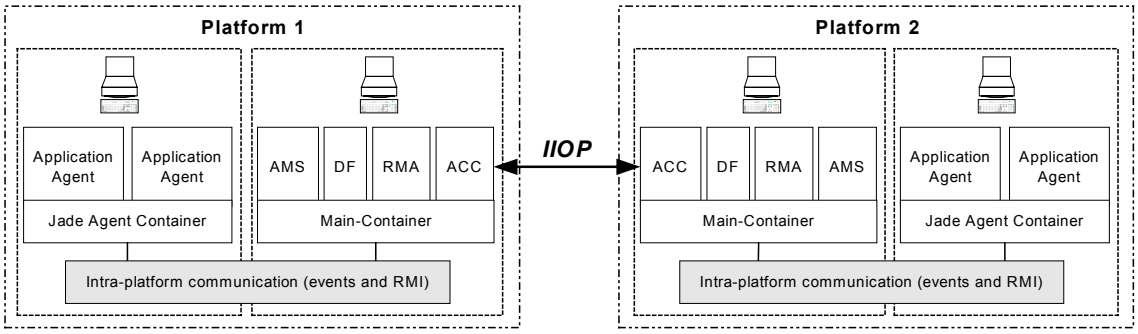

Figure 5-3 - Inter and intra platform communication

The type of communication mechanism used by JADE to establish communication between two agents varies according to the location of the agents:

1. Agents sharing the same container – A simple event message is used.
2. Agents sharing the same platform but in different containers – The communication is assured by Java RMI.
3. Agents living in different platforms – A direct call is performed towards remote ACC using multiple Message Transport Protocols (MTPs). The default MTP is the standard Internet Inter-ORB – IIOP[31] protocol and OMG IDL interface, according to the FIPA standard. The HTTP protocol is also supported by JADE for inter platform communication.

A major issue is the execution model for an agent platform, which both affects performance and the programming style. The JADE execution model is based on computational concurrency, which is mandatory for agents that need to be autonomous and have social behaviour. Autonomy implies that each agent should be executed within at least one independent thread of control, while sociality pushes towards multiple threads to be able to support simultaneous conversations. An agent can be engaged in multiple simultaneous conversations, besides carrying on other activities not involving message exchanges.

JADE uses the **Behaviour** abstraction to model the tasks that an agent is able to perform and agents instantiate their behaviours according to the needs and capabilities. From a concurrent programming point of view an agent is an active object, holding inside a thread of control. The various agent tasks or duties are represented by a collection of behaviours, whose execution is scheduled according to a certain order that can be defined by the programmer.

JADE agents instead of executing their behaviours as pre-emptive multithreads, and in order to keep small the number of threads required to run the agent platform[32], use a non pre-emptive approach. With this strategy all behaviours are run from a single stack frame following non pre-emptive cooperative scheduling. Each executing behaviour cannot be pre-empted by the others until it finishes. Therefore, JADE uses a *thread-per-agent* concurrency model instead of *thread-per-behaviour* one. Non pre-emptive cooperative scheduling implies that the execution context of behaviours is not saved. This means that each behaviour is executed from the beginning every time it is scheduled for execution. Thus, the behaviour state, required to be saved across multiple executions, must be stored into behaviour instance variables. This strategy affects the programming style. In fact, programming JADE behaviours has some similarities with PLC programming because of some likeness in their execution models, in which case JADE behaviours are compared to PLC rungs. The next PLC rung is executed only after the current one is executed and they are not pre-empted. After executing the entire set of rungs the PLC starts executing the first one, from its beginning.

To implement a JADE agent it is necessary to extend the Agent class and implement the agent-specific tasks by writing one or more Behaviour classes. The Agent class represents a common

[31] IIOP (Internet Inter-ORB Protocol) is a protocol that makes it possible for distributed programs written in different programming languages to communicate over the Internet. IIOP is a critical part of a standard for software interoperability, the Common Object Request Broker Architecture (CORBA).

[32] In the most popular current operating systems the number of threads that can be run effectively is not very high.

superclass for user-defined agents. Therefore, from the point of view of the programmer, a JADE agent is simply a Java class that extends the base Agent class. It allows inheriting a basic hidden behaviour (that deals with all agent platform tasks, such as registration, configuration, remote management, etc), and a basic set of methods that can be called to implement the application tasks of the agent (e.g. send/receive ACL messages, use standard interaction protocols, register with several domains, etc). Moreover, user agents inherit from their *Agent* superclass some methods to manage agent behaviours.

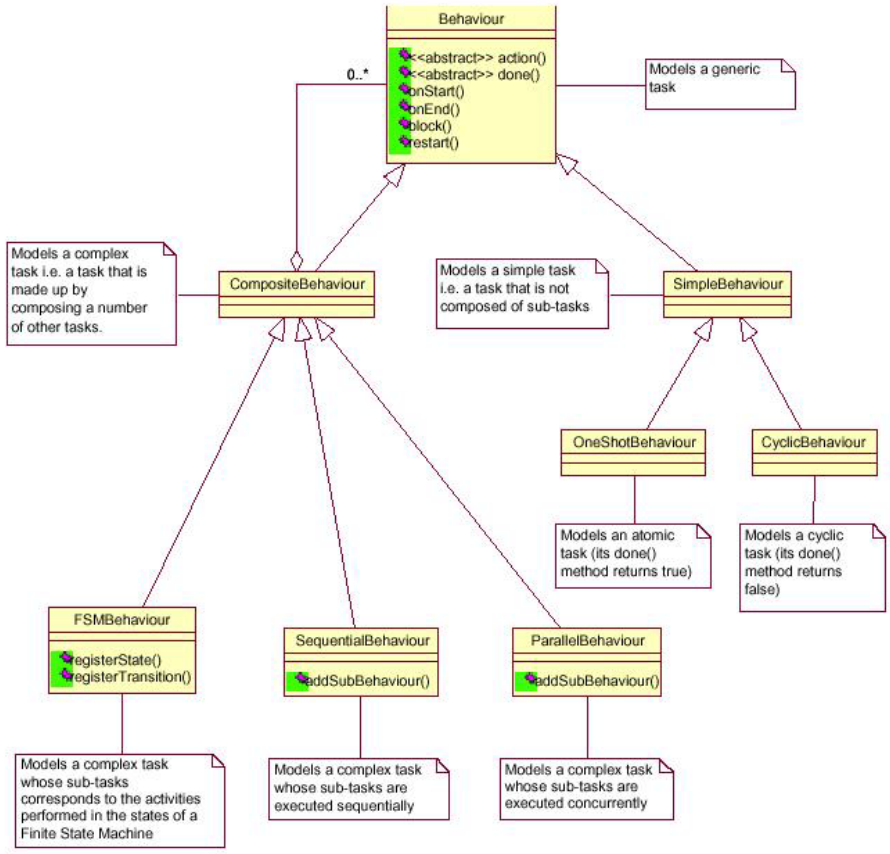

Figure 5-4 - JADE behaviours UML class diagram. (Bellifemine, Caire, Trucco, & Rimassa, 2003)

JADE also includes some ready to use behaviours for the most common tasks in agent programming, such as sending and receiving messages and structuring complex tasks as aggregations of simpler ones. The Figure 5-4 shows the different types of behaviours supported by JADE, which are depicted as an UML diagram because behaviours are implemented as a hierarchy of classes. Please note that the behaviours for sending and receiving messages as well as the ones to implement FIPA standard protocols (complex communication interaction) are not shown although supplied by JADE.

Besides communication support at physical level (how messages are exchanged), which were discussed when JADE as middleware was introduced, JADE also provides syntactical and semantic communication support (message content). JADE supplies mechanisms to help the programmer in constructing FIPA ACL messages as well as FIPA interaction protocols. An ACL message is modelled

as a Java object that have methods to fill and get the various attributes that compose an ACL message. Also defined within this class is the entire message performatives as defined by FIPA (REQUEST, INFORM, …).

To facilitate message reply, which, according to FIPA, must be formed taking into account a set of well formed rules such as setting the appropriate value for the attributes *in-reply-to*, using the same *conversation-id*, etc., the method *createReply()* is defined in the class that defines the ACL message.

Different types of primitives are also included to facilitate the implementation of content languages other than SL, which is the default content language defined by FIPA for ACL messages.

A set of ready-made behaviours is supplied by JADE to implement FIPA based protocols. The structure of protocols like FIPA-Request, FIPA-query, FIPA-propose, FIPA-Request-When, FIPA-recruiting, FIPA-brokering is similar. In these protocols the initiator sends a single message (i.e. it performs a single communicative act) within the scope of an interaction protocol in order to verify if the RE (Rational Effect) of the communicative act has been achieved or not. The initiator sends a message (in general it performs a communicative act). The responder can then reply by sending a `NOT-UNDERSTOOD`, or a `REFUSE` to achieve the rational effect of the communicative act, or also an `AGREE` message to communicate the agreement to perform (possibly in the future) the communicative act. This first category of reply messages has been identified here as a response. The responder performs the action and, finally, must respond with an `INFORM` of the result of the action (eventually just that the action has been done), or with a `FAILURE` if anything went wrong. This second category of reply messages has been identified here as a result notification. The behaviour *AchieveREInitiator* implements the initiator role in all these protocols, and it works both for 1:1 and 1:N conversation. The *AchieveREResponder* implements the responder role.

JADE supports ontologies based on a model of the content language, which is able to describe objects, and constructs that represent identifiable entities. It provides support for different content languages by providing objects to transform a content string into structures that can then be locally processed, and vice-versa. When the CoBASA prototype started to be developed, the ontology support offered by JADE consisted only of providing a weak ontology model represented in Java, and whose abstraction model was inappropriate for the CoBASA requirements. This was the reason why the prototype also used Protégé-2000. This has changed and the new JADE versions (JADE version 3.0b1 was released on 19th March 2003) ontology support has improved. Now better abstraction models such as predicates, modal logic, first order logic, and even deontic logic are included in JADE. The content language used in CoBASA messages is SL. The syntax analysis offered by JADE is used, however, all the semantic level was implemented separately, using new functions and integration with Protégé-2000, for the reason explained in the next section (5.2.2).

Three graphical tools are included within JADE to support the debugging phase, usually quite complex in distributed systems. These tools are: the *Dummy Agent*, the *Sniffer Agent*, and the *Debugger Agent*. The Dummy Agent is useful to edit, compose, and send ACL messages to agents, and to receive and view messages from agents. It is quite useful because it permits inspecting message exchanges among agents thoroughly without needing to develop a new agent. The dummy agent facilitates validation of an agent interface before its integration into the MAS and facilitates interrogative testing in the event that an agent is failing. The Sniffer Agent tracks messages exchanged in a JADE agent platform, which are displayed using a notation similar to UML Sequence Diagrams. The user can view, save, and load every message track for later analysis. The Debugger Agent traces the internal execution of each agent. It permits access to the input message queue, the agent life cycle state, and the behaviours running.

Final remarks. After using JADE intensively, the global impression about it is positive and it is still considered that JADE is a suitable and agreeable tool to develop systems such as the current version of the CoBASA prototype. The positive aspects that were promised when JADE was chosen proved to be true, such as easy to use, good set of supporting tools, adequate API, adequate agent communication functionalities, and very good support. However, despite the overall good impression about JADE there are some points that need to be pointed out as JADE weaknesses.

The first point is about the agent model. Although it was possible to implement successful running CoBASA agents using the JADE agent model, it can become less efficient when used to implement more advanced versions of the CoBASA prototype. This must be clarified because it might be inferred that the agent model was not adequate for the current state of the prototype. On the contrary, the simplicity of the agent model was one big advantage for the implementation of the current prototype. However, to develop newer CoBASA versions, including new facilities such as a broker agent with more advanced functionalities (automatic creation of coalitions) a more advanced agent model might be needed. This objective would be attained in an easier fashion if JADE could naturally support, for instance, the BDI framework. Although it is possible to implement JADE agents following the BDI model, its implementation is not so easy because the agent model behind JADE does not naturally support it. What is being defended here is that if JADE could support agent models closer to the Artificial Intelligence area this would be of tremendous help as the expressiveness of these models are, in general, more powerful because their abstraction level is higher.

The second point is about ontology support, which should be better developed in JADE. The support for knowledge-based models based on frames, first order logic, deontic logic, or temporal logic, is advantageous because this would permit an easy connection between the content language and the concepts that model each agent. The expressive power of the ontology model included in the first version used to implement the prototype was very weak. Despite late improvements, the newer JADE versions still have an ontology model far from being the most adequate for CoBASA needs.

The last point is about fault-tolerance. The existence of a front-end container that includes the AMS is recognised as a weak point in the architecture. In fact, a fault in this agent would invalidate some communication between agents. JADE developers are working at this point trying to improve this flaw.

5.2.2 Knowledge modelling support

As mentioned above, a knowledge modelling tool to provide knowledge representation within JADE in order to have frame-based semantics at programming level was needed. In specific, the frame-based paradigm is quite appropriate for the modelling of the concepts needed within the CoBASA agents. However, a tool was needed that could do more than just support frame-based models, because models were required to be called from inside the JADE agents, which means that some kind of connection mechanism was needed between the Java language and the tool. This connection should be made, preferably, by integrating the code of the tool within the code of the agent, rather than a connection using TCP/IP interprocess mechanism, for the sake of efficiency. Furthermore, the tool should provide a user-friendly environment that could facilitate the creation and development of the required models, and completely separate from agent programming. With this interface the details of the models are hidden behind it. On the other hand, a non-specialist can now create the knowledge models as well.

Therefore, the main requirements that should be offered by a tool to support the CoBASA needs in terms of knowledge modelling are:

- It should support knowledge-based models, preferably based on the frame paradigm.
- It should be able to be called from within the Java language environment.
- Easy integration.
- It should have a user-friendly environment for the development of the models, this being completely separated from its application engine. This implies a clear separation between the development and operational (use) phases of the knowledge models supporting the agents and, consequently, it requires an interface independent of the knowledge model.

Tool Search. The most difficult requirement when searching for a suitable tool was integration. Many interesting tools to support knowledge modelling and ontologies in particular suffer, in general, of poor integration capabilities. Loom (*Loom*, 2002) is one these tools that were considered to be integrated. It has an adequate frame based knowledge model, and nice functionalities. However, it is developed in Lisp and it is a standalone application that is not easily integrated with other applications. Loom was used in the beginning of this work but, later on, it was abandoned because of the enormous difficulties in integrating it with the JADE environment.

Ontolingua was also considered but in that case it was necessary to access it through the Internet, and it was not feasible to have a system that every time it needed to access a model a request through the Internet needed to be made. This was impractical for speed reasons.

Protégé-2000 was chosen among the available tools because it was the one that better answered the requirements defined above.

Protégé-2000. An attractive feature of Protégé-2000 is its knowledge model, which is based on the Open Knowledge Base Connectivity protocol (OKBC[33]) (Chaudhri, Farquhar, Fikes, Karp, & Rice, 1998), which in turn is based on the generic frame paradigm (Fikes & Kehler, 1985). Details about the Protégé knowledge model can be found in (Noy, Fergerson, & Musen, 2000). Protégé-2000 is simultaneously a tool to construct domain ontologies and a library, which other applications can use to access and display knowledge bases.

By providing an interface application independent of the knowledge model, Protégé-2000 enables such a clear separation between the development and the use phases of the models. The models are therefore developed using the interface and, hence, when the agents need to be run, only the API is exploited.

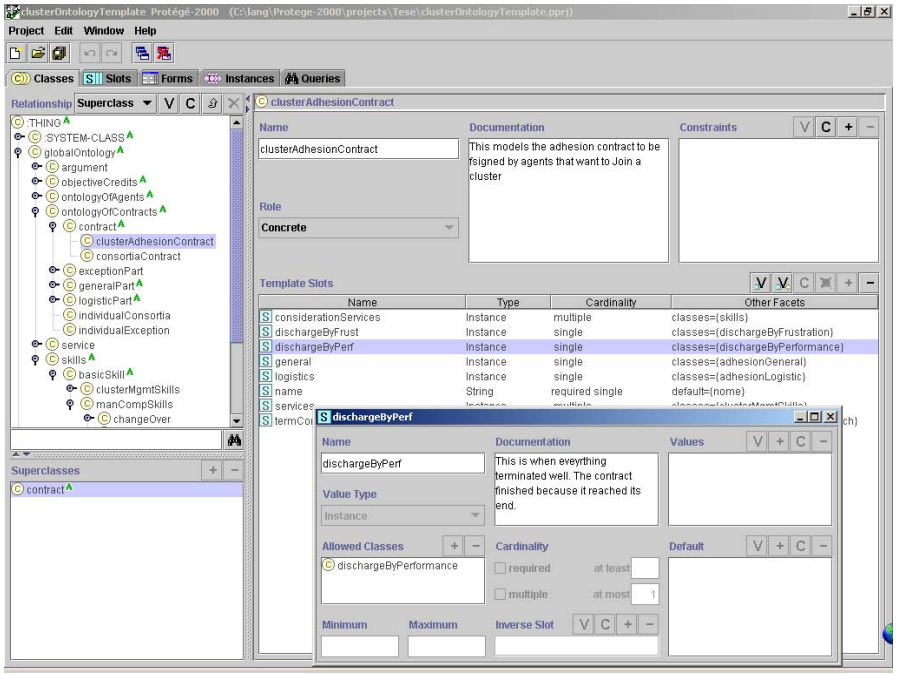

Figure 5-5 - The user interface of Protégé-2000

Figure 5-5 shows the user interface of Protégé-2000 in which the class or frame that represents a cluster adhesion contract is shown (right part of the figure). It also shows the window (lower right part) that characterises one of the slots used in the class: *dischargeByPerf*.

[33] The OKBC is an avolution of the Generic Frame Protocol (Karp, Myers, & Gruber, 1995)

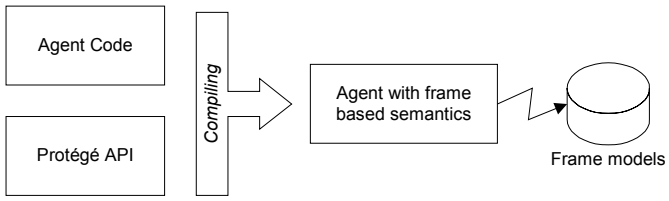

Figure 5-6 - Connection JADE with Protégé-2000

In Figure 5-6 it is shown how the tool is connected to the CoBASA Java agents. The only requirement to enable the connection of Protégé-2000 to the agent environment is the inclusion of the *protege.jar* file, which contains all the java classes required to operate the tool, in the Java compiler. The API of Protégé-2000 offers a set of objects that support the frame based knowledge model. The most important object to be understood is the object *KnowledgeBase*, which must be instantiated by any agent that wants to create or manage a knowledge base. The methods offered by this object give an idea of what is possible to do with this API:

- To create elements of the knowledge model – *createCls, createSlot, createInstance, ...*
- To delete elements – deleteCls, deleteSlot, deleteInstance, …
- To obtain information – getClses, getSlots, getInstances, …

From this object (*KnowledgeBase*) it is then possible to obtain other objects that represent classes or frames, slots, instances, etc. and, from these, it is now possible to access the methods that execute the functionality expected in a frame based knowledge model over slots, frames, instances, etc.

The advantages of using Protégé-2000 in the context of this thesis, over other analysed tools are:

- It is open source and Java based.
- It is easily accessible from external Java applications, through its API.
- The stand-alone application is very user-friendly and with a short period of learning.
- Easy to be installed and operated.
- Adequate knowledge model.
- Its behaviour can easily be extended and changed through *plugins*.
- Simple and easy connection with Jess, which is a Java rule based engine and scripting environment based on CLIPS.
- The model can be saved in any database that has a JDBC 1.0 driver (Oracle, MySQL, Microsoft SQL Server, and Microsoft Access).
- Good support. The e-mail list is very active and questions are swiftly answered.

Final remarks. Each CoBASA agent has its own knowledge base or ontology, which is configured during the creation of the agent. This knowledge base acts simultaneously as an ontology, since the concepts that needed to be understood and perceived by the agents are included, and as a pure knowledge base, in which the models required for the agent processing are instantiated and managed

during the execution of the agent in a very similar way to how traditional programs operate with their variables. The relation between the basic code of the agent and the Protégé-2000 code is shown in Figure 5-7.

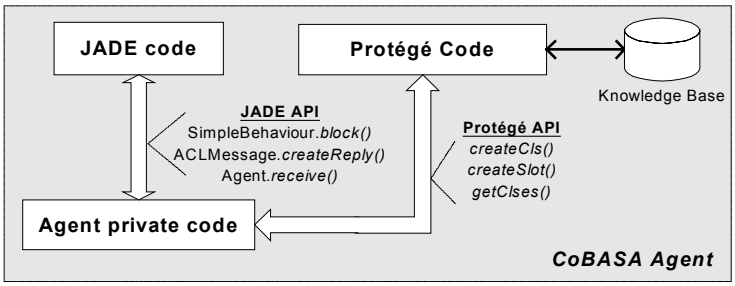

Figure 5-7 - Integration of Protégé-2000 in CoBASA agents

5.3 PROTOTYPE DESCRIPTION

This section details the implementation of the CoBASA prototype. For each agent type that compose the architecture: the Cluster Manager (CMgA), the Broker (BA), and the Generic Agent (GA), a brief description of the implementation details is made. To avoid repetition, the agents' common implementation aspects are described in a common subsection.

5.3.1 Common implementation details

The common implementation details worth considering before describing each agent are the behaviour mechanism, and the ontology.

5.3.1.1 Behaviours

As was mentioned before, when JADE was analysed, behaviours are the basic agent model mechanism. They are finite state machines, keeping their whole state in their instance variables. JADE also supports a compositional technique to build more complex behaviours out of simpler ones.
Starting from the basic class Behaviour, a class hierarchy is defined in the *jade.core.behaviour* package of the JADE framework.

Class Behaviour. This abstract class provides an abstract base class for modelling agent tasks, and it sets the basis for behaviour scheduling as it allows for state transitions (i.e. starting, blocking and restarting a Java behaviour object). The *block()* method allows to block a behaviour object until some event happens (typically, until a message arrives). Summarizing, a blocked behaviour can resume execution when one of the following three conditions occurs:

1. An ACL message is received by the agent this behaviour belongs to.
2. A timeout associated with this behaviour by a previous *block()* call expires.
3. The *restart()* method is explicitly called on this behaviour.

The *Behaviour* class also provides two placeholder methods, named *onStart()* and *onEnd()*. These methods can be overridden by user-defined subclasses when some actions are to be executed before and after running behaviour execution.

The prototype uses the various types of behaviours offered by JADE, which are subclasses of the basic *Behaviour* class.

5.3.1.2 Ontology

In the developed prototype Protégé-2000 was used to describe the concepts and supporting types needed by the CoBASA agents.

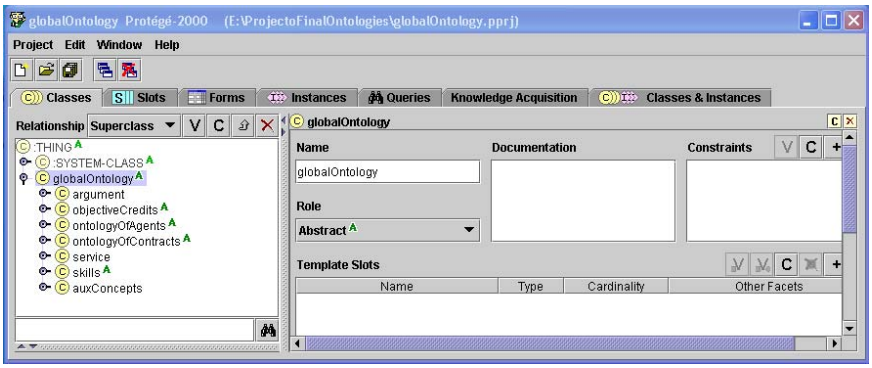

Figure 5-8 - The global ontology main concepts

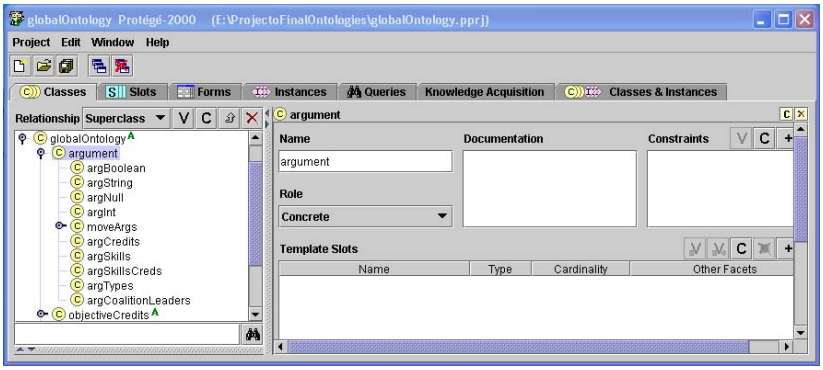

Figure 5-9 - The argument frame

The generic ontology is divided in two files: *globalOntology* and *global&CommandOntology*. The first one models the main concepts behind the agent's architecture, while the second models the content of each message per agent type. The basic concepts in the globalOntology are: *argument*, *objectiveCredits*, *ontologyOfAgents*, *ontologyOfContracts*, *service*, *skills* and *auxConcepts* (Figure 5-8).

Argument models the arguments that are going to be present in the messages to be exchanged. Different argument concepts were created to model the different types of arguments that compose the messages (Figure 5-9).

The ***ObjectiveCredits*** models the qualification of each manufacturing agent. It is organised as a hierarchy of credits to accommodate future extensions. The slots belonging to the frame *objectiveCredits* represent the different qualification aspects common to all agents (Figure 5-10) (Table 5-2).

Frame *objectiveCredits*	
Slots	**Description**
consortiaAverageScore	The average score awarded by the agent when working in a consortium.
badBehavioursPenalties	Represents the number of bad behaviours an agent had.
dateOfBirth	The date when the agent was created.
consortiaParticipation	The number of times the agent had participated in a consortium.
clusterAverageScore	The average score awarded by the agent when working in a cluster.
clusterParticipation	The number of times that one agent had participated in a cluster.

Table 5-2 – Frame *objectiveCredits*

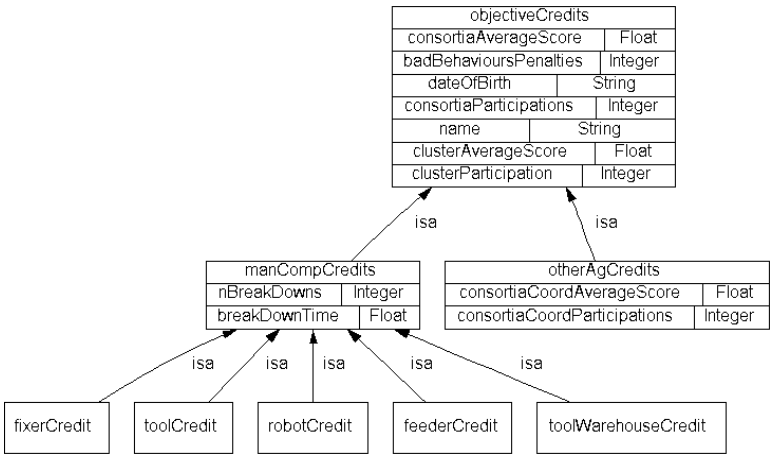

Figure 5-10 - The objectiveCredits hierarchy

The credits related to manufacturing agents (MRAs) are modelled by the frame *manCompCredits*. The credits for manufacturing agents include the generic characteristics defined in *objectiveCredits* and two specific ones (Table 5-3).

Frame *manCompCredits*	
Slots	**Description**
nBreakDowns	The number of breakdowns of the equipment.
breakDownTime	The equipment total breakdown time.

Table 5-3 – Frame *objectiveCredits*

Coordinator agents (CAs) have their credits modelled by the *otherAgCredits* concept (Table 5-4).

Frame *otherAgCredits*	
Slots	**Description**
consortiaCoordAverageScore	The average score awarded by the coordinator agent when working.
consortiaCoordParticipations	The number of successful participations as a coordinator.

Table 5-4 – Frame *otherAgCredits*

The concept of agent is represented by the hierarchy of agents (Figure 5-11), which is composed of the concepts *coordAgent*, which models coordinator agents (CAs), and *manCompAgent*, which models manufacturing agents (MRAs).

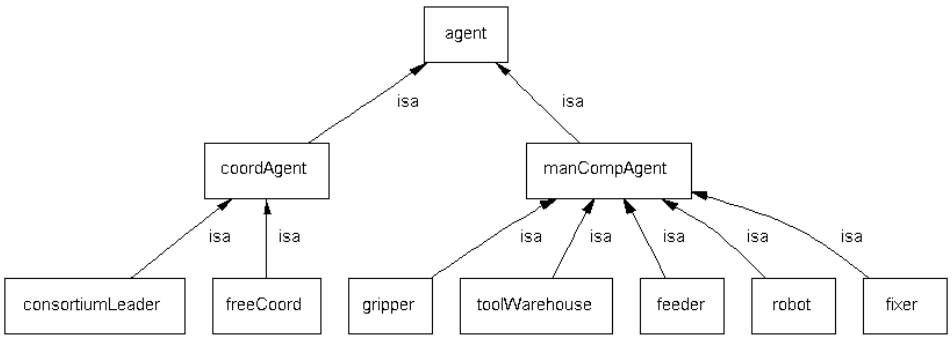

Figure 5-11 - Hierarchy of agents

The concepts *gripper*, *toolWarehouse*, *feeder*, *robot*, and *fixer* represent specific manufacturing equipment, while *consortiumLeader*, and *freeCoord* represent coordinator agents.

The slots that characterise the concept robot are shown in Figure 5-12 as an example. Please note that the slots that are shown in the concepts *manCompAgent* and *agent* are inherited by the concept *robot*.

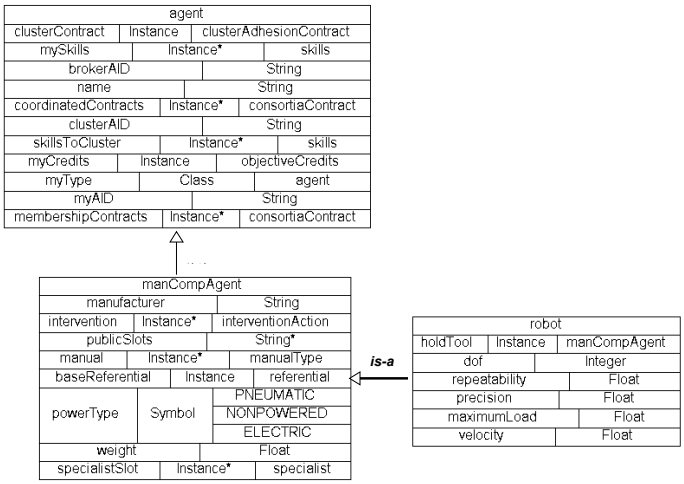

Figure 5-12 - The slots that describe the agent hierarchy

Frame *robot*	
Slots	**Description**
clusterContract	Contains the contracts established by the agent with a cluster manager.
membershipContracts	The contracts that regulate coalitions where the agent is a member.
coordinatedContracts	The contracts that regulate coalitions where the agent is the coordinator.
myAID	The agent identifier address in the multiagent architecture.
brokerAID	The broker agent identifier address in the multiagent architecture.
clusterAID	The cluster manager agent identifier address in the multiagent architecture.
mySkills	All the skills possessed by the agent. When an agent is a coordinator this slot is filled with the skills offered by the various members that compose the coalition it coordinates in addition to those that were created as compositions of the offered ones (complex skills)
skillsToCluster	The skills that were offered to the cluster. *mySkills* û *skillsToCluster*
interventionAction *specialistSlot* *manual*	They include information for maintenance purposes. The information stored here can be used to support future applications to help maintenance of the equipment as well as reengineering activities.
powerType *weight* *dof* *repeatability* *precision* *maximumLoad* *velocity*	Describe physical properties of the robot, which are important when manufacturing agents are being chosen to participate in coalitions. Each manufacturing agent has its own set of physical properties.

Table 5-5 – Frame *robot*

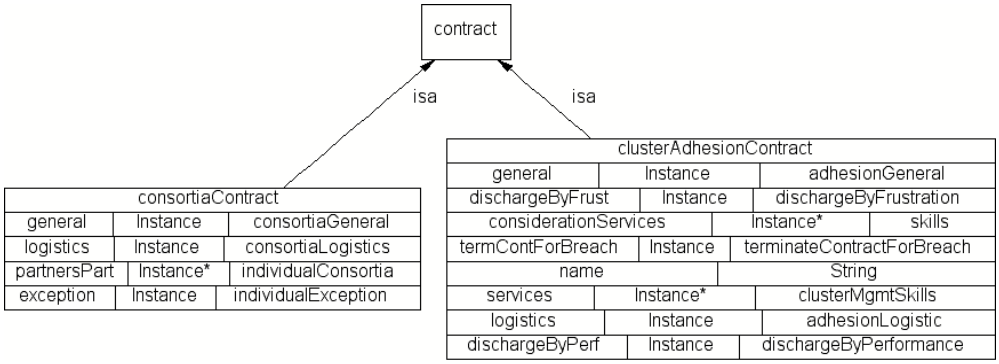

Figure 5-13 - Hierarchy for contracts

The ***ontologyOfContracts*** models the contracts established between agents. In fact, two kinds of contracts are concerned here: the cluster adhesion contract, which is represented by the concept *clusterAdhesionContract*, and the consortia contract, which is modelled by the concept *consortiaContract*. The slots that compose these two types of contracts were described when the CoBASA architecture was introduced (Chapter 4).

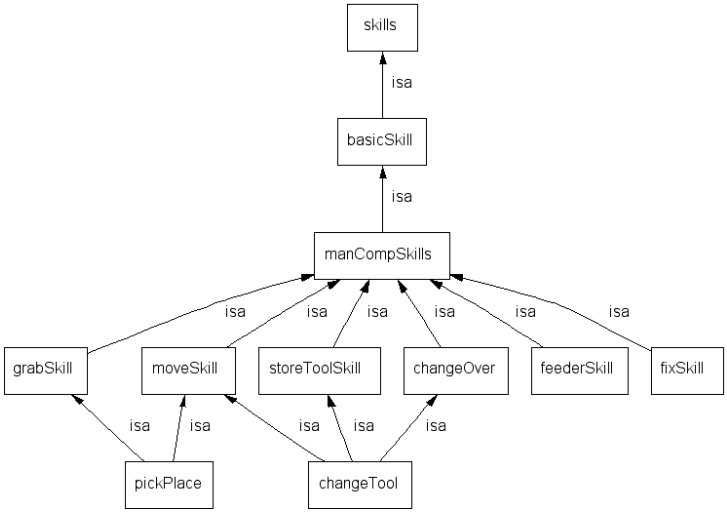

Figure 5-14 - Hierarchy of Skills

The hierarchy of ***skills*** includes all the skills supported by CoBASA. For instance, the skills of a robot are definitely different from the skills of a gripper and even two robots might have distinct skills such as the capability of exchanging the tool and moving their tips. Two types of skills can be found in CoBASA: the skills related to the operations of manufacturing equipment, which are represented by the concept *manCompSkills*, and cluster related skills, i.e. the skills that the cluster manager agent can offer to the agents that join the cluster.

The basic skills considered for this prototype are:

- ***changeover*** – The robot ability to exchange tools
- ***feedeSkillr*** – the feeder ability to supply raw material

- ***fixSkill*** – the holder ability to fixture a piece
- ***grabSkill*** – the gripper ability to grasp and/or ungrasp an object
- ***moveSkill*** – the robot ability to make movements from one place to another
- ***storeToolSkill*** – the toolwarehouse ability to store and pick robot tools.

The complex skills represented in Figure 5-14 are *pickPlace* and *changeTool*, which are compositions of basic skills. The slots that each skill contains are the attributes that characterise the skill. Further information about skills was provided when the CoBASA architecture was discussed (Chapter 4).

5.3.2 Communications

The interaction protocols used were the ***FIPA-Request*** and ***FIPA-query*** (Figure 5-15). These protocols are described in (Bellifemine et al., 2003; FIPA, 2002a, 2002b). FIPA protocols assume the existence of an initiator (the agent that initiates the conversation) and one or more responders (the agents which have been addressed by the conversation). JADE provides specific behaviours to address these two roles. The behaviour *achieveREInitiator* implements the initiator role, while *achiveREResponder* implements the responder or participant role.

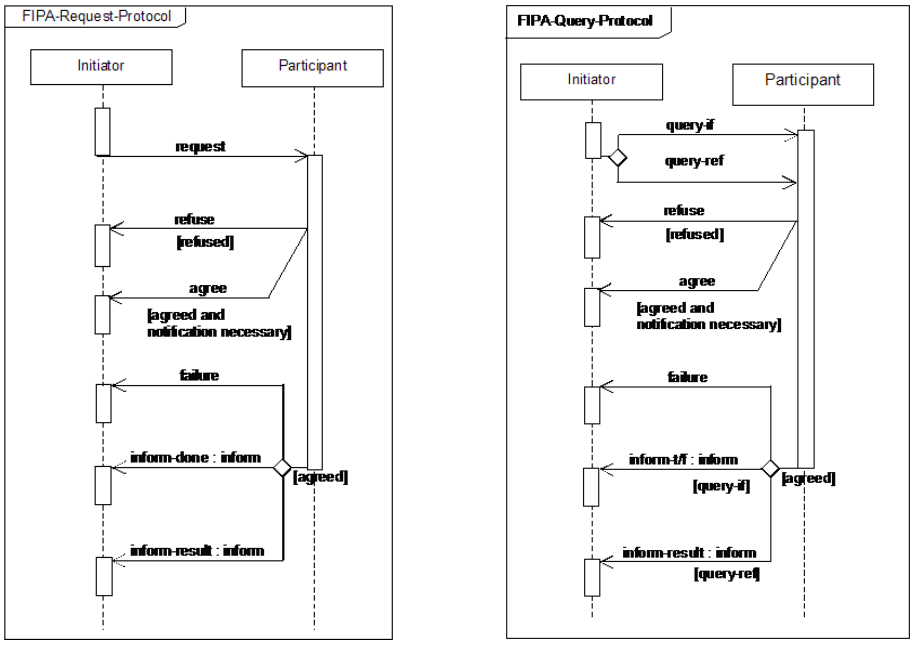

Figure 5-15 - FIPA REQUEST and QUERY protocols (FIPA, 2002a, 2002b).

The ***achieveREInitiator*** behaviour is represented by a class that contains the methods *handleAgree()*, *handleRefuse()*, *handleNotUnderstood()*, *handleInform()*, and *handleFailure()*, which are executed according to the type of answer sent by the responder (Figure 5-16). The behaviour *achieveREResponder* is also represented by a class whose methods *prepareResponse(),* and

prepareResultNotification() are executed when the responder receives a REQUEST message. The second method is only called if the first response was an `AGREE` message.

The process to create and receive the content of a message is important because its content is transmitted as a sequence of bytes (a string of characters). Furthermore, the agents need to exchange concepts that are represented using the Protégé format (frame). Therefore, to send a message it is necessary to convert it from the frame format to a string and vice-versa when a message is received.

The knowledge base contains all the necessary classes, slots, and values to create the content of messages. The frame that represents the hierarchy of messages is *cmdOntology* and it can be found in the *global&CmdOntology* file.

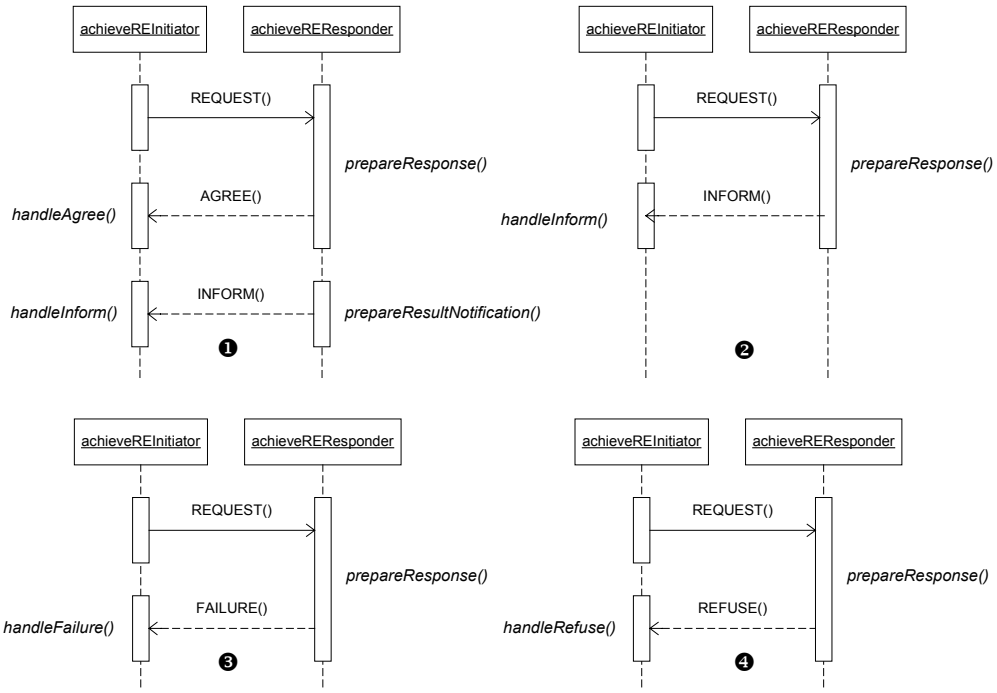

Figure 5-16 – Interaction between the achieveREInitiator and achieveREResponder behaviours

To send a message, it is required to first instantiate the respective frame and, afterwards the slots that compose the created instance are filled in with the values required to be exchanged in the message. Next, the instance corresponding to the message needs to be transformed into a string in order to be attached to the content attribute of the FIPA-ACL message. This is done using a parser developed specifically for CoBASA. This parser is a Java class with methods able to convert a frame into a String. The message is constructed by putting, first, the name of the class followed by the name of the first attribute (slot) and finally the slot value. If the class has more than one attribute, at the end of the last slot value, the name of the next slot and its value will be introduced, and so on. For instance, considering that the *Person* class has the attributes: *name*, *idDocument*, *birthdate*, *gender*, *profession* and *nationality*, the serialized object will result in the String illustrated below.

```
((Person name <String> idDocument <String> birthdate <String> gender
  <Symbol> profession <String> nationality <String>))
```

5.3.3 Cluster Manager – CMgA

5.3.3.1 Behaviours

The cluster manager is composed of the following four main behaviours: *initialBehaviour*, *clusterContractTermB*, *clusterContractNeg*, and *executeServicesB*.

InitialBehaviour. This is the first behaviour to be executed and it is implemented as a JADE *simpleBehaviour*. It registers the agent in the directory of the JADE platform.

ClusterContractNeg. This behaviour negotiates with the candidate agents that want to participate in the cluster manager. The *clusterContractNeg* behaviour is composed of four sub behaviours: *clusterAcceptanceResponder*, *clusterQueryInitiator*, *clusterQueryAcceptanceInitiator*, and *clusterRequestContractBehaviour*. These behaviours implement the protocol *clusterContractNeg*, already defined in chapter 4. The process is initiated when a candidate agent sends a REQUEST ACLMessage. Each REQUEST message sent to the cluster is includes a user defined parameter *joinCluster* to be distinguishable from the various request messages received by the cluster[34]. This means that the behaviour *clusterAcceptanceResponder* is only initiated when a REQUEST message with user-defined parameter *joinCluster* is received.

The behaviour *clusterAcceptanceResponder* extends the JADE behaviour *AchiveREResponder* to implement the receiving part of the FIPA-REQUEST protocol, which is handled by the method *prepareResponse*.

The *clusterQueryInitiator* behaviour is a *simpleBehaviour* used to simply create the query message that will be sent by the *clusterQueryAcceptanceInitiator*, which is an *achieveREInitiator* behaviour type. This FIPA-QUERY is used to query the candidate about its credits.

The *clusterRequestContractBehaviour*, which extends the JADE behaviour *achieveREInitiator*, initiates the FIPA-REQUEST protocol to negotiate the contract terms. The cluster adhesion contract to be proposed to the candidate agent is created before this behaviour is executed.

Figure 5-17 shows how the sub-behaviours that compose *clusterContractNeg* interact.

[34] This approach has been used by all CoBASA agents because each CoBASA agent can receive different messages with the same performative, which need to be executed by different behaviours. To facilitate the selection the behaviours need to filter the right messages from the common buffer. This task is only possible by adding a user defined parameter to the message that is then used by the receiving behaviour to filter the message.

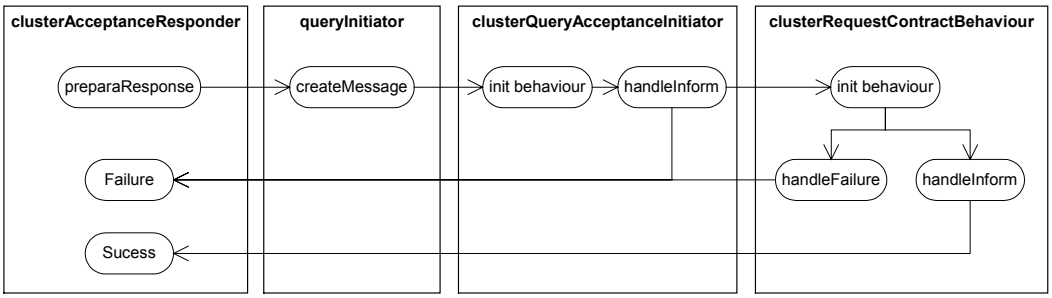

Figure 5-17 – Sub-behaviour interactions of clusterContractNeg

executeServices. This behaviour is executed when the CMgA receives a REQUEST message, in which the *userDefinedParameter* is set to "service", from a broker or a registered agent in the cluster. This is an *achieveREResponder* behaviour, in which the handle *prepareResponse()* is used to execute the services requested. The registered agents can request the following services: update credits, update coordinator credits, update skills, and update coordinator skills. The broker, on its turn, can ask for: types, skills, credits, and coalition leaders.

clusterContractTermB. This behaviour implements the various forms of contract termination and it is composed of three sub-behaviours: *dischargeByPerformanceInit*, *dischargeByFrustrationResp*, and *terminateContractForBreach*. *DischargeByPerformanceInit* and *terminateContractForBreach* implement initiator roles for the FIPA-REQUEST protocol. Consequently they are *achieveREInitiator* behaviours, while ***dischargeByFrustrationResp*** implements the participant role of the FIPA-REQUEST protocol, which makes it an *achieveREResponder* behaviour.

5.3.3.2 Supporting Ontology

The ontology that supports the cluster manager agent can be found in the *clusterOntology.pprj* file, which it inherits from the global ontology.

The *cluster* frame is the basic data structure to model the cluster (Figure 5-18). The slots that compose the frame that represent the cluster are indicated in Table 5-6.

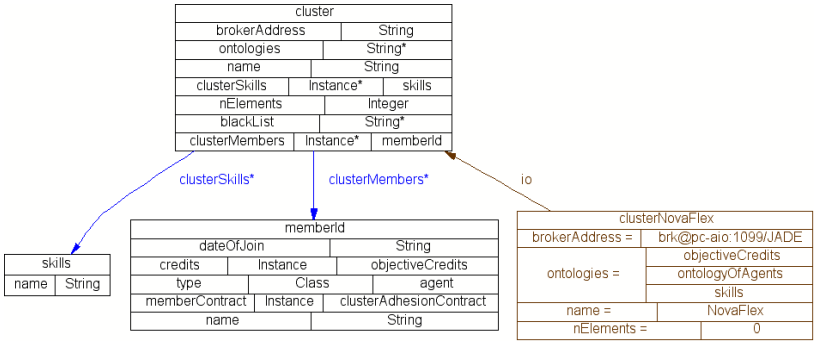

Figure 5-18 - The cluster knowledge representation

The *memberId* frame models a specific joined agent in the cluster and it has attributes such as: the *name* of the agent, its *type*, the date of joining (*dateOfJoin*), its *credits*, and the cluster adhesion contract that governs the adhesion of this member to the cluster (*memberContract*).

Frame *cluster*	
Slots	**Description**
brokerAddress	The broker address.
ontologies	The ontologies that candidate agents need to support.
clusterSkills	The set of all the skills brought to the cluster by its members.
nElements	The number of elements participating in the cluster.
blackList	A list containing the name of those agents with bad behaviour.
clusterMembers	A list containing a description (*memberId*) of each agent participating in the cluster.

Table 5-6 – Frame *cluster*

5.3.3.3 Communication

The messages received and sent by the CMgA are described according to its basic behaviours. Only the most relevant messages are shown because this would be too fastidious for the reader (Figure 5-19 and Figure 5-20). Many of the example messages contain real data taken from the case scenario.

From	To	Performative	Description
CMgA	GA	QUERY	Ask for credits
Content Message			
((askCredits what objectiveCredits))			

Figure 5-19 – ClusterContractNeg QUERY message

From	To	Performative	Description
CMgA	GA	REQUEST	The cluster adhesion contract is requested to be accepted. The contract is sent in the message.

Content Message

```
((acceptContract cont (clusterAdhesionContract services
 (SEQUENCE (clusterCapability interface (SEQUENCE clusterService)
 contributors "null" name "clusterCapability")(askForSkills interface
 (SEQUENCE clusterService) name  "askForSkills") (askForConsortia
 interface(SEQUENCE clusterService) name "askForConsortia") (askForMembers
 interface (SEQUENCE clusterService) name "askForMembers")) dischargeByFrust
 (dischargeByFrustration score 10.0 scoreParticipation 0) termContForBreach
 (terminateContractForBreach score 0.0 scoreParticipation 0) considerationServices
 (SEQUENCE null) logistics (adhesionLogistic timeToAnswer 45 numberOfTries 5
 contracteeAID "robot@jab:1099/JADE" clusterWorkingPlace "NovaFlex" contractorAID
 "agt@jab:1099/JADE") name "name" general (adhesionGeneral contractOntologies
 (SEQUENCE ontologyOfContracts ontologyOfAgents objectiveCredits) purpose
 "Adhesion contract for the NovaFlex" contractID "ref#3" contractor
 "clusterManager" contractStartingDate "14-07-2002 18:27" contractee "robot"
 contractPriority 50 name "name" duration 365)
 dischargeByPerf  (dischargeByPerformance score 50.0 scoreParticipation 1))))
```

Figure 5-20 – ClusterContractNeg REQUEST message

Although several messages requesting different services are exchanged by this behaviour, only one is shown (Figure 5-21).

From	To	Performative	Description
GA	CMgA	REQUEST	The GA requests the CMgA to update its skills.
Content Message			

```
((requestServCluster serviceName "updateSkills" arg1
  (argSkills argSkill
    (SEQUENCE (moveSkill maxVelocity 100.0 dof 6
               moveAgent boschSR840@jab:1099/JADE minVelocity 20.0 name "Bosch"
               movementType (SEQUENCE PTP LINEAR) interface (SEQUENCE
                             changeVelService  scaraMoveService moveJc5Service))
              (changeOver mechanismType SCHUNK name "Bosch"
               changeOverAgent boschSR840@jab:1099/JADE
               interface (SEQUENCE holdToolService releaseToolService)))) ))
```

Figure 5-21 – clusterContractTermB REQUEST message

The content of the messages exchanged to terminate the cluster contract (clusterContractTermB) are simple. Therefore, they are not shown.

5.3.3.4 User Interface

The cluster user interface is shown in Figure 5-22. The interface shows all the agent types that are registered. When an agent type is selected the addresses of all the agents that belong to that type are shown. In the same way, selecting an address implies that the corresponding skills of that agent are shown. The cluster interface also shows the agents in the black list.

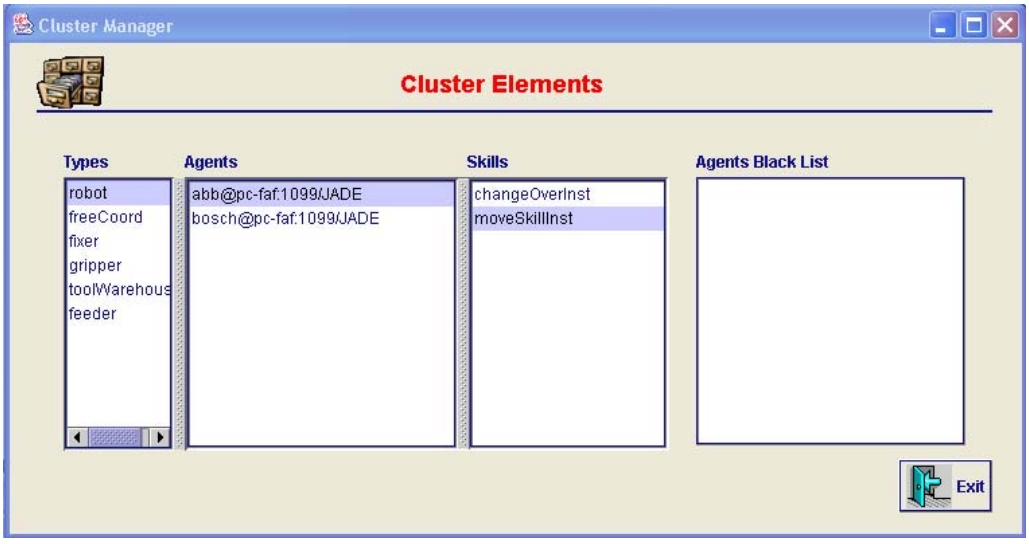

Figure 5-22 - Cluster user interface

5.3.4 Generic Agent – GA

5.3.4.1 Behaviours

The Generic Agent is composed of nine main behaviours that are responsible for its actions.

InitialBehaviour. In this behaviour the agent initialises its internal structures, connects to the Protégé file that contains its ontology, and prepares the other behaviours that will be executed during its operation.

JoinClusterBehaviour. As already mentioned in *clusterContractNeg* in the cluster Manager (CMgA) section, the purpose of this behaviour is to negotiate the adhesion contract with the CMgA, with the aim of participating in the cluster. This behaviour is divided into three sub-behaviours: *bRequestToJoinCluster*, *bQueryResponder*, and *bRequestResponder*. In order to initiate the process, the GA starts the *bRequestToJoinCluster* behaviour that sends a REQUEST ACLMessage to the CMgA requesting to participate in the cluster. The *bQueryResponder* behaviour is executed when the GA receives the request for credits message from the CMgA. As this behaviour extends the JADE AchiveREResponder behaviour, the credits are sent by the method *prepareResponse*. The *bRequestResponder* behaviour is executed when the GA receives the request from the CMgA to accept the cluster adhesion contract (CAC). As this behaviour extends the JADE AchiveREResponder behaviour, the contract acceptance or refusal is sent by the method *prepareResponse*. More details of this protocol can be seen in chapter 4.

ClusterTerminationB. This behaviour implements the three forms to terminate the contract with the CMgA. The three behaviours are respectively *clusterDischargeByFrustInit*, *clusterDischargeByPerformanceResp*, and *clusterTermContractForBreachResp*. The behaviours that implement these terminations extend the JADE *AchiveREResponder* with the exception of the *clusterDischageByFrustInit* that extends the *AchiveREInitiator*.

MembershipContractNegB. This behaviour is composed of several other sub-behaviours that implement the negotiation protocol with the BA agent for participation in a coalition.

MembershipContRenegB. This behaviour is composed of several other sub-behaviours that implement the negotiation protocol with the BA agent or other GA to renegotiate the participation of the agent in the coalition.

CoordContractNegB. Alike the *membershipContractNegB* the aim of this behaviour is to achieve a coalition contract, but here the GA performs the role of a coordinator. The only addition is that, after the consortium has been established, the coordinator agent gathers all the skills from the members

involved and generates the possible complex skills to be offered as the consortium skills. At the end, the GA also has to inform the CMgA about its new skills.

CoordContRenegB. This behaviour is executed when the GA is a coordinator and it is similar to the *membershipContRenegB*. Once again, at the end, the GA has to inform the CMgA about the skills earned from the members. If this coordinator is a member of some other consortia and the changes made in its consortium have modified its promises to the consortia where it is a member, it has to inform the corresponding coordinator.

membershipContTermB

This behaviour manages the coalition contract termination when the GA is acting as a member, that is, the contract termination is performed by the coordinator.

CoordContractsTermB. This behaviour looks out for the consortium termination, but on the coordinator side. Its sub-behaviours are: *genAgDischByPerfInitiator*, *genAgDischByFrustResp*, *genAggenAgBreachCoordResp*, and *genAgBreachCoordResponder*.

5.3.4.2 Supporting Ontology

The ontology that supports the generic agent (GA) can be found in the *genericAgentOntology.pprj* file, which inherits from the global ontology. In Figure 5-23 the frame concept that represents the agentified manufacturing component SCHUNK MPG 20 gripper is illustrated.

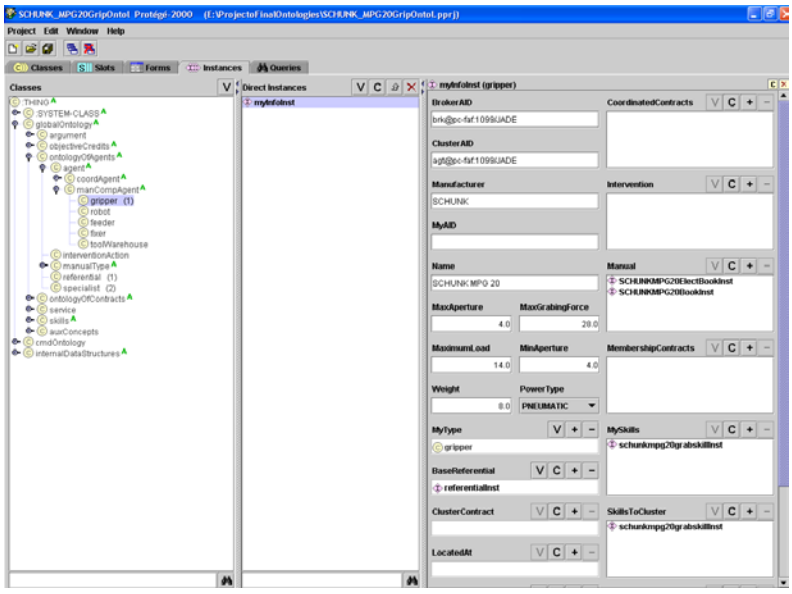

Figure 5-23 – An example of gripper ontology (SCHUNK MPG 20)

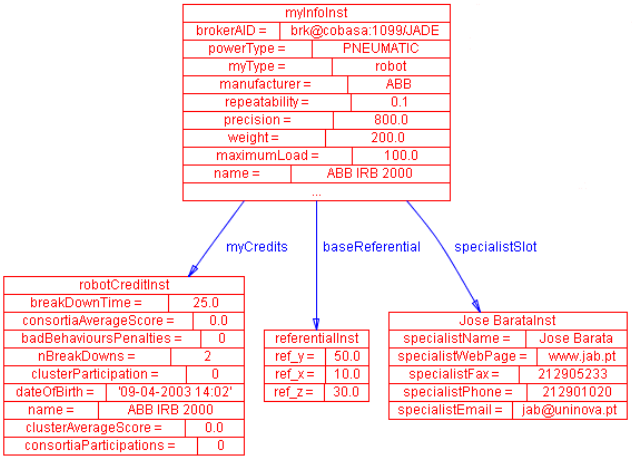

Figure 5-24 - The ABB robot agent knowledge representation (Part A)

In order to get a better visualisation of the agent's instances in the ontology the basic data structure of the ABB robot instance – *myInfoInst* is illustrated in Figure 5-24 and Figure 5-25. As it was difficult to put the whole structure together it is split into two figures.

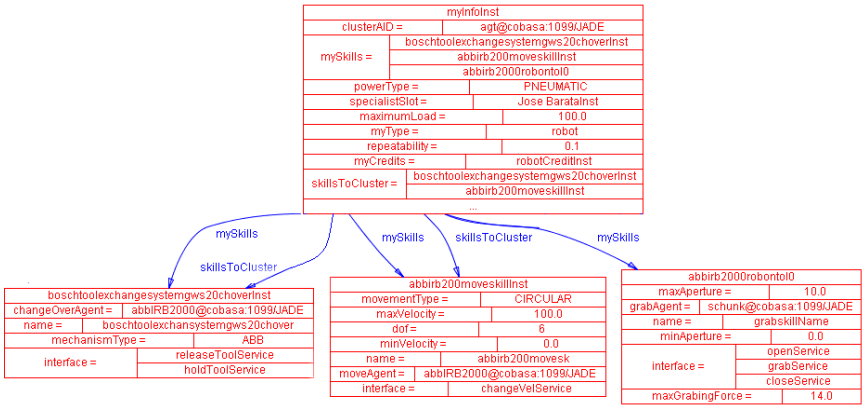

Figure 5-25 - The ABB robot agent knowledge representation (Part B)

5.3.4.3 Communication

As before, not all messages are described here. The main reason for showing only one message is to illustrate the reader what the message content looks like. This message illustrates real communication established between CoBASA agents operating in the NovaFlex cell (see section Experimental validation).

From	To	Performative	Description
GA or BA	GA	REQUEST	The request to renegotiate the membership contract

Content Message

```
((requestMembershipReneg coalitionContract (consortiaContract logistics
 (consortiaLogistics contractorAID "coord@cobasa:1099/JADE"
 contractWorkingPlace "Auto-Europa") partnersPart (SEQUENCE
 (individualConsortia indivException (individualException disByFrust
 (dischargeByFrustration score 10.0 scoreParticipation 0) disByPerf
 (dischargeByPerformance score 50.0 scoreParticipation 1) breach
 (terminateContractForBreach score 0.0 scoreParticipation 0))
 indPromises (SEQUENCE (changeover changeOverAgent
 "boschSR840@jab:1099/JADE" interface(SEQUENCE releaseToolService
 holdToolService) name "Bosch" mechanismType SCHUNK)) contractStartingDate
 "19-07-2003 17:43" indLogistics (individualLogistics timeToAnswer 12
 contracteeAID "boschSR840@jab:1099/JADE" admittedFailure
 (SEQUENCE NOT_SWITCHED_ON) numberOfTries 2) contractee
 "boschSR840@jab:1099/JADE" contractPriority 50 indMemberConsideration
 50.0)) exception (individualException disByFrust
 (dischargeByFrustration score 10.0 scoreParticipation 1) disByPerf
 (dischargeByPerformance score 50.0 scoreParticipation 1) breach
 (terminateContractForBreach score 0.0 scoreParticipation 0)) general
 (consortiaGeneral contractID "ref34" contractorName
 "coord@cobasa:1099/JADE" duration 365 purpose "This is the reneg contract
 membership" contractOntologies (SEQUENCE objectiveCredits))))))
```

Figure 5-26 – MembershipContRenegB REQUEST message

5.3.4.4 User Interface

After the creation and the registration processes in the *Cluster*, the agent user interface is the one shown in Figure 5-27.

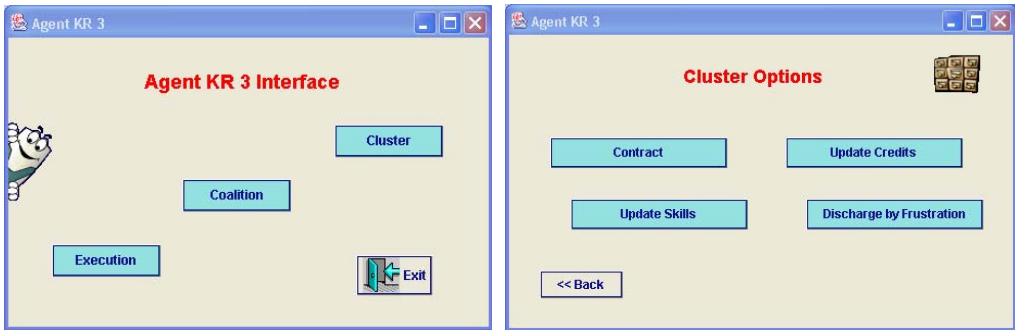

Figure 5-27 – Agent main interface and cluster options

In the *Cluster Options* window the user can find a copy of the contract established between the agent and the cluster – *Contract* button; the updating of agent credits and skills can be carried out – *Update Credits* and *Update Skills* button; and finally, discharging the cluster contract by frustration can also be done – *Discharge by Frustration* button.

In the *Coalition Options* window (Figure 5-28), the user can see which coalition contracts the agent has (MCCs). To observe the coalitions contracts of which the agent is a member the user has to click on the *Membership Contracts* button. To see the contracts where it is a coordinator the *Coordinator*

Contracts button must be clicked. Figure 5-29 illustrates the window corresponding to the membership contract.

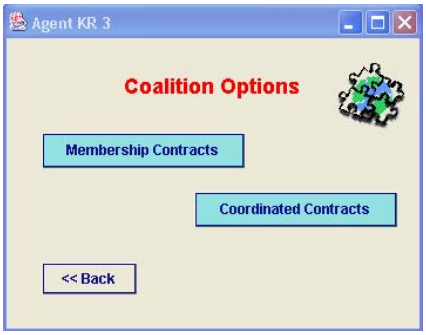

Figure 5-28 - Coalition options window

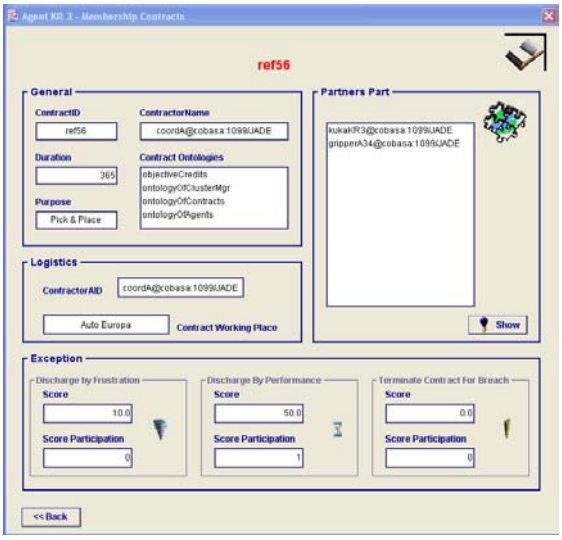

Figure 5-29 – Membership contract of the KUKA KR 3 robot window

5.3.5 Broker Agent – BA

5.3.5.1 Behaviours

All communication behaviours in the BA extend the JADE *AchiveREInitiator* since it starts all communication. A summary of the main behaviours that compose the BA are presented bellow:

- Initialisation and registration of the agent in the JADE platform.
- Behaviours for creating new coalitions.
- Behaviours for coalition alterations, which are composed of sub-behaviours to add or remove members.
- Behaviours to permit alterations in existing coalitions contracts (MCC).

5.3.5.2 Supporting Ontology

The file that supports the ontology for the broker agent (BA) is the *brokerOntology.pprj*, which inherits from the global ontology. The BA ontology has four main frames: *brokerInternalAg*, *consortia*, *clusterElements* and *consortiaChosenElems*.

The frame *clusterElements*, stores the information retrieved from the CMgA, and it is composed of the attributes *clusterAddress*, which stores the cluster address, and *existingElements*, which contains information about cluster members. Figure 5-30 illustrates an instance of the *clusterElements* with three elements of type robot.

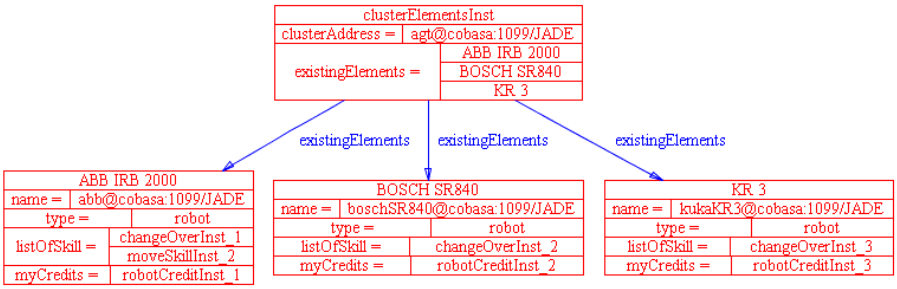

Figure 5-30 – Instances of clusterElements

The frame *consortia* contains attributes to store information about the coalitions being formed (Figure 5-31). The attribute *elements* contains the set of elements that compose the coalition, the attribute *coordinator* contains information about the coordinator, and finally the attribute *contractOfConsortia* stores the MCC contract.

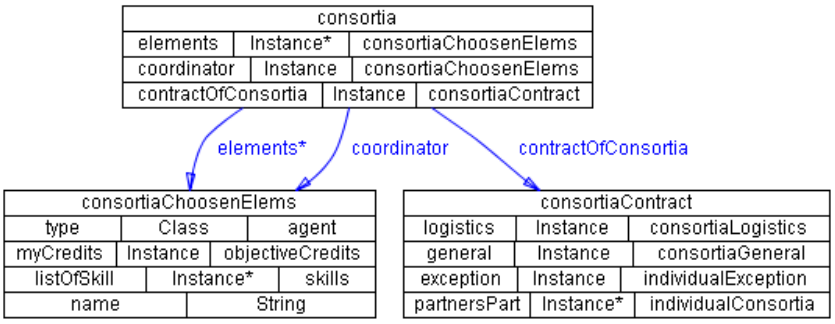

Figure 5-31 - Consortia frame and its dependencies

5.3.5.3 Communication

As before, not all messages associated with the broker agent are described here. It is only shown one message from each of the behaviours *requestMembership* and *requestCoalitionLeadersToCluster*.

From	To	Performative	Description
CMgA	BA	INFORM	The answer for the request of coalition Leaders registered in the cluster manager.
Content Message			
`((informServCluster serviceName "askForCoalitionLeaders" arg1 (argCoalitionLeaders argCoalitionLeadersAID (SEQUENCE coordA@cobasa:1099\JADE coordB@cobasa:1099\JADE coordC@cobasa:1099\JADE)))`			

Figure 5-32 – requestCoalitionLeadersToCluster INFORM message

From	To	Performative	Description
BA	GA	REQUEST	The request to negotiate the membership contract

Content Message

```
((requestMembership consortiaCont (consortiaContract partnersPart (SEQUENCE
(individualConsortia indivException (individualException disByPerf
(dischargeByPerformance scoreParticipation 1 score 50.0) breach
(terminateContractForBreach scoreParticipation 0 score 0.0) disByFrust
(dischargeByFrustration scoreParticipation 1 score 10.0)) contractPriority 20
contractStartingDate "22-07-2003 23:03" indMemberConsideration 10.0 indLogistics
(individualLogistcs timeToAnswer 60 admittedFailure (SEQUENCE
CONTROLLER_NOT_AVAILABLE) numberOfTries 2 contracteeAID
"abbIRB2000@cobasa:1099/JADE") indPromises (SEQUENCE (MoveSkill interface (SEQUENCE
) dof 6 minVelocity 20.0 name move maxVelocity 100.0 movementType (SEQUENCE PTP
LINEAR))) contractee "abbIRB2000@cobasa:1099\JADE") (individualConsortia
indivException (individualException disByPerf (dischargeByPerformance
scoreParticipation 1 score 50.0) breach (terminateContractForBreach
scoreParticipation 0 score 0.0) disByFrust (dischargeByFrustration
scoreParticipation 1 score 10.0)) contractPriority 20 contractStartingDate "22-07-
2003 23:03" indMemberConsideration 10.0 indLogistics (individualLogistcs
timeToAnswer 60 admittedFailure (SEQUENCE CONTROLLER_NOT_AVAILABLE) numberOfTries 2
contracteeAID "schunkMPG20@cobasa:1099/JADE") indPromises (SEQUENCE (GrabSkill
interface (SEQUENCE GrabService) name "schunkMPG20GrabSkill" maxGrabingForce 28.0
GrabAgent "schunkMPG20@cobasa:1099\JADE" minAperture 0.0 maxAperture 4.0 (SEQUENCE
PTP LINEAR))) contractee "schunkMPG20@cobasa:1099\JADE")) general (consortiaGeneral
contractID "ref001" duration 365 contractOntologies (SEQUENCE
ontologyOfClusterManager ontologyOfContracts ontologyOfAgents) purpose "Pick&Place"
contractorName "coordA@cobasa:1099\JADE") logistics (consortiaLogistics
contractWorkingPlace "Auto-Europa" contractorAID "coordA@cobasa:1099\JADE")
exception (individualException disByPerf (dischargeByPerformance scoreParticipation
1 score 50.0) breach (terminateContractForBreach scoreParticipation 0 score 0.0)
disByFrust (dischargeByFrustration scoreParticipation 1 score 10.0))))))
```

Figure 5-33 – requestMembership REQUEST message

5.3.5.4 User Interface

The broker agent (BA) user interface has three options: *Create Coalition*, *Change Coalition,* and *Change Contract*.

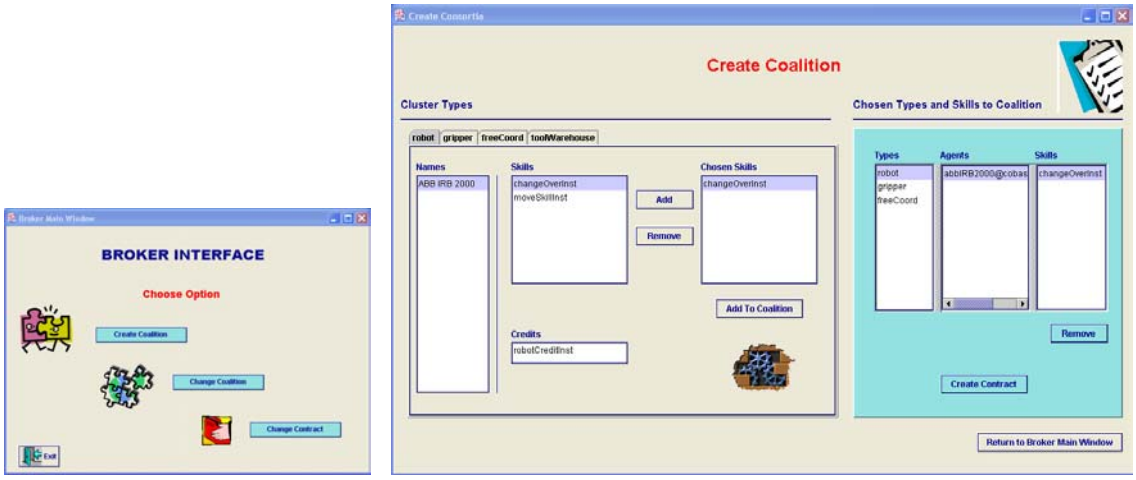

Figure 5-34 – Broker main interface and Create Coalition interface

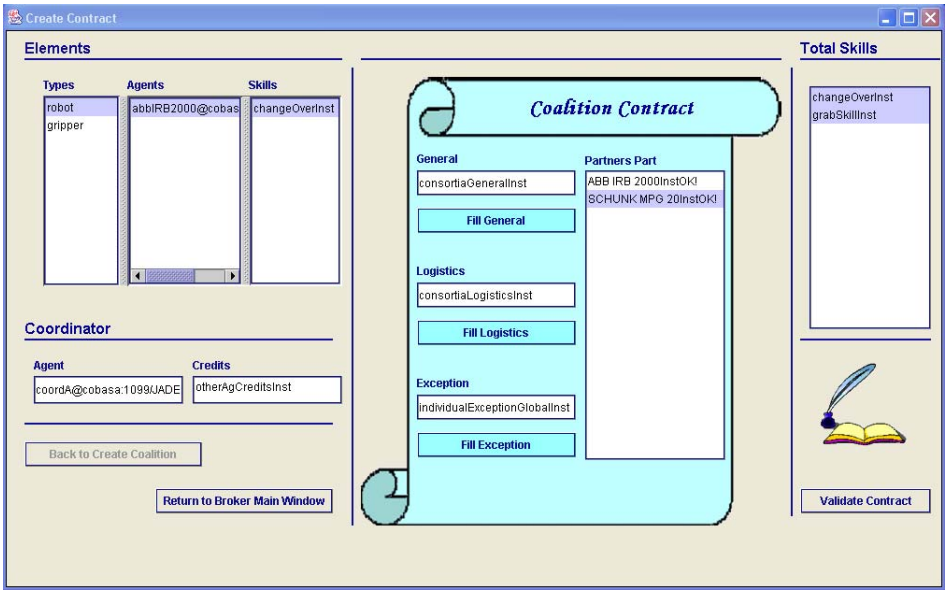

Figure 5-35 – Create Contract window

The window in the right part of Figure 5-34 is the one used for coalition creation. This window is composed of two parts: one where the information about all the registered agents in the CMgA is listed (*Cluster Types*), and the other part that is used to show the elements that have been selected by the user to participate in a coalition (*Chosen Types and skills to Coalition*). As it can be observed, in the case of Figure 5-34, the user has selected the ABB IRB 2000 robot and the SCHUNK MPG 20 gripper to be part of a coalition. Since a coalition does not exist without a coordinator, a free coordinator agent was also added to the chosen list of elements.

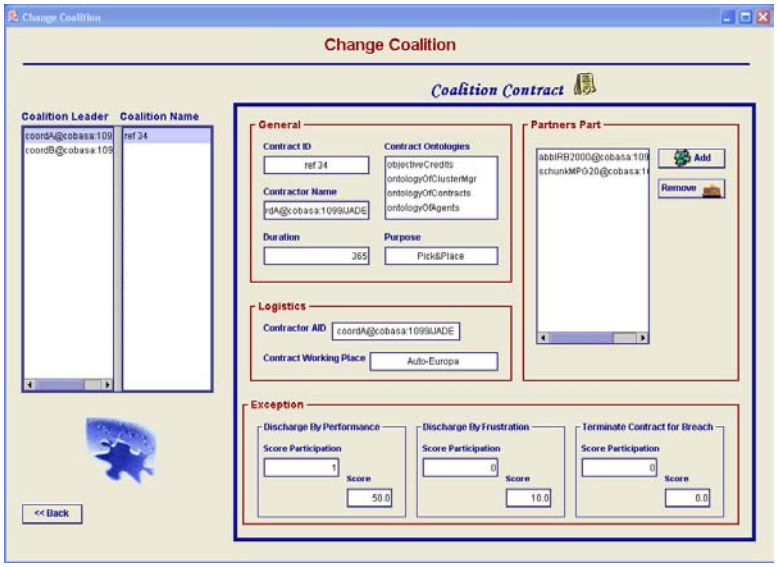

Figure 5-36 – Change coalition window

In order to create the coalition contract, the user needs to click on the *Create Contract* button and the window of Figure 5-35 opens. It is at this point that the contract is written (central part of the

window). On the left hand side, the user has the chance to review which elements he/she choose to the coalition and who the coordinator is. On the right hand side of the window, the skills brought in under the framework of this contract are listed.

The user interface depicted in Figure 5-36 is used for coalition contracts alterations. As it can be observed, all the coalition leader agents are listed in the *Coalition Leader* field. By selecting one leader the respective coalition contract appears in the right hand side of the window.

5.4 THE STEPS OF THE METHODOLOGY

This section describes the main steps required to operate CoBASA. This involves creating Manufacturing Resource Agents - MRAs that can be used as candidates in future manufacturing coalitions, and creating, changing and deleting consortia.

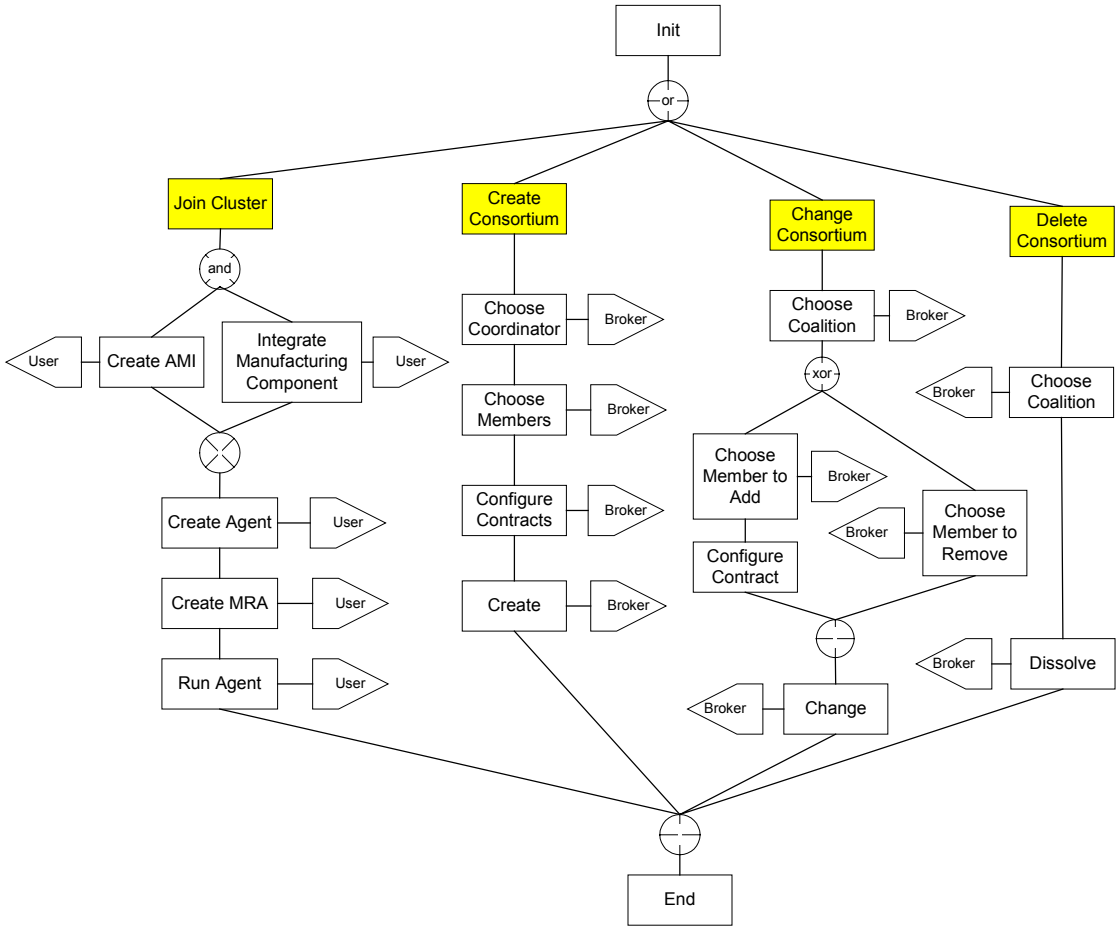

Figure 5-37 - Steps of the methodology

Figure 5-37 shows these steps by clearly indicating the most important functionalities under CoBASA. While the steps for creating, changing, and deleting consortia must be executed any time

the system is changed, the join cluster functionality needs only to be executed whenever a manufacturing component needs to be agentified. Therefore, the steps required to create a manufacturing agent are done only once in the life of a manufacturing component. Considering that in most situations the manufacturing components are reused the time spent with the agentification will be much smaller than its lifetime. This is an important fact since the agentification process is the most complex and time-consuming and, in addition, most of the used functionalities are creating, changing, and deleting consortia, which are simple and fast processes.

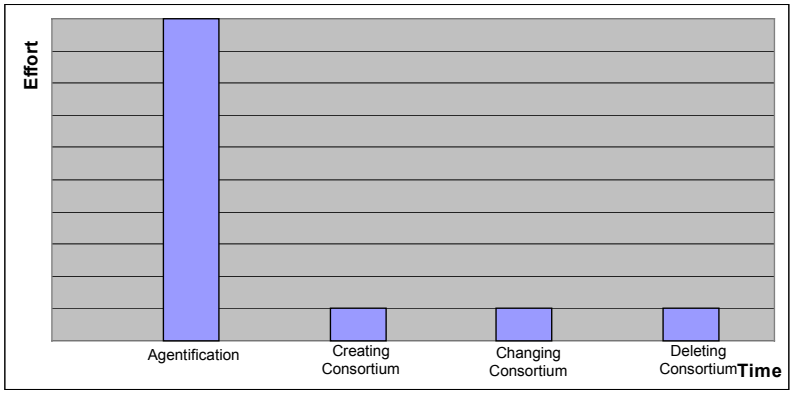

Figure 5-38 – Configuration effort variation for a manufacturing component

Figure 5-38 shows how the effort required to configure a manufacturing component evolves along its lifetime. The objective is not to indicate exact values about how much effort is required but rather to give an idea that the agentification process that is made initially, and only once, requires a bigger effort than the other phases of the lifetime. In the specific case of the manufacturing component illustrated in Figure 5-38 the manufacturing agent after being agentified, participated in a consortium, which was later on changed, and finally deleted.

5.4.1 Join the Cluster

The branch *Join Cluster* in Figure 5-37 describes the steps that must be performed by a manufacturing agent to participate in a cluster. Therefore, before joining the cluster the manufacturing equipment must be transformed into a manufacturing agent (MRA). Joining the cluster corresponds to the practical situation in which a manufacturing component is added to a given manufacturing cell. It is supposed that the broker agent and the cluster manager agent are already running. It must be remembered from chapter 4 that a MRA is more than an agentified manufacturing component and thus the following steps have been identified whenever a new manufacturing equipment needs to be transformed into a MRA:

1. ***Create the AMI***
2. ***Integrate Manufacturing Component***

3. ***Create Manufacturing Agent – Agentification of the Manufacturing Component***
4. ***Create the MRA***
5. ***Run the agent***

Steps 1 to 3 correspond to the agentification of the manufacturing component. At the end of step 4 a manufacturing resource agent exists composed of the configured generic agent plus the appropriate AMI for the manufacturing component. At the end of step 5 the agent has been registered in the cluster.

Create the AMI. In this step the user configures the generic AMI according to the functionalities and requirements of the manufacturing component to which the AMI is going to be connected. It must be recalled that the AMI establishes the link between the generic agent (GA) and the manufacturing component. This agent is specific to the type of the manufacturing component to which it is connected in terms of the functionalities it offers.

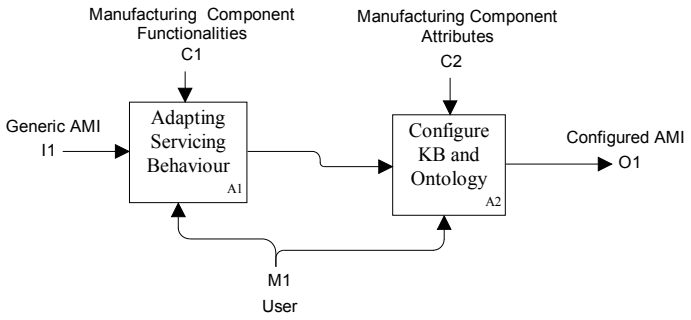

Figure 5-39 - Activities when configuring the AMI

The generic AMI includes roughly the behaviours required for user interface, and the generic behaviour to attend requests from the GA to which it will be connected. Configuring it corresponds to implement the specific functionalities of the agent to which it will be connected. This is represented in Figure 5-39 by the activity *Adapting Servicing Behaviour*. Furthermore, it is necessary to configure the individual KB and ontology of the agent to the individual attributes of the manufacturing component, which is represented by the activity *Configure KB and Ontology* in Figure 5-39.

Integrate Manufacturing Component. In this phase the functionalities of the manufacturing component controller are modelled using computer based abstraction mechanisms such as Remote Procedure Calls (RPCs) or distributed objects. When using RPCs, the functionalities of the controller are represented by the methods included in the RPC that models the controller. In the case of modelling the manufacturing controllers using objects or distributed objects, the attributes of the object model the static characteristics of the controller being modelled while its methods model the controller's functionalities. This modelling is fundamental since many of today's manufacturing controllers do not provide an abstraction at this level and, hence, to be controlled, they need special

commands usually sent by a RS232 protocol. This is the process of connecting the physical controller to the agent. This could be an easy task if every physical component was controlled directly by its own agent. However, outdated legacy controllers with close architectures control most of the existing physical components.

Independently of how the commands are sent to the specific manufacturing controller the important thing to keep in mind is that it is necessary to adapt the computational abstraction of the manufacturing component to an abstraction level that can be used by agents. This is so because, in the end, the commands that the physical manufacturing controllers receive are originated in the agents that compose CoBASA (MRA agents).

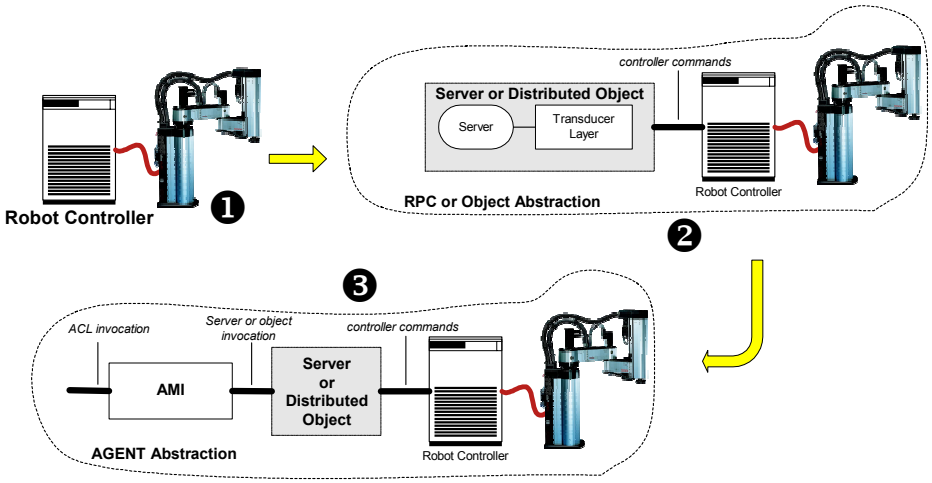

Figure 5-40 - Transformation of a manufacturing component into an agent

The problem in hands in this phase is nothing other than the integration problem already referred to in chapter 3. The objective of this phase is thus to build a server, using RPCs or a distributed object, using CORBA or DCOM, whose methods can be invoked from any computer system connected to a network. This approach decouples the physical manufacturing controller from the agent environment, which facilitates the integration. To integrate these legacy components in the agents' framework it is thus necessary to develop a software wrapper to hide the details of each component (Barata, 1995; Barata & Camarinha-Matos, 1995a, 1995b; Camarinha-Matos, Barata, & Flores, 1997; Camarinha-Matos, Seabra Lopes, & Barata, 1996). The wrapper acts as an abstract machine to the agent supplying primitives that represent the functionality of the physical component and its local controller. The steps of this phase are indicated in Figure 5-40 with the numbers ❶ and ❷. At the end of this phase the manufacturing component can be commanded through the activation of a RPC or the invocation of one of the methods of the object that mimics it.

Agentification of the Manufacturing Component. The physical manufacturing component is effectively transformed into a manufacturing agent at the end of this phase, hence, the name agentification. This is represented in the number ❸ of Figure 5-40.

Steps 1 to 3 occur only when the physical manufacturing component is first integrated in the community of agents. All the other situations happen after the component has been already agentified, like for instance changes on the consortium.

The activity that must be done in this phase is connecting the AMI, which has been already configured in step 1, to the object or server that represents the physical manufacturing controller. The AMI accesses the wrapper services using a local software interface (proxy), where all services implemented by the wrapper/legacy controller are defined. Figure 5-42 illustrates how the AMI connects to the wrapper by showing some details of it that were not presented in Figure 5-40.

The generic AMI agent is a simple agent with a simple behaviour to accept requests from other agents. When a REQUEST is received, the AMI calls the wrapper to execute the requested service and when the command is executed, it sends back a DONE message to the enquirer agent. Each AMI implements the services supported by the physical component by configuring what the services of the proxy to which it is connected are and the name/address of the component. Because issuing Agent Communication Language REQUEST commands to the AMI does the activation of the manufacturing component, it can be stated, then, that this software layer provides agent abstraction (Figure 5-40, number ❸).

Create the MRA. At the end of this phase a Manufacturing Resource Agent (MRA) is created. MRAs are composed of a Generic Agent connected to the agentified manufacturing component produced in the last phase. In the CoBASA framework only manufacturing components represented in this way (MRAs) are able to join the cluster and, hence, participate in coalitions.

In Figure 5-41 the activities that must be executed to create a MRA are shown. In the activity *Initialise GA* the user initialises the Generic Agent, which, in this case, is used to supply negotiation skills to the MRA agent. It is then necessary to configure the initialisation and the ontology file to be used by the agent, as it was indicated when Protégé-2000 was described. At the end of the activity, the GA is an initialised instance of a generic agent.

The connection to the AMI is guaranteed by creating a consortium contract between the generic agent and the AMI, establishing in this way a consortium. The member promise part (AMI) of the contract contains the services supplied by the AMI. This is supported by the activity *Create Contract*, which is done by the user using the Protégé 2000 environment to create the contract. The information existing in the *configured AMI*, which was configured before, constrains the output of this activity because the behaviour of the MRA agent depends on the skills brought in by the AMI to whom it is connected. The contract being created is attached to the variable *coordinated contracts* that contains all the contracts that this GA is coordinating. In this situation, the GA only coordinates, in fact, the AMI to which it is connected.

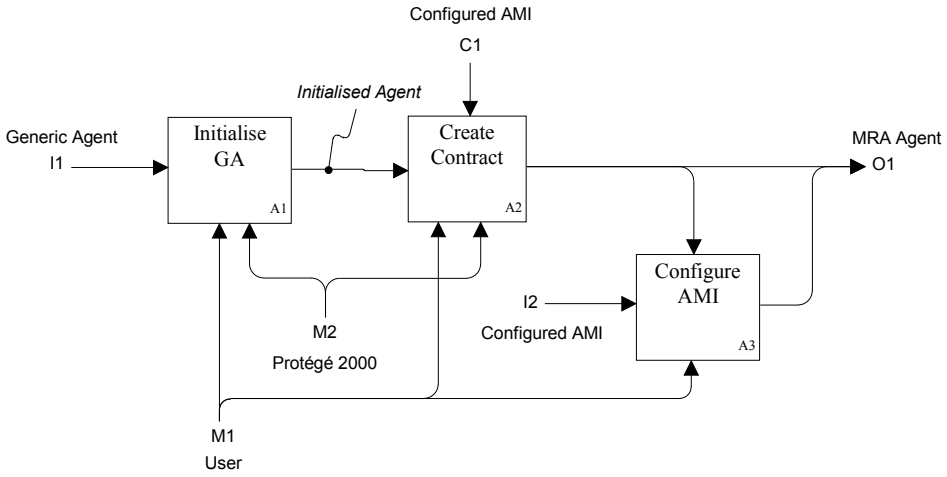

Figure 5-41 - Activities in the Create MRA phase

At the end of the *Create Contract*, activity the MRA agent is complete and composed of the generic agent GA attached through *coordinated contracts* to its AMI. However, the AMI is not yet configured in the sense that it should only accept requests from the GA that is attached to it. This is done in the activity *Configure AMI* in which the AMI must also be configured to include the name of the generic agent to which it is connected. This guarantees that an AMI refuses any requests from unknown agents.

Figure 5-42 shows the various entities that compose the MRA. It must be remembered again that a MRA needs only be created when the manufacturing component is used for the first time. Future uses of the manufacturing component through its representative (the MRA), in coalitions and possible different clusters, do not involve any changes to it.

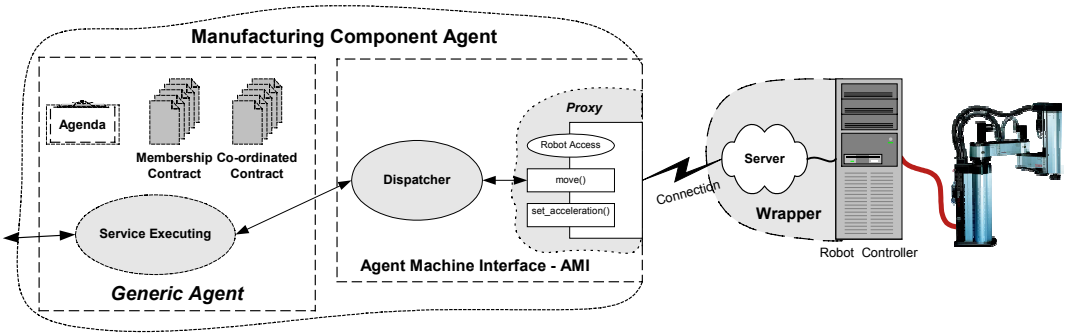

Figure 5-42 - The Manufacturing Resource Agent – MRA

Run the agent. This step is just to guarantee that the agent joins the cluster.

5.4.2 Create a Consortium

In Figure 5-37 this phase is represented under the label *Create Consortium*. It involves the following steps:

1. Choose coordinator
2. Choose members
3. Configure the Multilateral Consortium Contract (MCC) that governs the consortium
4. Create the Consortium.

At the end of this phase a manufacturing coalition is created that is able to perform some manufacturing operations, which are achieved through the composition of skills that each manufacturing component (MRA) brought in to the coalition.

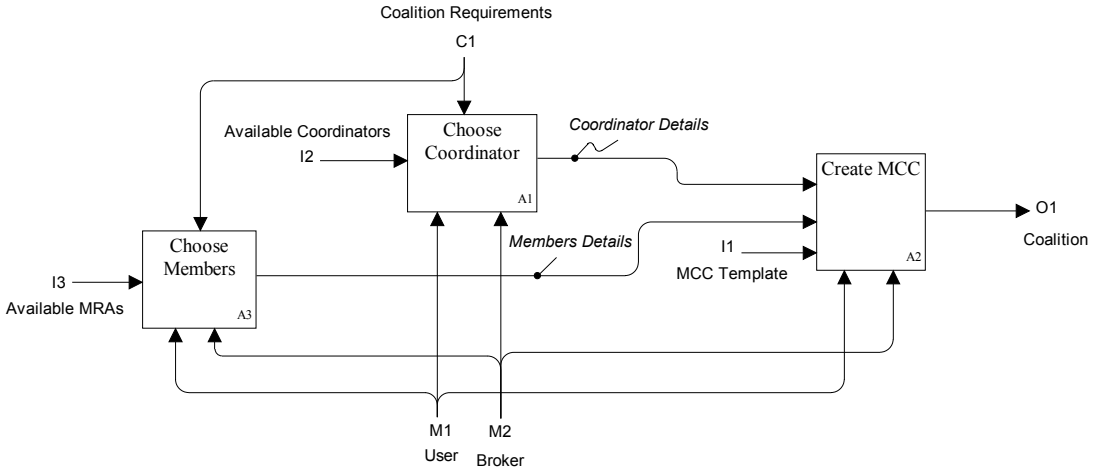

Figure 5-43 - Activities involved in Creating a Coalition

Using an IDEF0 diagram Figure 5-43 shows the activities involved in this phase. A coalition is effective after the MCC contract has been established. It is assumed that only MRAs belonging to the cluster can be chosen to be candidates for participation in coalitions. It is also assumed that some coordinator agents exist in the cluster. Although a human operator (user) is also involved in this phase, he/she creates a coalition using the broker agent.

Choose coordinator. In the first step the user chooses which coordinator will be used from the ones available in the cluster. It must be kept in mind that the broker requests the Cluster Manager Agent (CMgA) for information about which MRAs and coordinators it has registered. The details of the coordinator will be used to create the MCC contract.

Choose members. The user accesses the entire candidate MRAs and then, by analysing their functional characteristics and skills, chooses the ones that can fit the requirements needed by the

coalition being composed. The skills that each assigned agent will bring to the consortium are also chosen during this step.

Although in the current state of CoBASA this is a manual task, in the future this can be changed to a semi automatic task, in which an advisor tool exists to help the user in building the best coalition from the available candidates. As for the case of the coordinator, the details of each member (mainly the details of their skills) will be used to build the individual part of the MCC contract that governs the coalition.

Configure the MCC. In this phase the MCC contract is created, starting from the template contract that is available for coalitions. The broker semi-automatises this process by guiding the user in the fulfilment of the contract.

Create the consortium. This corresponds to an action of the user informing the broker that he/she accepts the coalition and that the broker can now negotiate the acceptance of the contract. The broker, then, automatically negotiates with the coalition members and gets the contract approved. In this specific case, the behaviour *MbshpContractMgmt* of each candidate MRA is activated and an interaction starts between it and the broker, ending with the subscription of the consortium contract. At the end of this negotiation, the MRA has two contracts: a co-ordination contract, the one established at the *Join Cluster* phase between the agent and the AMI, and the membership contract corresponding to the coalition that has just been formed.

5.4.3 Change a consortium

In Figure 5-37 this phase is represented under the label *Change Consortium*. The goal here is to change a coalition either by adding or removing elements. In both situations the user must be sure that the coalition, after being changed, still corresponds to its objectives. The activities that must be performed are:

1. Choose the coalition
2. Choose member to add or remove
3. Change the coalition MCC contract.

The coalition is chosen in the same way MRAs or coordinators are chosen using the broker. After the coalition is chosen, its members, the MCC contract that governs that coalition, and the individual characteristics of the members are displayed in the broker. If it is the case of a removal, the user just needs to select the member to be removed. If after being removed the set of skills is approved by the user, he/she can be committed to the changes by informing the broker that it can renegotiate the changes that were made with the coalition coordinator. This renegotiation process is done automatically between the broker and the changed coalition coordinator.

If, on the other hand, it is the case of an addition, the user chooses the new member(s) using the same process for the creation of a new coalition. When the user is satisfied with the team he/she needs to work, with the help of the broker, on the MCC individual parts related to the new members. After that the broker is informed to start the renegotiation process with the coordinator agent about the changes that have been made.

5.4.4 Delete a consortium

In Figure 5-37 this phase is represented under the label *Dissolve Consortium*. The goal here is to delete a consortium, which just corresponds to eliminating from each member of the coalition being dissolved the MCC contract that is stored in its variable *membership contract*. To achieve this situation the user must, using the broker again, choose the coalition and then inform the broker that the coalition is to be dissolved. The broker then informs autonomously the coordinator and the members of that coalition. The coordinator then deletes the MCC contract from its variable *coordinated contract*, while the members delete it from the *membership contracts* variable.

5.5 EXPERIMENTAL VALIDATION

The experimental validation was performed in order to prove the adequateness of CoBASA in a real industrial case. The idea was to test the prototype against a scenario that could represent a real manufacturing environment, which is the case with NovaFlex.

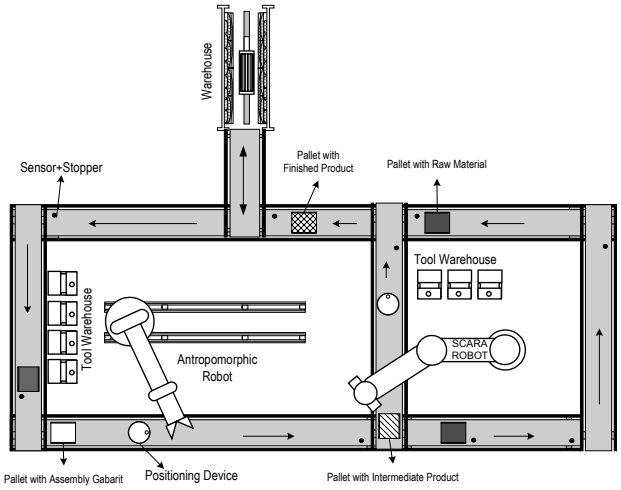

Figure 5-44 - The global validation scenario

The flexible manufacturing system (Novaflex), established at the UNINOVA, was a natural candidate because it resembles a real manufacturing system composed of industrial components. A

simplified diagram of part of the Novaflex is illustrated in Figure 5-44, while Figure 5-45 shows a partial picture.

Figure 5-45 – Partial view of Novaflex

Considering now the cluster concept introduced in the last chapter it became obvious that NovaFlex is itself the cluster. The cluster is then the abstraction of that entity (the physical NovaFlex) that aggregates all the manufacturing equipment. Different coalitions can be created out of this equipment. These coalitions represent no more than different ways of exploring the cell. Therefore, there are two ways of regarding the cell: the physical and the abstract.

In the physical way the NovaFlex is a cell composed of several manufacturing equipments that are related by physical relations. It is possible to imagine that parts of the entire cell can be operated independently as well as that equipment can be added or removed.

In the abstracted way the NovaFlex is a cluster composed of several manufacturing agents (agentified manufacturing equipment) whose entire set of skills represents the potential of this cluster to solve problems. Whenever a problem requiring a specific set of skills available in the cluster is needed a coalition can be created fitted with that specific set of skills. Furthermore, several problems can be answered simultaneously as long as the cluster has members able to answer the various problem requirements. The interesting point about this vision is the dynamics of the coalitions.

It is important to remember that any manufacturing equipment that might need to be added to the NovaFlex physical infrastructure (physical view) must join the cluster NovaFlex (abstract view). Hence, adding equipment corresponds to joining the cluster while removing equipment corresponds to leaving the cluster.

The scenario used to test the prototype was not the entire Novaflex but rather parts of it. This strategy was chosen to keep the complexity of the implementation to a level that could be managed during the period of the thesis. On the other hand, it is not necessary to include a large number of manufacturing components to prove the validity of the concepts behind CoBASA. To achieve this

objective two different areas of Novaflex were considered: the SCARA assembly system, and the ABB assembly system. The conveyors were not considered.

Figure 5-46 - The SCARA assembly system

The SCARA assembly system (Figure 5-46) is composed of one robot BOSCH SCARA SR-80 equipped with a tool exchange mechanism, a tool warehouse composed of 6 individual slots, 4 different grippers, a feeder, and a fixture.

Figure 5-47 - The ABB assembly system

Figure 5-48 - The product

The ABB assembly system (Figure 5-47) is composed of one robot ABB IRB 2000 equipped with a tool exchange mechanism, a tool warehouse composed of 4 individual slots, 4 different grippers, a feeder, and a fixture.

Each assembly system does assembly operations required for the assembly of a toy watch, which is the product being assembled at NovaFlex (Figure 5-48).

Equipment Type	Description	Agent Id
Robot	ABB IRB 2000	abbIRB2000@pc-faf:1099/JADE
	BOSCH SR 80	boschSR80@pc-faf:1099/JADE
Feeder	Vibratory Feeder	vibBowlFeeder1@pc-faf:1099/JADE
		vibBowlFeeder2@pc-faf:1099/JADE
Fixer	BOSCH Fixer	boschFixer1@pc-faf:1099/JADE
		boschFixer2@pc-faf:1099/JADE
Tool	SCHUNK PGN50	schunkPGN50@pc-faf:1099/JADE
	SCHUNK MPZ30	schunkMPZ30@pc-faf:1099/JADE
	SCHUNK MPG20	schunkMPG201@pc-faf:1099/JADE
		schunkMPG202@pc-faf:1099/JADE
Toolwarehouse	BOSCH Tool Warehouse	boschToolWarehouse1@pc-faf:1099/JADE
		boschToolWarehouse2@pc-faf:1099/JADE
	SCHUNK Tool Warehouse	schunkToolWarehouse1@pc-faf:1099/JADE
		schunkToolWarehouse2@pc-faf:1099/JADE
FreeCoord	Coordinator Agents	coordA@pc-faf:1099/JADE
		coordB@pc-faf:1099/JADE

Table 5-7 – Members of the NovaFlex cluster

Equipment Type	Description	Agent Id
Robot	ABB IRB 2000	abbIRB2000@pc-faf:1099/JADE
Feeder	Vibratory Feeder	vibBowlFeeder1@pc-faf:1099/JADE
Fixer	BOSCH Fixer	boschFixer1@pc-faf:1099/JADE
Tool	SCHUNK MPZ30	schunkMPZ30@pc-faf:1099/JADE
	SCHUNK MPG20	schunkMPG201@pc-faf:1099/JADE
Toolwarehouse	SCHUNK Tool Warehouse	schunkToolWarehouse1@pc-faf:1099/JADE
		schunkToolWarehouse2@pc-faf:1099/JADE
FreeCoord	Coordinator Agents	coordA@pc-faf:1099/JADE

Table 5-8 – Members of the coalition 1

Table 5-7 indicates all the manufacturing agents that belong to the NovaFlex cluster that are used in the experimental validation. Each agent identifier corresponds to a physical manufacturing component belonging to the NovaFlex cell. It must be recalled that the validation is done only with a subset of all the equipment that compose the NovaFlex. From this cluster it is possible to create, for instance, two different manufacturing coalitions.

Equipment Type	Description	Agent Id
Robot	BOSCH SR 80	boschSR80@pc-faf:1099/JADE
Feeder	Vibratory Feeder	vibBowlFeeder2@pc-faf:1099/JADE
Fixer	BOSCH Fixer	boschFixer2@pc-faf:1099/JADE
Tool	SCHUNK PGN50	schunkPGN50@pc-faf:1099/JADE
	SCHUNK MPG20	schunkMPG202@pc-faf:1099/JADE
Toolwarehouse	BOSCH Tool Warehouse	boschToolWarehouse1@pc-faf:1099/JADE
		boschToolWarehouse2@pc-faf:1099/JADE
FreeCoord	Coordinator Agents	coordB@pc-faf:1099/JADE

Table 5-9 – Members of the coalition 2

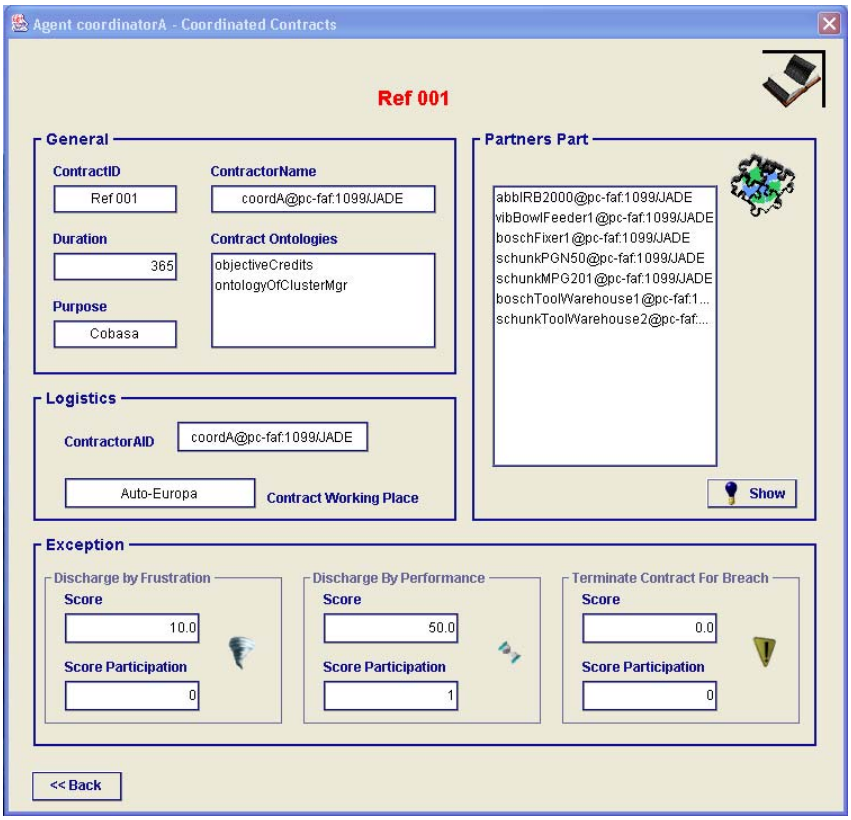

Figure 5-49 – The contract that regulates coalition 1

An example of the contract that regulates coalition 1 is illustrated in Figure 5-49. The identifiers for the coordinator as well as the members that compose the coalition can be found there. This contract is the basic reconfiguration mechanism because the contract is the entity that regulates the behaviour of the coalition members and, therefore, any changes to the coalition imply alteration in the contract. Please note that creation and changes are made in the broker agent.

Using the broker agent it is then possible to create coalition 2. The contract that regulates it is illustrated in Figure 5-50. Please note that the contract is shown using a different user-interface. In this case the contract is shown when the option to change a coalition is chosen.

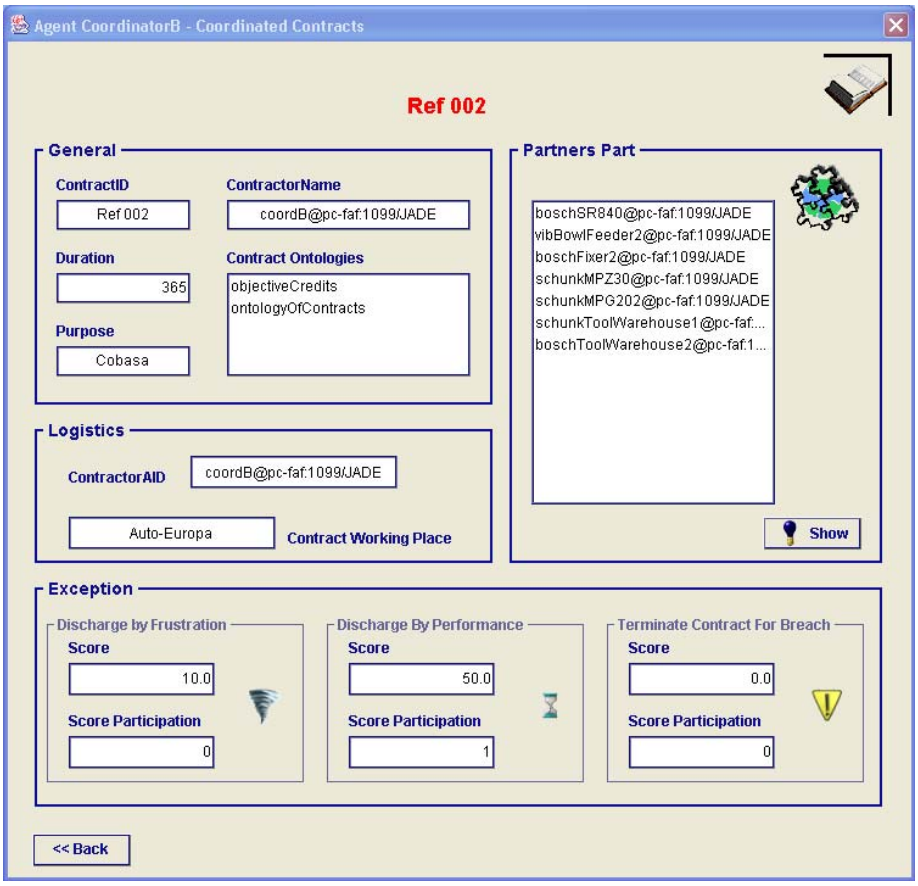

Figure 5-50 - The contract that regulates coalition 2

If, for instance, it was necessary to add a new gripper to one of the coalitions, the only thing that would be required was altering the contract by including the new gripper agent in the contract partners part and choosing the skills to be brought to the coalition.

The creation, changing, and deleting of coalitions is thus possible using real manufacturing components.

The experiences that were made under the described scenario proved that CoBASA is adequate to support shop floor reengineering because it was possible to reconfigure the NovaFlex using only contract reconfiguration.

5.6 EXTERNAL VALIDATION

Two independent external specialists, Eng. Luís Flores, and. Eng. Alberto Gavinhos, were asked to test and evaluate the CoBASA prototype under the scenario introduced at the beginning of this chapter. They were also asked to investigate the potential use of CoBASA in industrial shop floors based on their experience. Due to the wide experience of these two specialists in the shop floor, and especially in shop floor reengineering projects their opinion was considered a good source of validation of the work developed in this thesis.

Both specialists reported that they were positively impressed with the CoBASA architecture and the prototype they tested. They agreed and reported that CoBASA is a step further in the direction of better methods and tools to facilitate shop floor reengineering.

A brief description of the curriculum of each specialist as well as the report with their comments can be found in appendix A.

5.7 THEORETICAL VALIDATION

The validation method used here was defined by Olesen (Olesen, 1992) and it was chosen because it is well suited for applied research.

The main objective of this thesis has been to prove that:

> *Coalitions/consortia of manufacturing agents, which represent manufacturing components, and whose behaviour is governed by contracts is a suitable approach to deal with the shop floor reengineering problem.*

Internal logic. The internal logic, as defined in (Olesen, 1992), means *"that the result is based on known and accepted theories, and that there is a connection between the starting point, hypothesis, and the result."*

The starting point has been to try to create an adequate approach, or even an architecture, for solving the reengineering problem afflicting modern manufacturing systems. In order to achieve this goal a survey of the current control and planning systems and architectures, supportive technologies, and analogous applications such as contract law and virtual organisations, has been detailed. Therefore, the hypothesis presented in this thesis is based on a broad analysis of known, accepted and well-published theories, all of which have bearing on the details of the hypothesis. In as much as the hypothesis actually presents a new approach, there are no violations of known theories. The thesis then proceeds from the hypothesis to an implementation (validation) by directly applying to the proposed CoBASA architecture selected and established approaches, and technologies, including multiagent systems, negotiation, ontologies, etc. Hence, the link between the hypothesis and validation is based on known and accepted approaches and theories.

Truth. Truth, as defined in (Olesen, 1992), means " *that the theoretical and practical results can be used to explain "real" phenomena.*"

The implementation of CoBASA has been applied to a real manufacturing environment: the manufacturing cell Novaflex, at the Uninova Institute. A cluster representing the Novaflex has been implemented with agentified real manufacturing components (robots, grippers, tool warehouse, etc), and coalitions of real manufacturing components have been created. In addition the integration of legacy equipment was proved when it was connected to the agent world. There is therefore evidence to support the claim that the results of this thesis are valid since they do explain "real" events.

Acceptance. Acceptance, as defined in (Olesen, 1992), means "*that other researchers accept the theories used in the project, and that professionals use tools based on the theory*."

It is, at this stage, difficult to quantify in detail the acceptance of CoBASA in terms of the criterion related to tools. The results of this thesis have been demonstrated and discussed with a large number of field experts from academia and industry. The participation in several recognised international conferences, the participation in several European projects, and the long collaboration between the research group to which the author belongs and industry, have provided the necessary framework for the dissemination and discussion of this thesis' concepts and ideas, at theoretical level.

The acceptance of the work may be quantified by the following points:

1. Invitations to participate in discussions about possible use and further development of some of the ideas developed in the framework of this thesis in European research project proposals: EUPASS (Evolvable Ultra Precision Assembly Systems), and EMAT (Evolvable Micro Assembly Technology).
2. The author has participated in other European Projects (ESCN, Assembly-Net, ProSME, ThinkCreative, VO-MAP, etc), in which it was possible to discuss with different researchers the work being done in this thesis. The discussions with these highly skilled professionals and researchers have given rise to no major objections to the ideas given here.
3. In addition to the several papers published in books as well as in refereed conference proceedings, four publications have been submitted and accepted for publication in the following established journals:
    a. The International Journal of Computer Integrated and Manufacturing
    b. The International Journal of Network and Virtual Organisations
    c. The Assembly Automation Journal
    d. IEEE Transactions on Robotics and Automation.
4. Several papers were presented in various research conferences with referees. The discussions and the number of questions raised at the end of the sessions indicated that the work was in the right direction, and no major objections were raised.

5. Exploratory contacts with the group that is intending to create a working group under the FIPA organisation to develop FIPA-compliant standards for the next generation global enterprises that encompass generic technologies supporting collaborative work in a dynamic distributed environment, covering both the higher level and lower levels of manufacturing companies. One of the main foreseen activities for this working group is tackling agility by the agentification of manufacturing components.
6. CoBASA has been discussed with some specialists from industry, in particular the automotive industry, and questions regarding the feasibility of industrial implementation have been thoroughly analysed. Although far from being an industrial product, the concluding opinion of these specialists was that the ideas presented in this thesis represent a sound and realistic foundation for further investigation and development.

Applicability. Applicability, as defined in (Olesen, 1992), means "*that the use of the tools allows the probability for success to increase with repeated use. It does not necessarily lead to success every time, but over a period of time will give better results than if not used.*"

In terms of applicability of the results given in the thesis, no major problems were identified. In fact, better implementations can be achieved as better technological solutions will emerge; the issue is not the theoretical base, but the technological support which may enable its full implementation in terms of industrial applicability. The work being done and the standardisation effort that is being pursued at the FIPA level can only increase the applicability potential. The thesis implemented a working prototype and its results are verifiable.

Novelty value. Novelty value, as defined in (Olesen, 1992), means "*that new solutions are presented, or that new ways of looking at a particular problem are introduced*."

From the discussions carried out at the different forums, an impression emerged that the work being presented is novel and that a new way of marrying concepts from different areas, in an innovative approach, is being offered. To the author's knowledge there are no similar solutions despite the existence of other multiagent approaches dealing with the shop floor. Novelty arises out of addressing shop floor as a living entity with a dynamic life cycle (reengineering/continuous changes), in which coalitions of manufacturing components are dynamically created favouring configuration rather than programming by the use of contracts to regulate the behaviour of the coalitions.

In spite of being recognised that further work is needed for industrial use, and that further research development still might be needed to be carried out, it is believed that the hypothesis and the results have been proved. The open research points that need to be carried out will be addressed in the last chapter.

6 Conclusions and Future Work

This chapter summarises the results achieved in this thesis and proposes future work. The results achieved are divided into the following categories: scientific and technical contributions, prototypes, publications, and new project proposals. The barriers that can hinder industrial dissemination are briefly discussed, and finally future research work that may be proposed as a logical consequence of this thesis is listed before a concluding remark is stated.

6.1 CONCLUSIONS

The subject of this thesis is shop floor reengineering to which a multiagent based architecture featuring coalitions of agentified manufacturing components (CoBASA) has been designed and developed. A prototype applied to a real manufacturing cell (the NovaFlex at Uninova) was implemented to prove that CoBASA was feasible. Complementarily, two external specialists were asked to test the prototype and assess its validity and usability.

The research question driving this thesis is about which methods and tools should be developed to make current manufacturing control/supervision architectures reusable and quickly modifiable in order to simplify the shop floor reengineering process.

To show that this is a valid and useful research question the current challenges faced by manufacturing companies were highlighted as well as that shop floor reengineering activities are becoming more and more important to the point that the competitiveness of manufacturing companies can only be ensured if appropriate agile shop floor control/supervision architectures are developed to

tackle this problem. An historical overview of the main manufacturing business paradigms and external factors that have constrained their evolution have been also described to accentuate the link between them and the shop floor control/supervision architecture requirements. For instance, the need for shop floor reengineering support, at the time of the mass production paradigm, was low because changes ofthe shop floor were almost inexistent. At that time, an architecture such as CoBASA would not have been necessary. As agility is becoming an ever more pressing requirement, the need for agile control and supervision architectures grows.

The departing research question was answered by proposing a shop floor reengineering architecture and methodology that supports shop floor life cycle evolution by simplifying any changes and modifications required to be made to the shop floor control/supervision architecture. This architecture, which is called CoBASA as an acronym for Coalition Based Approach for Shop floor Agility, uses a multiagent approach, and is inspired on the concepts of collaborative networks.

The design and development of such a shop floor reengineering architecture, targeting real shop floor applications, required the study and analysis of a wide range of concepts and supporting technologies, which were detailed to enable the reader to better understand the concepts and technologies behind CoBASA.

The validity of the methodology and architecture was proved following three different dimensions. The first dimension addresses the feasibility of the proposed solution, the second verifies if the system is adequate or serves its purpose (adequateness), and the third validates theoretically the proposed solution. The feasibility was proved by showing that a prototype system based on the CoBASA architecture was implemented. In addition, the selection of the agent development environment, and the supporting tool for knowledge models (or ontology development environment) was discussed, and the selected tools described. Also related to the feasibility dimension, the steps required to operate CoBASA were also discussed, namely creating Manufacturing Resource Agents (MRAs) that can be used as candidates in future manufacturing coalitions, and creating, changing, and deleting coalitions. The adequateness of the architecture was proved by testing it against a complex scenario composed of real manufacturing components (the NovaFlex pilot manufacturing system), and asking external experts to assess the prototype. The theoretical validation consisted in answering several questions that seems adequate for applied research.

6.1.1 Results

The work done led to the achievement of several results and effects that will be summarised as:

- Scientific and technical contributions
- Prototypes
- Publications
- New projects proposals.

Scientific and Technical Contributions

The relevant scientific and technical contributions can be divided into three groups: conceptual contributions, which include the most significant and original ones, technological contributions, which include those that although relevant are more related to technology, and other contributions, which include those that although important have minor scientific relevance.

- **Conceptual Contributions**

1. **Solutions for shop floor agility should consider a new phase in the manufacturing equipment life cycle – the reengineering phase**. This is an important contribution because it has resulted in a new vision on the way shop floor agility should be tackled. Traditional approaches to shop floor agility consider an almost static life cycle. This type of systems is essentially designed to support the operative phase, and, therefore, changing or modifying them is difficult since their functionalities were designed to optimise their operation rather than their modifiability. This thesis took a different approach by recognising that it is impossible to design any production system in which all the present and future requirements might be foreseen. Therefore, the thesis work sustains the principle that the production system should be designed with attributes (modularity, compositions of manufacturing components easily created and modifiable, favouring configuration rather than programming, …) to facilitate its evolution along its life cycle phases. Therefore, since real manufacturing systems are not completely flexible and agile and will at some stage require changes, the reengineering phase is introduced to represent the phase during which the system is being changed.

2. **Partial support for the shop floor reengineering methodology**. This methodology collects, organises and structures the different methods, tools and activities that are needed to support the production system life cycle. CoBASA, for instance, covers the operative and reengineering phases of the life cycle in what is related to the control/supervision architecture. This thesis did not cover all phases and activity areas that are required for full life cycle support. However, it might be important to add that the work developed under this thesis has been discussed with other European researchers in the framework of some European projects, which eventually recognised the relevance and importance of having a reengineering methodology. The recognition that a reengineering methodology is required is expressed by the fact that three of the projects that were submitted to the 6th framework program of the EU (EUPASS, EMAT, and RAMP-UP) have tasks assigned to the development of a reengineering methodology.

3. **The CoBASA architecture**. The proposed architecture, featuring coalitions/consortia of agentified manufacturing components, whose members are chosen from an entity (cluster) that contains all the manufacturing components of a given cell, using a broker agent as an

intermediary, proved to be an adequate solution for the agile shop floor problem (see validation in chapter 5).

4. **The use of contracts to govern the interaction between autonomous agents**. From the author's knowledge, this was the first time that contracts inspired on legal principles have been used to govern the relationships between autonomous agents. By using this approach, the interactions are constrained by what the contract says and not by the individual program of each interacting agent. As long as the agents are able to read and understand the role of contracts they can be generically involved in any relationship constrained by contracts.

5. **Shop floor agility can benefit from concepts inspired in the VO/VE area**. Before the start of this thesis, VO/VE concepts had already been considered a useful and adequate measure to improve the ability of enterprises to cope with turbulent environments – improves their agility. However, these concepts had not ever been applied to improve shop floor agility (the enterprise lower level). Therefore, this thesis is original in the way it has proved that concepts already proved in the higher levels might be successfully applied to the lower ones. This helps reducing the gap between the higher and lower levels in manufacturing companies.

- **Technological Contributions**

1. **Application of multiagent technology to real shop floor manufacturing equipment**. This implied the utilisation of several technologies and methods to adapt the higher-level agent world to the lower-level physical manufacturing equipment. By showing that the multiagent paradigm can be applied to real shop floor equipment in a realistic way, this thesis contributed to closing the gap between the people from the shop floor (industry) and the ones from the agent oriented paradigm (mainly researchers). In fact, this may be considered disseminating agents as a valid paradigm for solving real shop floor problems within industry.

2. **Implementation of electronic contracts**. CoBASA supports contract modelling, negotiation, performance, and termination. Although the vast amount of work on contracts has been carried out within the e-commerce area, the real fact is that there still are few implementations of electronic contracts. Although CoBASA does not address this particular area, it implemented a contract-based system.

3. **Utilisation of FIPA based protocols and FIPA ACL language**. CoBASA agent communication is carried out using solely FIPA ACL language. The different interactions that occur between CoBASA agents are essentially supported on FIPA protocols. This is an important contribution because despite the vast work developed by FIPA its standards are not yet widely used, especially in the manufacturing world. Therefore, using FIPA standards

contributes to prove its adequateness and validity to be used in multiagent applications applied to the manufacturing domain.

4. **Integration of different software paradigms**. CoBASA agents could not be implemented with available commercial tools. Hence the work done to integrate different tools into an applicable architecture. The agents were developed integrating the Java compiler, Jade, Protégé-2000, a Prolog interpreter (Jinni) written in Java, and the Java parser and lexer JavaCC. In addition middleware such as DCOM and CORBA was used to integrate the various manufacturing controllers in CoBASA. These legacy manufacturing controllers required additional programming effort to transform them from standalone units into ones that could be controlled by remote hosts (computers) in order to be agentified.

- **Other Contributions**

1. **The concept that manufacturing equipment should be designed taking into consideration the support for its life cycle**. Continuous changes to the shop floor are better supported if both new and existing controllers provide functionalities to enable the transition from one life cycle phase to another. These functionalities help the composition of different manufacturing systems, using individual manufacturing equipment. If manufacturing equipment already includes these functionalities it will be much easier to have agile shop floor systems. This is a recommendation that is being disseminated through the forums in which the author is participating (conferences, projects, etc) to the companies that produce manufacturing equipment.

2. **The concept that legacy controllers must be included**. The ideas behind this thesis can only be applied successfully within industry if existing controllers may be integrated within the new architecture. It has been shown that an approach has been developed to integrate legacy controllers within the CoBASA framework (chapter 5). The emphasis on this point might seem inflated from the research point of view. However, for the people from industry, this is one of the most important factors when taking decisions regarding changes in shop floor architectures. If a shop floor controller cannot adapt older systems to its framework, it will most probably not allow thorough upgrades of its own structure.

3. **The finding that there it is not possible to create truly agile shop floor systems in the assembly domain without no well-established definition or classification of the assembly process**. This became evident when CoBASA agents started to be implemented. This is in accordance with the findings of other researchers who have also achieved the same conclusion (Onori, 2002; Tichem, 2000; Vos, 2001) when tried to find solutions for the design and implementation of truly flexible assembly systems using a more mechanical-oriented

approach. The basic agentified manufacturing components that can be used to compose assembly systems can only be defined properly if the assembly process is well structured and classified. Only if the basic components are well defined and structured it will be possible to use these components interchangeable in the creation of different types of assembly systems.

4. These findings resulted in EUPASS – Evolvable Ultra-Precision Assembly Systems, which is a European project submitted to the 6th framework program, in which special attention is devoted to the aspect of finding definitions and structuring the assembly process. The author was invited to contribute to this project on the basis of the results given in this thesis.

Prototypes

The following prototypes were developed along the period covered by this thesis:

1. **Integration of real manufacturing controllers**. The various manufacturing components (robots, tools, transportation system, automatic warehouse, etc.) that compose the NovaFlex assembly cell were integrated and transformed into distributed objects using DCOM and CORBA. Without this integration, it would have been impossible to connect the agents to the real manufacturing components.

2. **The CoBASA prototype**. This prototype included the implementation of the various agents that compose the architecture. Therefore, it implemented the following agents: the cluster manager agent, the broker, the generic agent, and the agent machine interface. The mechanisms used by the agents to negotiate/renegotiate contracts and to ensure their performance are included in each of the CoBASA agents.

Publications

The results achieved during the period of this thesis were submitted to international scientific conferences (PROVE-2000 (Barata & Camarinha-Matos, 2000), WESIC-2001 (Barata et al., 2001), LCA-2001 (Camarinha-Matos & Barata, 2001; Leitão, Barata, Camarinha-Matos, & Boissier, 2001), PROVE-2002 (Barata & Camarinha-Matos, 2002a), ISR-2002 (Barata & Camarinha-Matos, 2002d), HOLOMAS-2002 (Barata & Camarinha-Matos, 2002c), CARS&FOF-2002 (Barata & Camarinha-Matos, 2002b), ISATP-2003 (Onori, Camarinha-Matos, & Barata, 2003b)) to be validated by the scientific community. Only refereed conferences were considered. In addition, four publications have been submitted and accepted for publication in the following journals:

- The International Journal of Computer Integrated Manufacturing (Barata & Camarinha-Matos, 2004; Camarinha-Matos, Pinheiro-Pita, Rabelo, & Barata, 1995)

- The International Journal of Networks and Virtual Organisations (Barata & Camarinha-Matos, 2003)
- The Assembly Automation Journal (Onori, Camarinha-Matos, & Barata, 2003a)
- IEEE Transactions on Robotics and Automation (Camarinha-Matos, Seabra Lopes, & Barata, 1996).

New Project Proposals

Some of the concepts and ideas developed in this thesis have been used in various European project proposals that were submitted to the 6th Framework Program on Research, Technological Development and Demonstration of the European Union. The following projects are considered:

- **OENSY** – Optimisation and Reengineering of Production Systems (FP6-2002.SME.1)
- **EUPASS** – Evolvable Ultra-Precision Assembly Systems (*FP6-2002-IST-NMP – Priority 2 & 3 Joint Call IST-NMP – Expanding Manufacturing Knowledge in Micro-assembly in Europe*)
- **RAMP-UP** – Environment for Syncronising Production RAMP-UP Processes in the Value Chain (*FP6-2002-IST-NMP – Priority 2 & 3 Joint Call IST-NMP – Manufacturing, Products and Services Engineering in 2010*)
- **EMAT** – A Distributed Network of Excellence for Evolvable Micro Assembly Technology (*IST NoE*).

6.1.2 Obstacles to Industrial Dissemination

Beyond the technological and methodological issues covered in this thesis, the main barriers to its industrial application have been considered. These may be summed up as follows:

1. The dominating culture among shop floor people which still consider that anything, even if slightly related to AI, is just theory without any chance of being applicable.

2. The large number of outdated legacy controllers and equipment that renders the equipment retrofitting quite challenging and difficult.

3. The large numbers of technicians, who although competent and well trained, are too accustomed to old architectures, and, therefore, have little or no will at all to accept or lead processes for the introduction of new architectures. This is partially due to the fact that they are too absorbed with introducing changes to, or maintaining the output of, production systems!

4. Agent programming is not yet mature enough when compared to other software engineering paradigms. Agent oriented software engineering is still a programming area for a few specialists. For instance, the number of agent oriented methodologies for software design is

yet small, and not standardised at all. Furthermore, despite the fact that the number of agent development frameworks has increased in recent years, they still miss many supporting functionalities and tools to help programmers. The number of details that must be taken into consideration by the programmers is vast, and, furthermore, several tools need to be integrated when implementing real applications. Please note that CoBASA only requires programming in some circumstances (when a MRA agent is first created). However, since CoBASA is based on the novel multiagent paradigm, people even if they do not have to use it much tend to be suspicious of it and try to avoid it as much as they can. This is particular relevant in the case of industry.

5. The lack of well established definitions, or classifications, of the production processes. One process where this lack is significantly felt is assembly.

6. The reluctance of the big shop floor control equipment suppliers to accept and commercialise new paradigms among industry. Although some suppliers have already started research about applying the multiagent paradigm to their controllers, others are yet far from that point, even if some signs[35]show that they are trying to change this situation.

As a conclusion it may be also important to state that a PhD thesis is not supposed to develop finished products or robust technology.

6.2 SUGGESTIONS FOR FURTHER RESEARCH

As with any PhD thesis this work cannot be considered completely finalised, and it has given rise to new ideas for further research. The most important aspects needing further research and new developments are highlighted in the next paragraphs:

1. **Automatic support for coalition formation**. The process of coalition formation needs to be gradually rendered more automatic. This means that the broker agent should be more autonomous in taking decisions on the user's behalf, and able to make reliable suggestions about possible alternatives. For instance, based on the functionalities and requirements introduced by the systems integrator, the broker should suggest what would be the most appropriate coalition. *This functionality implies further research on specification of requirements and goal definition*. The characterisation and definition of which goals are relevant to create manufacturing coalitions is mandatory because suggestions can only be

[35] One of these signs is the Siemens leadership in the RAMP-UP project proposal submitted recently to the 6th framework program, which clearly states the use of the multiagent paradigm as the programming paradigm.

made if the goal is perfectly characterised in advance. The requirements are associated to the goal by detailing it.

2. **Improve error recovery within the generic agents**. The error recovery process that occurs when a coordinator agent is executing a complex skill must be improved to accommodate the possibility of asking help from another coalition member with similar functionalities, in case the coalition member being requested returns an error. For instance, in a coalition composed of one robot and two different grippers with similar functionality, consider that the coordinator is executing the complex skill *pick&place()*, which is supported by the robot and the gripper 1; if the gripper 1 for some reason fails, the coordinator should be able to transfer the requests to the gripper 2. Once again, this requires a precise knowledge of the manufacturing processes being executed, which was beyond the scope of this thesis and is now a well-detailed problem area for the global R&D community.

3. **Increase automation on the skill composition process**. The composition of skills is generated automatically in each generic agent by rules that are manually codified by users. If the set of complex skills do not change it is not necessary to make any alterations and the generic agents normally generate the skills. If, on the other hand, new skills need to be added or existing skills must be changed, this requires the rules to be rewritten. It is this process of changing or creating new rules that needs to be automatic in order to become simpler and more user friendly.

4. **Using other agent models**. Although the agent model used in the current CoBASA implementation proved to be adequate, testing and evaluating the performance of CoBASA agents under other agent models should not be left out. One of the models the author proposes as important to test and evaluate is BDI. The main reason for this lies within the BDI appropriateness for rational agents. Furthermore, it would be interesting to compare the BDI model with the currently used one in terms of efficiency.

5. **Connecting CoBASA with scheduling and planning tools**. CoBASA needs further work to link the created manufacturing coalitions with the higher level planning tools at execution time. The highest manufacturing coalition receives orders from a scheduler or planner. These executing orders must be harmonised with the orders that each CoBASA coalition offers.

6. **Further research on the shop floor reengineering methodology**. This thesis has been focused on providing an agile shop floor supervision architecture, which is part of the global shop floor reengineering methodology. The methodology here sketched needs to be furthered developed and improved, and its different activities and phases characterised and defined. This

activity has been planned as one of the main actions to be taken within the projects whose proposals have been already submitted.

7. **Supporting tools for the reengineering methodology**. This activity is directly connected with the previous one. The goal is to develop computer based tools to support the activities defined for the shop floor reengineering activity. One of the most important tools is the e-Book, which will be the main interface for the whole computer based tools to cover the entire shop floor life cycle.

8. **Improve the library of agentified manufacturing equipment**. More manufacturing equipment needs to be modelled. The final goal of this activity is to create of library of agentified manufacturing components ready to be instantiated as manufacturing agents. This requires a wider collaboration involving the producers of manufacturing equipment.

9. **Standardisation work under FIPA**. Multiagents applied to manufacturing need further standardisation work. This activity has been planned as one of the main actions to be taken within the recently created FIPA Holonic Enterprises working group. A standardisation effort is required at manufacturing agent level. This work should cover, among others, the following topics: types of manufacturing agents, message types, supported protocols, manufacturing ontologies, and connecting standards between the agents and physical equipments (agentification).

6.3 CONCLUDING REMARKS

The essence of this thesis could have focussed more elaborately on a specific issue, such as the development of theories and software architectures. It could have even delved into the more intricate issues of collaborative networks or virtual organisations, creating new ideas, approaches or theories. All of this could be discussed at length, but the major objective from the very outset was not to broaden the divide between industry and the research community but, rather, to attempt to bridge it. Hence the detailed account given of the industrial needs, the current drawbacks and the inadequate approaches. This detailed account may also be seen as the development of the list of real operative requirements, which were later used for the development of the CoBASA architecture. This thesis must therefore be viewed as a work of applied research, in which the latest R&D developments have been brought a small step closer to the industry at large, with an applicable architecture as a result. The comforting thought is that, although problems still persist, a considerable amount of interest has been aroused within the production engineering community, which was the targeted group to begin with, and the work will be continued within the several European initiatives detailed earlier.

References

Abernethy, N. F., Wu, J. J., Hewet, M., & Altman, R. B. (1999). SOPHIA: A Flexible, Web-Based Knowledge Server. *IEEE Intelligent Systems & Their Applications, 14*(4), 79-85.

Adlemo, A., Andréasson, S.-A., Fabian, M., Gullander, P., & Lennartson, B. (1995). Towards a True Flexible manufacturing System. *Control Engineering Practice, 3*(4), 545-554.

AgentBuilder. (2001). *http://www.agentbuilder.com/* [web site]. Retrieved Jan 2002, 2002, from the World Wide Web:

Agha, G., Jamali, N., & Varela, C. (2001). Agent Naming and Coordination: Actor BasedModels and Infrastructures. In A. Omicini & F. Zambonelli & M. Klusch & R. Tolksdorf (Eds.), *Coordination of Internet Agents: Models, Technologies, and Applications* (pp. 225-246). Berlin: Springer-Verlag.

Agre, P., & Chapman, D. (1987). *PENGI: An Implementation of aTheory of Activity.* In Proceedings of 6th National Conference on Artificial Intelligence (AAAI-87), Seattle - WA.

Albus, J. S., Lumia, R., Fiala, J., & Wavering, A. (1989, October 4-6). *NASREM -- The NASA/NBS Standard Reference Model for Telerobot Control System Architecture.* In Proceedings of 20th International Symposium on Industrial Robots, Tokyo, Japan.

Allen, R. K. (1997). *Journal of Workplace Learning, 9*(1), 34-42.

Allen, T., & Widdison, R. (1996). Can Computers Make Contracts? *Harvard Journal of Law and Technology, 9*(1), 26-54.

Almeida, C. F. (2000). *Contratos I - Conceitos; Fontes; Formação*. Coimbra: Almedina.

Altman, R. B. (2001). Challenges for Intelligent Systems in Biology. *IEEE Intelligent Systems, 16*(6), 14-18.

Altman, R. B., Bada, M., Chai, X. J., Carillo, M. W., Chen, R. O., & Abernethy, N. F. (1999). RiboWeb: An Ontology-Based System for Collaborative Molecular Biology. *IEEE Intelligent Systems & Their Applications, 14*(5), 68-76.

Anderson, D. R., Sweeney, D. J., & Williams, T. A. (1986). *Quantitative Methods for Business*. St. Paul, MN: West Publishing Company.

Angelov, S., & Grefen, P. (2001). *B2B eContract Handling - A Survey of Projects, Papers and Standards* (TR 01-21). Twente: University of Twente - CTIT.

Arentsen, A. L. (1995). *A Generic Architecture for Factory Activity Controls*. Unpublished PhD thesis, University of Twente, Enschede.

AS-I. (2002). *AS-Interface* [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.as-interface.com/

Assembly-Net. (2002). *Assembly-Net - Precision Assembly Technologies for Mini and Micro Products (Growth Thematic Network)* [Web Site]. Retrieved Nov 2002, from the World Wide Web: http://www.assembly-net.org

AUML. (2002). *Agent Unified Modelling Language* [web site]. Retrieved October 2002, 2002, from the World Wide Web: http://www.auml.org

Austin, J. L. (1955). *How to Do Things with Words* (2nd ed.). Oxford: Oxford University Press.

Axelrod, R. M. (1984). *The Evolution of Co-operation*. London: Penguin Books.

Axelrod, R. M. (1997). *The Complexity of Cooperation: agent-based models of competition and collaboration*. Princeton, N.J.: Princeton University Press.

Balzer, W., & Tuomela, R. (2001). Social Institutions, Norms, and Practices. In R. Conte & C. Dellarocas (Eds.), *Social Order in Multiagent Systems* (pp. 161-180). Boston: Kluwer Academic Publishers.

Banville, M. (1996). Sonia: An Adaptation of Linda for Coordination of Activities in Organisations. In P. Ciancarini & C. Hankin (Eds.), *Proceedings of the 1st International Conference on Coordination Models and Languages* (Vol. 1061 - LNCS). Berlin: Springer-Verlag.

Barata, J. (1995). *Modelação e Integração em Sistemas Flexíveis de Produção*. Unpublished Master Thesis, Universidade Nova de Lisboa, Monte de Caparica.

Barata, J., & Camarinha-Matos, L. M. (1994). Development of a FMS/FAS System. *Studies in Informatics and Control, 3*(2-3), 231-239.

Barata, J., & Camarinha-Matos, L. M. (1995a). Dynamic Behaviour Objects in Modelling Manufacturing Processes. In Q. Sun & Z. Tang & Y. Zhang (Eds.), *Computer Applications in Production Engineering* (Vol. 1, pp. 499-508). London: Chapman & Hall.

Barata, J., & Camarinha-Matos, L. M. (1995b). Integration of Object-Oriented Programming and Petri Nets for Modelling and Supervision of FMS/FAS. In L. M. Camarinha-Matos (Ed.), *Architectures and Design Methods for Balanced Automation Systems* (Vol. 1). New York: Chapman & Hall.

Barata, J., & Camarinha-Matos, L. M. (2000). Shopfloor Reengineering To Support Agility in Virtual Enterprise Environments. In L. M. Camarinha-Matos & H. Afsarmanesh & R. Rabelo (Eds.), *E-Business and Virtual Enterprises* (Vol. 1, pp. 287-291). London: Kluwer Academic Publishers.

Barata, J., & Camarinha-Matos, L. M. (2002a). Contract Management in Agile Manufacturing Systems. In L. M. Camarinha-Matos (Ed.), *Collaborative Business Ecosystems and Virtual Enterprises* (Vol. 1, pp. 109-122). New York: Kluwer Academic Publishers.

Barata, J., & Camarinha-Matos, L. M. (2002b). *Contract-Governed Agents in Agile Manufacturing*. In Proceedings of CARs&FOF'2002 - 18th International Conference on CAD/CAM, Robotics, and Factories of the Future, Porto, Portugal.

Barata, J., & Camarinha-Matos, L. M. (2002c). *Implementing a Contract-based Multi-Agent Approach for Shop Floor Agility*. In Proceedings of Holomas 2002, Aix-en-Provence.

Barata, J., & Camarinha-Matos, L. M. (2002d). *Shop Floor Re-engineering Using Agents*. In Proceedings of ISR2002 - 33rd International Symposium on Robotics, Stockholm, Sweden.

Barata, J., & Camarinha-Matos, L. M. (2003). Coalitions of Manufacturing Components for Shop Floor Agility - The CoBaSA Architecture. *International Journal of Networking and Virtual Organisations, 2*(1), 50-77.

Barata, J., & Camarinha-Matos, L. M. (2004). Multiagent Coalitions in Shop Floor Reengineering. *To appear in International Journal of Computer Integrated Manufacturing*.

Barata, J., Camarinha-Matos, L. M., Boissier, R., Leitão, P., Restivo, F., & Raddadi, M. (2001, 27-29 Jun). *Integrated and Distributed Manufacturing, a Multi-Agent Perspective*. In Proceedings of WESIC 2001 - 3rd Workshop on European Scientific and Industrial Collaboration, Enschede, The Netherlands.

Barata, J., Camarinha-Matos, L. M., & Chavarria, J. F. R. (1994). Modelling, Dynamic Persistence and Active Images for Manufacturing Process. *Studies in Informatics and Control, 3*(2-3), 173-183.

Barata, J., Camarinha-Matos, L. M., & Colombo, A. W. (2001). Uma Aproximação Baseada em Componentes para Integrar Recursos Software e Hardware de Manufactura, *7as Jornadas Hispano Lusas de Ingenieria Eléctrica* (Vol. 4, pp. 291-296). Leganés (Madrid) - España: Universidad Carlos III de Madrid.

Barata, J., Camarinha-Matos, L. M., & Freitas, P. (1999). Integração de um Sensor com Magneto-Resistência Gigante em Células Flexíveis de Produção. In H. Duarte-Ramos & A. Leão-Rodrigues (Eds.), *Engenharia Electrotécnica Luso-Espanhola* (Vol. 4, pp. 489-496). Lisboa: EDINOVA.

Barata, J., Vieira, W., & Camarinha-Matos, L. M. (1996). *Integration and MultiAgent Supervision of Flexible Manufacturing Systems*. In Proceedings of Mechatronics'96 - The 5th UK Mechatronics Forum International Conference, Guimarães - Portugal.

Bartezzaghi, E. (1999). The Evolution of Production Models: Is a New Paradigm Emerging? *International Journal of Operations & Production Management, 19*(2), 229-250.

Barwise, J., & Etchemendy, J. (1992). *The Language of First-order Logic : including the IBM-compatible Windows version of Tarski's world 4.0* (3rd ed.). Stanford, CA: Center for the Study of Language and Information.

Bauer, A., Bowden, R., Browne, J., Duggan, J., & Lyons, G. (1991). *Shop Floor Control Systems: from design to implementation*. London ; New York: Chapman & Hall.

Beam, C., & Segev, A. (1997). Automated Negotiations: A Survey of the State of the Art. *Wirtschaftsinformatik, 39*(3), 263-268.

Beck, U., & Ritter, M. (1992). *Risk Society : towards a new modernity*. London: Sage.

Bell, D. (1976). *The Coming of Post-Industrial Society : a venture in social forecasting* (New ed.). Harmondsworth: Penguin.

Bell, D. (1981). The Social Framework of the Information Society. In T. Forester (Ed.), *The Microelectronics Revolution: The Complete Guide to the New Technology and Its Impact on Society* (pp. 500-550). Cambridge, Ma.: MIT Press.

Bell, D. (2000). *The End of Ideology: On the Exhaustion of Political Ideas in the Fifties - with a New Introduction*. Cambridge, Mass.: Harvard University Press.

Bellavista, P., & Magedanz, T. (2001). Middleware Technologies: CORBA and Mobile Agents. In A. Omicini & F. Zambonelli & M. Klusch & R. Tolksdorf (Eds.), *Coordination of Internet Agents: Models, Technologies, and Applications* (pp. 110-152). Berlin: Springer-Verlag.

Bellifemine, F., Caire, G., Trucco, T., & Rimassa, G. (2003). *JADE Programmer's Guide*.

Bellifemine, F., Poggi, A., & Rimassa, G. (2001). Developing Multi-Agent Systems with a FIPA-Compliant Agent Framework. *Software-Practice & Experience, 31*(2), 103-128.

Bellifemine, F., Poggi, A., & Rimassa, G. (2001). Developing Multiagent Systems with JADE. In C. Castelfranchi & Y. Lespérance (Eds.), *Intelligent Agents VII - Agent Theories Architectures and Languages* (Vol. LNCS 1986, pp. 89-102). Berlin: Springer-Verlag.

Berggren, C. (1992). *Alternatives to Lean Production: work organization in the Swedish auto industry*. Ithaca, N.Y.: ILR Press.

Berggren, C. (1993). *The Volvo Experience: alternatives to lean production in the Swedish auto industry*. London: Macmillan.

Bergman, E. M., & Feser, E. J. (1999). *Industrial and Regional Clusters: Concepts and Comparative Applications*. Regional Research Institute - WVU. Retrieved 19 Sep 2002, 2002, from the World Wide Web: http://www.rri.wvu.edu/WebBook/Bergman-Feser/contents.htm

Berners-Lee, T., & Fischetti, M. (1999). *Weaving the Web: the past, present and future of the World Wide Web*. London: Orion Business.

Berners-Lee, T., Hendler, J. A., & Lassila, O. (2001). The Semantic Web. *Scientific American, 284*(5), 34-43.

Bernhardt, R. (Ed.). (1992). *Integration of Robots into CIM* (1. ed.). London-New York: Chapman & Hall;Van Nostrand.

Berquist, A., & Berquist, K. (1996). *Managing Information Highways: the PRISM book :principles, methods and case studies for designing telecommunications management systems*. Berlin: Springer-Verlag.

Biazzo, S., & Panissolo, R. (2000). The Assessment of Work Organization in Lean Production: the relevance of the worker's perspective. *Integrated Manufacturing Systems, 11*(1), 6-15.

Bichler, M., Beam, C., & Segev, A. (1998). Services of A Broker in Electronic Commerce Transactions. *International Journal of Electronic Markets, 8*(1), 27-31.

Biemans, F. P., & Vissers, C. A. (1989). Reference Model for Manufacturing Planning and Control Systems. *Journal of Manufacturing Systems, 8*(1), 35-46.

Bishop, P., & Warren, N. (2003). *JavaSpaces in Practice*. London: Addison-Wesley.

Bluetooth. (2003). *The official bluetooth web site* [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.bluetooth.com

Boari, C. (2001). *Industrial Clusters, Focal Firms, and Economic Dynamism: a Perspective from Italy* (Stock No 37186). Washington, D.C.: World Bank Institute.

Boella, G., & Lesmo, L. (2001). Deliberate Normative Agents. In R. Conte & C. Dellarocas (Eds.), *Social Order in Multiagent Systems* (pp. 85-110). Boston: Kluwer Academic Publishers.

Bongaerts, L., Monostori, L., McFarlane, D. C., & Kádár, B. (2000). Hierarchy in Distributed Shop Floor Control. *Computers in Industry, 43*(2), 123-137.

Bongaerts, L., Van Brussel, H., & Valckenaers, P. (1999, 24-26 June). *Generic Concepts for Holonic Manufacturing Control.* In Proceedings of FAIM99 - 9th International Conference on Flexible Automation and Intelligent Manufacturing, Tilburg.

Bratko, I. (2001). *Prolog Programming for Artificial Intelligence* (3. ed.). Harlow: Addison-Wesley.

Bratman, M. E. (1987). *Intention, Plans, and Practical Reason*. Cambridge - Mass. ; London: Harvard University Press.

Bratman, M. E., Israel, D. J., & Pollack, M. E. (1988). Plans and Resource Bounded Pactical Reasoning. *Computational Intelligence, 4*, 349-355.

Brennan, R. W., Fletcher, M., & Norrie, D. H. (2002). An Agent-Based Approach to Reconfiguration of Real-Time Distributed Control Systems. *IEEE Transactions on Robotics and Automation, 18*(4), 444-451.

Brill, D. (1993). *Loom Reference Manual - Version 2.0*: University of Southern California.

Brooks, R. A. (1986). A Robust Layered Control System for a Mobile Robot. *IEEE Transactions on Robotics and Automation, 2*(1), 14-23.

Brooks, R. A. (1991a). Intelligence Without Representation. *Artificial Intelligence, 47*, 139-159.

Brooks, R. A. (1991b). New Approaches to Robotics. *Science*(253), 1227-1232.

Brooks, R. A. (1999). *Cambrian Intelligence: the early history of the new AI*. Cambridge, Mass.: MIT Press.

Brooks, R. A. (2002). *Flesh and Machines: how robots will change us* (1. ed.). New York: Pantheon.

Brown, J. S., & Duguid, P. (2000). *The Social Life of Information*. Boston, Mass.: Harvard Business School.

Browne, J., Harhen, J., & Shivnan, J. (1988). *Production Management Systems: a CIM perspective.* Wokingham: Addison-Wesley.

Brynjolfsson, S., & Arnström, A. (1990). Error detection and Recovery in Flexible Assembly Systems. *International Journal of Advanced Manufacturing Technology, 5*, 112-125.

Builder, C. (1993). Is it a Transition or a Revolution? *Futures*, 155-168.

Bussmann, S., & McFarlane, D. C. (1999). *Rationales for Holonic Manufacturing.* In Proceedings of Second International Workshop on Intelligent Manufacturing Systems, Leuven, Belgium.

Byrne, J. A., Brandt, R., & Port, O. (1993). The Virtual Corporation: the company of the future will be the ultimate in adaptability. *Business Week*(February, 8), 98-102.

Cabri, G., Leonardi, L., & Zambonelli, F. (2000). MARS: A Programmable Coordination Architecture for Mobile Agents. *IEEE Internet Computing*.

Cahil, E., Garello, P., & Jering, D. (2001). *Technology Maps for Manufacturing, Production and Services*: European Commission - IMS Secretariat.

Camarinha-Matos, L., & Afsarmanesh, H. (1996). *Balanced Automation Systems II - Implementation Challenges for Anthropocentric Manufacturing*. Boston, Mass.: Chapman & Hall.

Camarinha-Matos, L., Afsarmanesh, H., & Erbe, H.-H. (2000). *Advances in Networked Enterprises: virtual organizations, balanced automation, and systems integration*. Boston, Mass. ; London: Kluwer Academic.

Camarinha-Matos, L. M. (2002a). Virtual Organisations in Manufacturing, *Proceedings of FAIM 2002, 12th International Conf. on Flexible Automation and Intelligent Manufacturing* (pp. 1036-1054). Munich: Oldenbourg.

Camarinha-Matos, L. M. (Ed.). (1995). *Balanced Automation I - Architectures and Design Methods for Balanced Automation Systems* (Vol. 1). New York: Chapman & Hall.

Camarinha-Matos, L. M. (Ed.). (2002b). *Collaborative Business Ecosystems and Virtual Enterprises: Third Working Conference on Infrastructures for Virtual Enterprises (PRO-VE'02)*: Kluwer: Boston.

Camarinha-Matos, L. M., & Afsarmanesh, H. (1999a). The Virtual Enterprise Concept. In L. M. Camarinha-Matos & H. Afsarmanesh (Eds.), *Infrastructures for Virtual Enterprises: networking industrial enterprises : IFIP TC5 WG5.3/PRODNET Working Conference on Infrastructures for Virtual Enterprises (PRO-VE '99)* (pp. 3-14). Boston: Kluwer Academic.

Camarinha-Matos, L. M., & Afsarmanesh, H. (Eds.). (1999b). *Infrastructures for Virtual Enterprises: networking industrial enterprises : IFIP TC5 WG5.3/PRODNET Working Conference on Infrastructures for Virtual Enterprises (PRO-VE '99)*. Boston: Kluwer Academic.

Camarinha-Matos, L. M., & Afsarmanesh, H. (Eds.). (2002). *Interim Green Report on New Collaborative Forms and their Needs*: THINKcreative project.

Camarinha-Matos, L. M., Afsarmanesh, H., & Marik, V. (1998). *Intelligent systems for manufacturing: multi-agent systems and virtual organizations*. Boston: Kluwer.

Camarinha-Matos, L. M., Afsarmanesh, H., & Rabelo, R. J. (Eds.). (2001). *E-business and Virtual Enterprises: managing business-to-business cooperation*. Boston ; London: Kluwer Academic.

Camarinha-Matos, L. M., & Barata, J. (2001). Contract-Based Approach for Shop-Floor Re-engineering. In R. Bernhardt & H. H. Erbe (Eds.), *Cost Oriented Automation* (First ed., Vol. 1, pp. 141-148). Oxford - UK: Pergamon - Elsevier.

Camarinha-Matos, L. M., Barata, J., & Flores, L. (1997). Shopfloor Integration and MultiAgent Supervision. In I. Rudas (Ed.) (Vol. 1, pp. 457-462).

Camarinha-Matos, L. M., Pinheiro-Pita, H., Rabelo, R., & Barata, J. (1995). Towards a Taxonomy of CIM Activities. *International Journal of Computer Integrated Manufacturing, 8*(3), 160-176.

Camarinha-Matos, L. M., Seabra Lopes, L., & Barata, J. (1994, 8-14 May). *Execution Monitoring in Assembly with Learning Activities.* In Proceedings of ICRA'94 - IEEE International Conference on Robotics and Automation, San Diego - USA.

Camarinha-Matos, L. M., Seabra Lopes, L., & Barata, J. (1996). Integration and Learning in Supervision of Flexible Assembly Systems. *IEEE Transactions on Robotics and Automation (Special Issue on Assembly and Task Planning), 12*(2), 202-219.

Camarinha-Matos, L. M., & Vieira, W. (1999). Intelligent Mobile Agents in Elderly Care. *Journal of Robotics and Autonomous Systems, 27*(1-2), 59-75.

Cammarano, J. L. (1997). *Lessons to Be Learned Just In Time*. Georgia: Engineering and Management Press.

CAN. (2002). *CAN - Controller Area Network* [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.can.bosch.com/

Carriero, N., & Gelernter, D. (1986). The S/Net's Linda Kernel. *ACM Transactions on Computer Systems, 4*(2), 110-129.

Carriero, N., & Gelernter, D. (1989). How to Write Parallel Programs: A Guide to the Perplexed. *Acm Computing Surveys, 21*(3), 323-357.

Carriero, N., & Gelernter, D. (1990). Tuple Analysis and Partial Evaluation Strategies in the LINDA Precompiler. In D. Gelernter & A. Nicolau & D. Padua (Eds.), *Languages and Compilers for Parallel Computing* (pp. 114-125): MIT Press.

Castel, F. (2002). Viewpoint: Ontological Computing. *Communications of the ACM, 45*(2), 29-30.

Castelfranchi, C. (1998). Modelling Social Action for AI Agents. *Artificial Intelligence, 103*, 157-182.

Castelfranchi, C., Micelli, M., & Cesta, A. (1992). Dependence Relations Among Autonomous Agents. In E. Werner & Y. Demazeau (Eds.), *Decentralized A.I. 3* (pp. 215-227). Amsterdam, NL: Elsevier Science Publishers B. V.

Castells, M. (1996). *The Information Age: Economy, Society and Culture - the rise of the network society* (Vol. 1). Cambridge, Mass.; Oxford, UK: Blackwell.

Castells, M. (1997a). *The Information Age: Economy, Society and Culture - the end of the millennium* (Vol. 3). Cambridge, Mass.; Oxford, UK: Blackwell.

Castells, M. (1997b). *The Information Age: Economy, Society and Culture - the power of identity* (Vol. 2). Cambridge, Mass.; Oxford, UK: Blackwell.

CE-NET. (2002). *CE-NET - Concurrent Enterprising Network of Excellence (IST Project)* [Web Site]. Retrieved Nov 2002, from the World Wide Web: http://www.ce-net.org

Chalmers, A. F. (1999). *What is this Thing Called Science?* Buckingham: Open University Press.

Chandrasekaran, B., Josephson, J. R., & Benjamins, V. R. (1999). What are ontologies, and why do we need them? *Ieee Intelligent Systems & Their Applications, 14*(1), 20-26.

Chang, T.-C., Wysk, R. A., & Wang, H.-P. (1998). *Computer-Aided Manufacturing* (2. ed.). Upper Saddle River, N.J.: Prentice Hall.

Chapman, D. (1990). Planning for Conjunctive Goals. In J. Allen & J. A. Hendler & A. Tate (Eds.), *Readings in Planning* (pp. 537-558). San Mateo, Calif.: Morgan Kaufmann Publishers.

Chase, R. B., & Aquilano, N. J. (1989). *Gestão da Produção e das Operações - Perspectiva do ciclo de vida*. Lisboa: Monitor.

Chaudhri, V. K., Farquhar, A., Fikes, R., Karp, P. D., & Rice, J. P. (1998). *OKBC: A Programatic Foundation for Knowledge base Interoperability.* In Proceedings of AAAI-98, Madison, WI, USA.

Cheng, J. M. J., Simmons, J. E. L., & Ritchie, J. M. (1997). Manufacturing System Flexibility: the "capability and capacity" approach. *Integrated Manufacturing Systems, 8*(3), 147-158.

Chiu, E. (2002). *EbXML Simplified: a guide to the new standard for global e-commerce*. New York: John Wiley.

Chung, S. J. (Ed.). (1996). *Information Highways for a Smaller World and Better Living: Proceedings of ICCC'95, (12th International conference on computer communication)*. Amsterdam: IOS Press.

Ciancarini, P., Tolksdorf, R., Vitali, F., Rossi, D., & Knoche, A. (1998). Coordinating Multiagent Applications on the WWW: A Reference Architecture. *IEEE Transactions on Software Engineering, 24*(5), 362-375.

Clarke, T., & Clegg, S. (1998). *Changing Paradigms - The transformation of Management Knowledge for the 21st Century*. London: HarperCollinsBusiness.

Cohen, P. R., Cheyer, A., Wang, M., & Baeg, S. C. B. (1998). An Open Agent Architecture. In M. N. Huhns & M. P. Singh (Eds.), *Readings in Agents* (pp. 197-204). San Francisco, CA, USA: Morgan Kaufmann Publishers.

Cohen, P. R., & Levesque, H. J. (1990). Rational Interaction as the Basis for Communication. In P. R. Cohen & J. Morgan & M. E. Pollack (Eds.), *Intentions in Communication*. Cambridge, MA: MIT Press.

Colombo, A. W., Camarinha-Matos, L. M., & Barata, J. (1999). An Approach for Design Optimisation and Implementation of Flexible Production Systems Integrating Discrete Event and 3-D Kinematics Simulations. In H. Szczerbicka (Ed.), *Modelling and Simulation: A Tool for the Next Millennium* (Vol. 1, pp. 461-467). Warsaw: SCS.

CommerceNet (2003). [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.commerce.net/

Conant, R. C. (1976). Laws of Information Which Govern Systems. *IEEE Transactions on Systems, Man and Cybernetics, 6*(4), 334-338.

Conte, R., & Castelfranchi, C. (2001). Are Incentives Good Enough To Achieve (Info) Social Order? In R. Conte & C. Dellarocas (Eds.), *Social Order in Multiagent Systems* (pp. 45-61). Boston: Kluwer Academic Publishers.

Conte, R., & Dellarocas, C. (2001a). Social Order in Info Societies: An Old Challenge for Innovation. In R. Conte & C. Dellarocas (Eds.), *Social Order in Multiagent Systems* (pp. 1-15). Boston: Kluwer Academic Publishers.

Conte, R., & Dellarocas, C. (Eds.). (2001b). *Social Order in Multiagent Systems*. Boston: Kluwer Academic Publishers.

Conte, R., & Dignum, F. (2001). From Social Monitoring to Normative Influence. *Journal of Artificial Societies and Social Simulation, 4*(2), <http://jasss.soc.surrey.ac.uk/4/2/7.html>.

ControlNet (2002). [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.controlnet.org

CORBA. (2001). *Object Management Group. CORBA Specification. http://www.omg.org* [web site]. Retrieved Jan 2002, 2002, from the World Wide Web:

Corbin, J. R. (1990). *The Art of Distributed Applications: programming techniques for remote procedure call*. New York: Springer-Verlag.

Cortés, U., & Dignum, F. (Eds.). (2001). *Agent-Mediated Electronic Commerce III: current issues in agent based electronic commerce systems* (Vol. LNAI 2003). Berlin: Springer.

Cost, R. S., Labrou, Y., & Finin, T. (2001). Coordinating Agents using Agent Communication Languages Conversations. In A. Omicini & F. Zambonelli & M. Klusch & R. Tolksdorf (Eds.), *Coordination of Internet Agents: Models, Technologies, and Applications* (pp. 183-196). Berlin: Springer-Verlag.

COVE. (2002). *COVE - COoperation infrastructure for Virtual Enterprise and electronic business* [Web Site]. Retrieved Nov 2002, from the World Wide Web: http://www.uninova.pt/~cove

Cremonini, M., Omicini, A., & Zambonelli, F. (2000). Rulling Agent Motion in Structured Environments. In M. R. Bubak & H. Afsarmanesh & R. Williams & B. Hertzberger (Eds.), *High Performance Computing and Networking - Proceedings of the 8th International Conference (HPCN Europe 2000)* (Vol. 1823 - LNCS, pp. 187-196). Amsterdam: Springer-Verlag.

CYC (2002). [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.cyc.com/

DAML. (2000). *DAML - DARPA Agent Markup Language* [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.daml.org/

Dan, A., Dias, D. M., Kearney, R., Lau, T. C., Nguyen, T. N., Parr, F. N., Sachs, M. W., & Shaikh, H. H. (2001). Business-to-business integration with tpaML and a business-to-business protocol framework. *IBM Systems Journal, 40*(1), 68-90.

Dan, A., Dias, D. M., Nguyen, T. N., Sachs, M. W., Shaikh, H. H., King, R., & Duri, S. (1998). The Coyote Project: Framework for Multi-party E-Commerce. In C. Nikolaou & C. Stephanidis (Eds.), *Research and Advanced Technology for Digital Libraries* (Vol. LNCS 1513, pp. 873-890). Berlin: Springer-Verlag.

Daoud, F. (1998). A Business Contracting Model for TINA Architecture. *International Journal of Electronic Markets, 8*(3), 23-27.

Daskalopulu, A. (1998). *Legal Contract Drafting at the Micro-Level.* In Proceedings of 5th International Conference of the IDG-CNRI.

Daskalopulu, A., & Maibaum, T. (2001). *Towards Electronic Contract Performance.* In Proceedings of LISA'01 - Legal Information Systems Applications in DEXA'01 - 12th International Conference and Workshop on Database and Expert Systems Applications.

Dautenhahn, K. (2001). Socially Intelligent Agents - The Human in the Loop. *IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans, 31*(5), 345348.

Davenport, T. H. (1993). *Process innovation: reengineering work through information technology*. Boston, Mass.: Harvard Business School Press for Ernst & Young Center for Information Technology and Strategy.

Davidow, W. H., & Malone, M. S. (1993). *The Virtual Corporation: structuring and revitalizing the corporation for the 21st century* (1. pbk ed.). New York: HarperBusiness.

DCOM. (2001). *Microsoft. DCOM Specification. http://www.microsoft.com* [web site]. Retrieved Jan 2002, 2002, from the World Wide Web:

De Masi, D. (2000). *A Sociedade Pós Industrial*. São Paulo: SENAC.

De Nicola, R., Ferrari, G. L., & Pugliese, R. (1998). KLAIM: A Kernel Language for Agents Interaction and Mobility. *IEEE Transactions on Software, 24*(5), 315-330.

Dean, N. (1997). *The Essence of Discrete Mathematics*. London: Prentice Hall.

Decker, S., Erdmann, M., Fensel, D., & Studer, R. (1999). Ontobroker: Ontology based access to distributed and semi-structured information. In R. Meersman & Z. Tari & S. Stevens (Eds.), *DS-8: Semantic Issues in Multimedia Systems* (pp. 351-369). Boston, MA: Kluwer Academic Publisher.

DEDEMAS. (1998). *DEDEMAS - Esprit Project EP 24986* [Web Site]. Retrieved Feb 2003, 2003, from the World Wide Web: http://dedemas.ifw.uni-hannover.de/

Dellarocas, C., & Klein, M. (2001). Contractual Agent Societies. In R. Conte & C. Dellarocas (Eds.), *Social Order in Multiagent Systems* (pp. 113-133). Boston: Kluwer Academic Publishers.

Dertouzos, M. L. (1997). *What Will Be : how the new world of information will change our lives* (1. ed.). San Francisco, Calif.: Harper Edge.

DeviceNet (2002). [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.odva.org

Dictionary. (2002). *Dictionary*. http://www.dictionary.com/. Retrieved, from the World Wide Web: http://www.dictionary.com/

Dignum, F., & Sierra, C. (Eds.). (2001). *Agent Mediated Electronic Commerce: the European AgentLink perspective* (Vol. LNAI 1991). Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo: Springer.

Dilts, D. M., Boyd, N. P., & Whorms, H. H. (1991). The Evolution of Control Architectures for Automated manufacturing Systems. *Journal of Manufacturing Systems, 10*(1), 79-93.

d'Inverno, M., Kinny, D., Luck, M., & Wooldridge, M. (1997). A Formal Specification of dMARS. In M. P. Singh & A. S. Rao & M. Wooldridge (Eds.), *Intelligent Agents IV: Agent Theories, Architectures, and Languages* (Vol. LNAI 1365). Heidelberg - Germany: Springer-Verlag.

Domingue, J. (1998). *Tadzebao and WebOnto: Discussing, Browsing, and Editing Ontologies on the Web.* In Proceedings of KAW'98 - 11th Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff - Canada.

Domingue, J., & Motta, E. (2000). Planet-Onto: From News Publishing to Integrated Knowledge Management Support. *IEEE Intelligent Systems, 15*(3), 26-32.

Donald, B. R. (1989). *Error Detection and Recovery in Robotics*. New York ; Berlin: Springer-Vlg.

Doumeingts, G., Vallespir, B., & Chen, D. (1995). Methodologies for Designing CIM Systems: A Survey. *Computers in Industry, 25*, 263-280.

Drucker, P. F. (1969). *The Age of Discontinuity: guidelines to our changing society*. London: Pan books.

Drucker, P. F. (1988). The Coming of the New Organisation. *Harvard Business Review*(January-February), 44-53.

Drucker, P. F. (1990). The Emerging Theory of Manufacturing. *Harvard Business Review*(May-June), 1-8.

Drucker, P. F. (1992). *Managing for the Future: the 1990s and beyond*. New York: Truman Talley/Dutton.

Duffie, N. A. (1990). Synthesis of Heterarchical Manufacturing Systems. *Computers in Industry, 14*(1-3), 167-174.

Duffie, N. A., Chitturi, R., & Mou, J. (1988). Fault-Tolerant Heterarchical Control of Heterogeneous Manufacturing System Entities. *Journal of Manufacturing Systems, 7*(4), 315-327.

Duffie, N. A., & Piper, R. S. (1986). Non-Hierarchical Control of Manufacturing Systems. *Journal of Manufacturing Systems, 5*(2), 137-139.

Duffie, N. A., & Piper, R. S. (1987). Non-Hierarchical Control of a Flexible Manufacturing Cell. *Robotics and Computer Integrated Manufacturing, 3*(2), 175-179.

Duffie, N. A., & Prabhu, V. V. (1994). Real-Time Distributed Schedulling of Heterarchical Manufacturing Systems. *Journal of Manufacturing Systems, 13*(2), 94-107.

Duffie, N. A., & Prabhu, V. V. (1996). Heterarchical Control of Highly Distributed Manufacturing Systems. *International Journal Computer Integrated Manufacturing, 9*(4), 270-281.

Duguay, C. R., Landry, S., & Pasin, F. (1997). From mass production to flexible/agile production. *International Journal of Operations & Production Management, 17*(12), 1183-1195.

ebXML. (2003). *Electronic Business using XML* [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.ebxml.org

Egashira, S., & Hashimoto, T. (2001). The Formation of Common Norms on the Assumption of 'Fundamentally' Imperfect Information. In R. Conte & C. Dellarocas (Eds.), *Social Order in Multiagent Systems* (pp. 181-198). Boston: Kluwer Academic Publishers.

Ellegard, K., Jonsson, D., Engstrom, T., Johansson, M. I., Medbo, L., & Johansson, B. (1992). Reflective Production in the Final Assembly of Motor Vehicles - An Emerging Swedish Challenge. *International Journal of Operations & Production Management, 12*(7/8), 117-133.

Englemore, R. S., & Morgan, A. J. (Eds.). (1988). *Blackboard Systems*. Reading - MA: Addison-Wesley.

Engstrom, T. (1994). Intra-group Work Patterns in Final Assembly of motor Vehicles. *International Journal of Operations & Production Management, 14*(3), 101-113.

Engstrom, T. (1998). The Volvo Uddevalla Plant and Interpretations of Industrial Design Process. *Integrated Manufacturing Systems, 9*(5), 279-295.

Engstrom, T., Jonsson, D., & Medbo, L. (1996). Production model discourse and experiences from the Swedish automotive industry. *International Journal of Operations & Production Management, 16*(2), 141-158.

Erixon, G. (1996). Design for Modularity. In G. Q. Huang (Ed.), *Design for X: Concurrent engineering imperatives* (1. ed., pp. 489). London: Chapman & Hall.

Erixon, G. (1998). *Modular Function Deployment - A Method for Product Modularisation.* Unpublished PhD Thesis, The Royal Institute of Technology - KTH, Stockholm.

ESPRIT Consortium AMICE. (1993). *CIMOSA: open system architecture for CIM* (2nd., rev. and extended ed.). Berlin: Springer-Verlag.

ESPRIT Consortium CCE-CNMA. (1995). *MMS: a communication language for manufacturing*. Berlin: Springer-Verlag.

Farquhar, A., Fikes, R., & Rice, J. (1997). The Ontolingua Server: A tool for collaborative ontology construction. *International Journal of Human-Computer Studies, 46*(6), 707-727.

Feng, M., Gao, Y., & Yuen, C. (1995). Implementing Linda Tuple Space on a Distributed System. *International Journal of High Speed Computing, 7*(1), 125-144.

Fensel, D., van Harmelen, F., Horrocks, I., McGuinness, D. L., & Patel-Schneider, P. F. (2001). OIL: An Ontology Infrastructure for The Semantic Web. *IEEE Intelligent Systems, 16*(2), 38-45.

Ferber, J. (1999). *Multi-Agent Systems : an Introduction to Distributed Artificial Intelligence*. Harlow: Addison-Wesley.

Ferguson, I. A. (1992). *Touring Machines: an Architecture for Dynamic, Rational, Mobile Agents.* Unpublished PhD Thesis, University of Cambridge, Cambridge.

Ferguson, I. A. (1995). Integrated Control and Coordinated Behaviour: a Case for Agent Models. In M. Wooldridge & N. R. Jennings (Eds.), *Intelligent Agents: Theories, Architectures and Languages* (Vol. LNAI 890, pp. 203-218). Berlin: Springer Verlag.

Fernández, M., Gómez-Pérez, A., & Juristo, N. (1997). METHONTOLOGY: From Ontological Art Towards Ontological Engineering. In A. Farquhar & M. Gruninger (Eds.), *Ontological Engineering - AAAI Spring Symposium* (pp. 33-40). Stanford, CA.

FieldBus (2002). [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.fieldbus.org

Fikes, R., & Kehler, T. (1985). The Role of Frame-Based Representation in Reasoning. *Communications of the ACM, 28*(9), 904-920.

Fikes, R., & Nilsson, N. J. (1971). STRIPS: A New Approach to the Applicaton of Theorem Proving to Problem Solving. *Artificial Intelligence, 2*(3-4), 189-208.

Fikes, R., & Nilsson, N. J. (1990). STRIPS: A New Approach to the Applicaton of Theorem Proving to Problem Solving. In J. Allen & J. A. Hendler & A. Tate (Eds.), *Readings in Planning* (pp. 88-97). San Mateo, Calif.: Morgan Kaufmann Publishers.

Filipe, J. (2000). *Normative Organisational Modelling using Intelligent MultiAgent Systems.* Unpublished PhD, University of Staffordshire - UK.

Filipe, J. (2002a). *Bureaucratic Agents in Business Organisations.* In Proceedings of ACI 2002 - IASTED International Conference on Artificial and Computational Intelligence, Tokyo - Japan.

Filipe, J. (2002b). *A Normative and Intentional Agent Model for Organisation Modelling.* In Proceedings of ESAW 2002 - 3rd Workshop on Engineering Societies in the Agents World, Madrid - Spain.

Filipe, J. (2003). *Information Fields in Organization Modeling using an EDA Multi-Agent Architecture.* In Proceedings of AMKM-2003 - AAAI Spring Symposium on Agent-Mediated Knowledge Management, Stanford - USA.

Filipe, J., & Liu, K. (2001). An Organisational Semeiotics Perspective for Norm-Based Agent Design. In R. Conte & C. Dellarocas (Eds.), *Social Order in Multiagent Systems* (pp. 135-158). Boston: Kluwer Academic Publishers.

Finin, T., Labrou, Y., & Mayfield, J. (1997). KQML as an Agent Communication Language. In J. M. Bradshaw (Ed.), *Software Agents* (pp. 291-316). Cambridge, Massachussets: MIT Press.

Finin, T., Weber, J., Wiederhold, G., Genesereth, M. R., Fritzson, R., McKay, D., McGuire, J., Pelavin, R., Shapiro, S. C., & Beck, C. (1993). *Specification of the KQML Agent-Communication Language -- plus example agent policies and architectures*: The DARPA Knowledge Sharing Initiative External Interfaces Working Group.

FIPA. (2000). *FIPA SL Content Language Specification* (XC0008D). Geneve: FIPA - Foundation For Intelligent Physical Agents.

FIPA. (2001a). *FIPA CCL Content Language Specification* (XC00009B). Geneve: FIPA - Foundation For Intelligent Physical Agents.

FIPA. (2001b). *FIPA Ontology Service Specificationn* (XC00086D). Geneve: FIPA - Foundation For Intelligent Physical Agents.

FIPA. (2001c). *FIPA RDF Content Language Specification* (XC00011B). Geneve: FIPA - Foundation For Intelligent Physical Agents.

FIPA. (2001d). *FIPA Request Interaction Protocol Specification* (XC00026F). Geneve: FIPA - Foundation for Intelligent Physical Agents.

FIPA. (2002a). *FIPA ACL Message Structure Specification* (XC00061F). Geneve: FIPA - Foundation For Intelligent Physical Agents.

FIPA. (2002b). *FIPA Communicative Act Library Specification* (XC00037J). Geneve: FIPA - Foundation For Intelligent Physical Agents.

FIPA. (2002c). *FIPA Query Interaction Protocol Specification* (SC00027H). Geneve: FIPA - Foundation for Intelligent Physical Agents.

FIPA. (2002d). *FIPA Request Interaction Protocol Specification* (SC00026H). Geneve: FIPA - Foundation for Intelligent Physical Agents.

FIPA. (2002e). *The Foundation for Intelligent Physical Agents* [web site]. Retrieved October 2002, 2002, from the World Wide Web: http://www.fipa.org

FIPA. (2003). *FIPA KIF Content Language Specification* (XC00010C). Geneve: FIPA - Foundation For Intelligent Physical Agents.

FIPA-OS. (2001). *http://www.emorphia.com/research/about.htm* [web site]. Retrieved Jan 2002, 2002, from the World Wide Web:

Fischer, L. (Ed.). (2003). *The Workflow Handbook 2003 - Published in association with the Workflow Management Coalition (WfMC)*. Lighthouse Point, FL, USA: Future Strategies Inc.

Fletcher, M., Brennan, R. W., & Norrie, D. H. (2003). Modeling and Reconfiguring Intelligent Holonic Manufacturing Systems with Internet-based Mobile Agents. *Journal of Intelligent Manufacturing, 14*(1), 7-23.

Ford, H., & Crowther, S. (1988). *Today and Tomorrow*. Cambridge, Mass.: Productivity Press.

Fox, M. S. (1992). The TOVE Project: A Common-sense Model of the Enterprise. In F. Belli & F. J. Radermacher (Eds.), *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems* (Vol. LNAI 604, pp. 25-34). Berlin: Springer-Verlag.

Fox, M. S., & Gruninger, M. (1998). Enterprise Modeling. *Ai Magazine, 19*(3), 109-121.

Franklin, S., & Graesser, A. (1997). Is it an Agent or Just a Program? A Taxonomy for Autonomous Agents. In J. P. Muller & M. Wooldridge & N. R. Jennings (Eds.), *Intelligent Agents III - Agent Theories, Architectures, and Languages* (Vol. 1193, pp. 21-35). Berlin: Springer-Verlag.

Freeman, E., Hupfer, S., & Arnold, K. (1999). *JavaSpaces: Principles, Patterns, and Practices*: Addison-Wesley.

Freyssenet, M. (Ed.). (1998). *One best way? trajectories and industrial models of the world's automobile producers*. Oxford: Oxford Univ. Press.

Gabbay, D. M., Finger, M., & Reynolds, M. (2000). *Temporal logic*.

Garita, C., Afsarmanesh, H., & Hertzberger, L. O. (2001). The PRODNET Federated Information Management Approach for Virtual Enterprise Support. *Journal of Intelligent Manufacturing, 12*, 151-170.

Garita, C., Afsarmanesh, H., & Hertzberger, L. O. (2002). A Survey of Distributed Information Management Approaches for Virtual Enterprise Infrastructures. In U. J. Franke (Ed.), *Managing Virtual Web Organizations in the 21st Century: Issues and Challenges*: Idea Group Publishing.

Gates, B., Myhrvold, N., & Rinearson, P. (1996). *The Road Ahead* (Completely rev. and updated ed.). London: Penguin.

Gelernter, D., & Carriero, N. (1992). Coordination Languages and their Significance. *Communications of the ACM, 35*(2), 97-107.

Genesereth, M. R., & Fikes, R. E. (1992). *Knowledge Interchange Format, Version 3.0 Reference Manual* (logic-92-1): Computer Science Department, Stanford University.

Genesereth, M. R., & Ketchpel, S. P. (1994). Software Agents. *Communications of the ACM, 37*(7), 48-53.

Genesereth, M. R., & Nilsson, N. J. (1987). *Logical Foundations of Artificial Intelligence*. Los Altos, Calif.: Morgan Kaufmann.

Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubézy, M., Eriksson, H., Noy, N. F., & Tu, S. W. (2003). The Evolution of Protégé: an environment for knowledge-based systems development. *International Journal of Human-Computer Studies, 58*(1), 89-132.

Georgeff, M. P., & Ingrand, F. F. (1989). *Decision Making in an Embedded Reasoning System.* In Proceedings of IJCAI-89 - Eleventh International Conference on Artificial Intelligence, Detroit - MI.

Georgeff, M. P., & Lansky, A. L. (1987). *Reactive Reasoning and Planning.* In Proceedings of AAAI-87 - sixth National Conference on Artificial Intelligence, Seattle - WA.

Georgeff, M. P., Pell, B., Pollack, M., Tambe, M., & Wooldridge, M. (1999). The belief-Desire-Intention Model of Agency. In J. P. Muller & M. P. Singh & A. S. Rao (Eds.), *Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages (ATAL-98)* (Vol. 1555, pp. 1-10). Heidelberg - Germany: Springer-Verlag.

Giampapa, J. A., Paolucci, M., & Sycara, K. (2000). *Agent Interoperation Across Multagent System Boundaries.* In Proceedings of Fourth International Conference on Autonomous Agents (Agents 2000), Barcelona - Spain.

Ginsberg, M. L. (1987). *Readings in Nonmonotonic Reasoning*. Los Altos, CA: M. Kaufmann Publishers.

Ginsberg, M. L. (1991). Knowledge Interchange Format: the KIF of Death. *AI Magazine, 12*(3), 81-94.

Girle, R. (2000). *Modal Logics and Philosophy*. Teddington: Acumen.

Gisler, M., Stanoevska-Slabeva, K., & Greunz, M. (2000, 5-6 June). *Legal Aspects of Electronic Contracts.* In Proceedings of IDSO'00 - Workshop on Infrastructure for Dynamic Business-to-Business Service Outsourcing, Stockholm, Sweden.

Goldman, S. L., Nagel, R. N., & Preiss, K. (1995). *Agile competitors and virtual organizations: strategies for enriching the customer*. New York: Van Nostrand Reinhold.

Gómez-Pérez, A. (2001). Evaluation of Ontologies. *International Journal of Intelligent Systems, 16*(3), 391-409.

Gómez-Pérez, A., & Corcho, O. (2002). Ontology Languages for the Semantic Web. *IEEE Intelligent Systems, 17*(1), 54-59.

Goodchild, A., Herring, C., & Milosevic, Z. (2000, 5-6 June). *Business Contracts for B2B.* In Proceedings of IDSO'00 - Workshop on Infrastructure for Dynamic Business-to-Business Service Outsourcing, Stockholm, Sweden.

Goranson, H. T. (1999). *The Agile Virtual Enterprise: cases, metrics, tools*. Westport, Conn.: Quorum Books.

Gou, L., Luh, P. B., & Kyoya, Y. (1998). Holonic Manufacturing Scheduling: Architecture, Cooperation Mechanism, and Implementation. *Computers in Industry, 37*(3), 213-231.

Grady, J. O. (1994). *System Integration*. Boca Raton, Fla. ; London: CRC.

Graefe, U., & Thomson, V. (1989). A Reference Model for Production Control. *International Journal Computer Integrated Manufacturing, 2*(2), 86-93.

Greunz, M., Schopp, B., & Stanoevska-Slabeva, K. (2000, 10-13 Aug). *Supporting Market Transactions through XML Contracting Container.* In Proceedings of AMCISS'00 - Sixth Americas Conference on Information Systems, Long Beach, CA.

Griffel, F., Boger, M., Weinreich, H., Lamersdorf, W., & Merz, M. (1998). *Electronic Contracting with COSMOS - How to Establish, Negotiate and Execute Electronic Contracts on the Internet.* In Proceedings of EDOC'98 - 2nd Int. Enterprise Distributed Object Computing Workshop.

Grondahl, P., & Onori, M. (2000). Standardised Flexible Automatic Assembly - evaluating the Mark IV approach. *Assembly Automation, 20*(3), 217-224.

Grosof, B., Labrou, Y., & Chan, H. Y. (1999). *A Declarative Approach to Business Rules in Contracts: Courteous Logic Programs in XML.* In Proceedings of EC'99 - 1st ACM Conference on Electronic Commerce, Denver, Colorado, USA.

Gross, D. (1997). *Forbes' Greatest Business Stories of All Time*. New York ; Chichester: Wiley.

Grosso, W. (2002). *Java RMI*. Beijing ; Sebastopol, CA: O'Reilly.

Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition, 5*(2), 199-220.

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies, 43*(5-6), 907-928.

Gruninger, M., & Lee, J. (2002). Ontology: Applications and Design. *Communications of the ACM, 45*(2), 39-41.

Guarino, N. (1995). Formal ontology, conceptual analysis and knowledge representation. *International Journal of Human-Computer Studies, 43*(5-6), 625-640.

Guarino, N. (1997). Understanding, building and using ontologies. *International Journal of Human-Computer Studies, 46*(2-3), 293-310.

Guarino, N., Masolo, C., & Vetere, G. (1999). OntoSeek: Content-Based Access to the Web. *IEEE Intelligent Systems, 14*(3), 70-80.

Guarino, N., & Welty, C. (2002). Evaluating Ontological Decisions with ONTOCLEAN. *Communications of the ACM, 45*(2), 61-65.

Guha, R. V., & Lenat, D. B. (1994). Enabling Agents to Work Together. *Communications of the ACM, 37*(7), 127-142.

Gullander, P. (1999). *On Reference Architectures for Development of Flexible Cell Control Systems.* Unpublished PhD Thesis, Gotenborg University, Sweden.

Gupta, D., & Buzacott, J. A. (1991). A Framework for Understanding Flexibility of Manufacturing Systems. *Journal of Manufacturing Systems, 8*(2).

Hall, R. (1995). *The Soul of the Enterprise: Creating a Dynamic Vision for American Manufacturing*: John Wiley & Sons.

Halter, S. L. (2002). *JavaSpaces Example by Example*. Upper Saddle River, N.J. ;: Prentice Hall PTR.

Hames, R. D. (1999). *The Management Myth: Exploring the Essence of Future Organizations*: Business & Professional Publishing.

Hammer, M., & Champy, J. (1995). *Reengineering the Corporation: a manifesto for business revolution* (Rev. ed.). London: Nicholas Brealey Publishing.

Hammer, M., & Stanton, S. A. (1995). *The Reengineering Revolution Handbook*. London: HarperCollins.

Handy, C. B. (1991). *The Age of Unreason* (2. ed.). London: Arrow Business Books.

Handy, C. B. (1994). *The Age of Paradox*. Boston, Mass.: Harvard Business School Press.

Hansen, W. C. (1991). *The Integrated Enterprise.* In Proceedings of Foundations of World-Class Manufacturing Systems.

Harrington, J. (1973). *Computer Integrated Manufacturing*. New York: Industrial Press.

Hatvany, J. (1985). Intelligence and Cooperation in Heterarchic Manufacturing systems. *International Journal on Robotics and Computer Integrated Manufacturing, 2*(2), 101-104.

Hatvany, J. (1990). Dreams, Nightmares and Reality. *Computers in Industry, 14*(1-3), 11-16.

Hayes, P. J. (1980). The Logic of Frames. In D. Metzing (Ed.), *Frame Conceptions and Text Understanding* (pp. 46-61). Berlin: deGruyter.

Heilala, J., & Voho, P. (2001). Modular Reconfigurable Flexible Final Assembly Systems. *Assembly Automation, 21*(1), 20-28.

Hendler, J. A. (2001). Agents and The Semantic Web. *IEEE Intelligent Systems, 16*(2), 30-37.

Hendler, J. A., & Stoffel, K. (1999). Back-End Technology for High-Performance Knowledge-Representation Systems. *IEEE Intelligent Systems, 14*(3), 63-69.

Heragu, S. S., Graves, R. J., Kim, B.-I., & Onge, A. S. (2002). Intelligent Agent Based Framework for Manufacturing Systems Control. *IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans, 32*(5), 560-573.

Higgins, P. G. (1995). Review of Anthropocentric Production Systems - the European Response to Advanced Manufacturing and Globalization by Franz Lehner. *International Journal of Human Factors in Manufacturing, 5*(1).

Hirsch, B., & Actis-Dato, M. (1987). *ESPRIT CIM : design, engineering, management, and control of production systems*. Amsterdam ; New York: North-Holland.

HMS. (2000). *Holonic Manufacturing Systems* [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://hms.ifw.uni-hannover.de/

Hogg, L. M. J., & Jennings, N. R. (2001). Socially Intelligent Reasoning for Autonomous Agents. *IEEE Transactions on Systems, Man and Cybernetics - Part A, 31*(5), 381-393.

Holsapple, C. W., & Joshi, K. D. (2002). A Collaborative Approach to Ontology Design. *Communications of the ACM, 45*(2), 42-47.

Hopp, W. J., & Spearman, M. L. (2001). *Factory Physics: foundations of manufacturing management* (2. ed.). New York: Irwin/McGraw-Hill.

Hormozi, A. M. (2001). Agile Manufacturing: the next logical step. *Benchmarking, An International Journal, 8*(2), 132-143.

Huff, B. L., & Edwards, C. R. (1999). Layered Supervisory Control Architecture for Reconfigurable Automation. *Production Planning & Control, 10*(7), 659-670.

Huhns, M. N., & Stephens, L. M. (1999). Multiagent Systems and Societies of Agents. In G. Weiss (Ed.), *Multiagent Systems: a Modern Approach to Distributed Artificial Intelligence* (pp. 79-120). Cambridge, Masschusetts: MIT Press.

IBM-Aglets. (2001). *http://www.trl.ibm.co.jp/aglets/aglets* [web site]. Retrieved Jan 2002, 2002, from the World Wide Web:

IFIP-IFAC. (1999). *GERAM: Generalised Enterprise Reference Architecture and Methodology*: IFIP-IFAC Task Force.

Imai, M. (1988). *Kaizen: the key to Japan's competitive success*. New York: McGraw-Hill Education.

IMS. (2000). *Intelligent Manufacturing Systems* [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.ims.org

IMTI. (1999). *Intelligent Controls for Discrete Manufacturing*. Oak Ridge - Tennessee: IMTI - Intelligent Manufacturing Technology Initiative (USA) - http://www.imti21.org.

IMTI. (2000a). *Information Systems for the Manufacturing Enterprise*. Oak Ridge - Tennessee: IMTI - Intelligent Manufacturing Technology Initiative (USA) - http://www.imti21.org.

IMTI. (2000b). *Manufacturing Processes & Equipment*. Oak Ridge - Tennessee: IMTI - Intelligent Manufacturing Technology Initiative (USA) - http://www.imti21.org.

IMTI. (2000c). *Manufacturing Success in the 21st Century: A Strategic View*. Oak Ridge - Tennessee: IMTI - Intelligent Manufacturing Technology Initiative (USA) - http://www.imti21.org.

JACK (2000). [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.agent-software.com/shared/products/index.html

JADE. (2001). *http://sharon.cselt.it/projects/jade/* [web site]. Retrieved Jan 2002, 2002, from the World Wide Web:

Jain, A. K., Aparicio, M., & Singh, M. P. (1999). Agents for Process Coherence in Virtual Enterprises. *Communications of the ACM, 42*(3), 62-69.

James-Moore, S. M., & Gibbons, A. (1997). Is Lean Manufacture Universally Relevant? An Investigative Methodology. *International Journal of Operations & Production Management, 17*(9), 899-911.

Jennings, N. R. (2001). An Agent-Based Approach for Building Complex Software Systems. *Communications of the ACM, 44*(4), 35-41.

Jennings, N. R., Faratin, P., Lomuscio, A. R., Parsons, S., Sierra, C., & Wooldridge, M. (2001). Automated Negotiations: Prospects, Methods and Challenges. *International Journal of Group Decision and Negotiation, 10*(2), 199-215.

Jeon, H., Petrie, C., & Cutkosky, M. R. (2000). JATLite: A Java Agent Infrastructure with Message Routing. *IEEE Internet Computing, 4*(2), 87-96.

Johnson, S. (2001). *Emergence*. London: Penguin group.

Jones, A. J. I., & Sergot, M. J. (1992). Deontic Logic in the Representation of Law: Towards a Methodology. *Artificial Intelligence and Law, 1*(1), 45-64.

Jones, A. J. I., & Sergot, M. J. (1993). On the Characterisation of Law and Computer Systems: The Normative Systems Perspective. In J.-J. C. Meyer & R. J. Wieringa (Eds.), *Deontic Logic in Computer Science: Normative System Specification*. Chichester: John Wiley and Sons.

Jones, A. T., & McLean, C. R. (1986). A Proposed Hierarchical Control Model for Automated Manufacturing Systems. *Journal of Manufacturing Systems, 5*(1), 15-25.

Jorysz, H. R., & Vernadat, F. B. (1990a). CIMOSA Part 1: Total Enterprise Modelling and Function View. *International Journal of Computer Integrated Manufacturing, 3*(3-4), 144-156.

Jorysz, H. R., & Vernadat, F. B. (1990b). CIMOSA Part 2: Information View. *International Journal of Computer Integrated Manufacturing, 3*(3-4), 157-167.

Jorysz, H. R., & Vernadat, F. B. (1990c). CIMOSA Part 3: Integrating Infrastructure - The operational basis for integrated manufacturing systems. *International Journal of Computer Integrated Manufacturing, 3*(3-4), 168-180.

Kaelbling, L. P., & Rosenschein, S. J. (1990). Action and Planning in Embedded Agents. In P. Maes (Ed.), *Designing Autonomous Agents* (pp. 35-48). Cambridge, Mass.: MIT Press.

Kaighobadi, M., & Venkatesh, K. (1994). Flexible Manufacturing Systems: An Overview. *International Journal of Operations & Production Management, 14*(4), 26-49.

Kalenka, S., & Jennings, N. R. (1999). Socially Responsible Decision making by Autonomous Agents. In K. Korta & E. Sosa & X. Arrazola (Eds.), *Cognition, Agency and Rationality* (pp. 135-149). Dordrecht: Kluwer Academic Publishers.

Kambil, A., & van Heck, E. (1998). Re-engineering the Dutch Flower Auctions: A Framework for Analyzing Exchange Organizations. *Information Systems Research, 9*(1).

Karagiannis, D. (Ed.). (1991). *Information Systems and Artificial Intelligence: integration aspects*. Berlin ; New York: Springer-Verlag.

Karlsson, C. (1996). Radically New Production Systems. *International Journal of Operations & Production Management, 16*(11), 8-19.

Karp, P., Myers, K., & Gruber, T. (1995). *The Generic Frame Protocol.* In Proceedings of 1995 International joint Conference on Artificial Intelligence.

Katayama, H., & Bennet, D. (1996). Lean Production in a Changing Competitive World: a Japanese perspective. *International Journal of Operations & Production Management, 16*(2), 8-23.

Kateel, G., Kamath, M., & Pratt, D. (1996). An Overview of CIM Enterprise Modeling Methodologies. In J. M. Charnes & D. J. Morrice & D. T. Brunner & J. J. Swain (Eds.), *Winter Simulation Conference* (pp. 1000-1007).

Katsh, M. E. (1995). *Law in a Digital World*. New York ; Oxford: Oxford University Press.

Kersten, G., Noronha, S. J., & Teich, J. (2000). *Are All E-Commerce Negotiations Auctions?* In Proceedings of COOP'2000 - Fourth International Conference on the Design of Cooperative Systems, Sophia-Antipolis - France.

KIF. (2002). *KIF - Knowledge Interchange Format* [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www-ksl.stanford.edu/knowledge-sharing/kif/

Kinnie, N., Hutchinson, S., & Purcell, J. (1997). Downsizing: is it always lean and mean? *Personnel Review, 27*(4), 296-311.

Klein, S. (1997). Introduction to Electronic Auctions. *International Journal of Electronic Commerce, 7*(4), 3-6.

Klusch, M., & Gerber, A. (2002). Dynamic Coalition Formation Among Rational Agents. *IEEE Intelligent Systems, 17*(3), 42-47.

Klusch, M., & Sycara, K. (2001). Brokering and Matchmaking for Coordination of Agent Societies: A Survey. In A. Omicini & F. Zambonelli & M. Klusch & R. Tolksdorf (Eds.), *Coordination of Internet Agents: Models, Technologies, and Applications* (pp. xxvii, 523). Berlin: Springer-Verlag.

Knoblock, C. A. (1992). An Analysis of ABSTRIPS. In J. A. Hendler (Ed.), *Artificial Intelligence Planning Systems: Proceedings of the First International Conference (AIPS 92)* (pp. 126-135). Maryland - USA: Morgan Kaufmann.

Kochan, A. (1998). Automotive Industry Looks for Lean Production. *Assembly Automation, 18*(2), 132-137.

Koestler, A. (1989). *The Ghost in the Machine*. London: Arkana.

Koetsier, M., Grefen, P., & Vonk, J. (2000). *Contracts for Cross-Organizational Workflow Management.* In Proceedings of 1st International Conference on Electronic Commerce and Web Techn, London, UK.

Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritchow, G., Ulsoy, A. G., & Van Brussel, H. (1999). Reconfigurable Manufacturing Systems. *CIRP Annals, 48*(2).

Kornhauser, W. (1960). *The Politics of Mass Society*. London: Routledge & Kegan Paul.

Kotok, A., & Webber, D. (2001). *ebXML: the new global standard for doing business over the internet*. Indianapolis, Ind.

Hemel Hempstead: New Riders ;

Prentice Hall.

Kovács, I., & Castillo, J. J. (1998). *Novos Modelos de Produção*. Oeiras: Celta Editora.

Kowalski, R. (1995). Legislation as Logic Programs. In Z. Bankowski & I. White & U. Hahn (Eds.), *Informatics and the Foundations of Legal Reasoning* (pp. 325-356). Dordrecht: Kluwer Academic Publishers.

KQML (2000). [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.cs.umbc.edu/kqml/

Kraus, S. (1997). Negotiation and Cooperation in Multi-Agents Environments. *Artificial Intelligence (Special Issue on Economic Principles of Multi-Agent Systems), 94*(1-2), 79-98.

Kraus, S. (2002). Automated Negotiation and Decision Making in Multiagent Environments. In M. Luck & V. Marík & O. Stepankova & R. Trappl (Eds.), *Multi-Agent Systems and Applications: 9th ECCAI Advanced Course ACAI 2001 and Agent Link's 3rd European Agent Systems Summer School, EASSS 2001, Prague, Czech Republic, July 2-13, 2001, Selected Tutorial Papers* (Vol. LNAI 2086, pp. xii, 375). Berlin: Springer-Verlag.

Kraus, S., Sycara, K., & Evenchik, A. (1998). Argumentation in Negotiation: A Formal Model and Implementation. *Artificial Intelligence, 104*(1-2), 1-69.

Kuhn, T. S. (1970). *The Structure of Scientific Revolutions* (2. ed.). Chicago: University of Chicago Press.

Kumar, S., & Meade, D. (2002). Has MRP run its course? A review of contemporary developments in planning systems. *Industrial Management & Data Systems, 102*(8), 453-462.

Labrou, Y., Finin, T., & Peng, Y. (1999). Agent Communication Languages: The Current Landscape. *IEEE Intelligent Systems, 14*(2), 45-52.

Lassila, O., & McGuinness, D. (2001). The Role of Frame-Based Representation on the Semantics Web. *Linköping Electronic Articles in Computer and Information Science, 6*.

Lathrop, R. (2001). Intelligent Systems in Biology: Why the Excitement? *IEEE Intelligent Systems, 16*(6), 8-13.

Lawrenz, W. (1997). *CAN System Engineering: From Theory to Practical Applications*. New York: Springer-Verlag New York Inc.

Lebow, I. (1995). *Information Highways and Byways: from the telegraph to the 21st century*. New York: IEEE Press.

Lee, R. M. (1998). Towards Open Electronic Contracting. *International Journal of Electronic Markets, 8*(3), 3-8.

Leger, J. B., & Morel, G. (2001). Integration of maintenance in the enterprise: towards an enterprise modelling-based framework compliant with proactive maintenance strategy. *Production Planning & Control, 12*(2), 176-187.

Leitão, P., Barata, J., Camarinha-Matos, L. M., & Boissier, R. (2001). Trends in Agile and Co-operative Manufacturing. In R. Bernhardt & H. H. Erbe (Eds.), *Cost Oriented Automation* (Vol. 1, pp. 149-158). Oxford - UK: Pergamon - Elsevier.

Leitão, P., & Quintas, A. (1997). *A Manufacturing Cell Controller Architecture.* In Proceedings of Flexible Automation and Intelligent Manufacturing Conference, Middlesbrough.

Leitão, P., & Restivo, F. (2002). *Holonic Adaptive Production Control System.* In Proceedings of 28th Annual Conference of the IEEE Industrial Electronics Society.

Lenat, D. B. (1995). CYC: A Large Scale Investment in Knowledge Infrastructure. *Communications of the ACM, 38*(11), 33-38.

Lenat, D. B. (1998). From 2001 to 2001: Common Sense and the Mind of HAL. In D. G. Stork (Ed.), *Hal's Legacy: 2001's Computer as Dream and Reality* (pp. 416). Cambridge, Mass.; London: MIT.

Lenat, D. B., & Guha, R. V. (1989). *Building Large Knowledge-based Systems: representation and inference in the Cyc project*. Reading, Mass.: Addison-Wesley.

Lennartson, B., Fabian, M., & Gullander, P. (2002, 7-11 Ocober). *CHAMP - A Generic Architecture for Flexible Production.* In Proceedings of ISR'2002 - 33rd International Symposium on Robotics, Stockholm, Sweden.

Lenz, K., Oberweis, A., & Schneider, S. (2001). Trust Based Contracting in Virtual Organisations - A concept based on contract workflow management systems. In B. Schmid & K. Stanoevska-Slabeva & V. Tschammer (Eds.), *Towards the E-Society* (pp. 3-16). Boston: Kluwer Academic.

Lewis, D. (1969). *Convention - A Phylosophical Study*. Cambridge - MA: Harvard University Press.

Lewis, M. A. (2000). Lean production and sustainable competitive advantage. *International Journal of Operations & Production Management, 20*(8), 959-978.

Lind, J. (2001). Issues in Agent-Oriented Software Engineering. In P. Ciancarini & M. Wooldridge (Eds.), *Agent-Oriented Software Engineering - AOSE* (Vol. 1957, pp. 45-58): Springer Verlag.

Lindemann, M. (1997). Permanent IT-Support in Electronic Commerce Transactions. *International Journal of Electronic Markets, 7*(1), 18-20.

Lindemann, M., & Runge, A. (1997, April). *Non-Repudiation within the Electronic Contracting Phase of Electronic Commerce Transactions.* In Proceedings of OBEC'97 - First Overcoming Barriers to Electronic Commerce Conference, Malaga, Spain.

Lomuscio, A. R., Wooldridge, M., & Jennings, N. R. (2001). A Classification Scheme for Negotiation in Electronic Commerce. In F. Dignum & C. Sierra (Eds.), *Agent-Mediated Electronic Commerce: A European AgentLink Perspective* (Vol. LNAI -1957, pp. 19-33). Berlin: Springer-Verlag.

Loom (2002). [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.isi.edu/isd/LOOM/

Lyotard, J.-F. (1984). *The Postmodern Condition : a report on knowledge*. Manchester: Manchester Univ. Press.

Machado, R., & Bordini, R. H. (2002). Running AgentSpeak(L) Agents on SIM_AGENT. In J.-J. C. Meyer & M. Tambe (Eds.), *Intelligent Agents VIII: Agent Theories, Architectures and Languages* (Vol. LNAI 2333, pp. 158-174). Berlin: Springer-Verlag.

Maglica, R. (1996a). *On Programming and Control of Robots in Flexible Manufacturing Cells.* Unpublished PhD Thesis, Chalmers University of Technology, Chalmers - Sweden.

Maglica, R. (1996b). *Using a Functional Architecture to Support Structured Robot Integration.* In Proceedings of 28th CIRP International Seminar on Manufacturing Systems, Johannesburg.

Maleki, R. A. (1991). *Flexible Manufacturing Systems: the technology and management*. Englewood Cliffs, N.J.: Prentice Hall.

Manders, A. J. C., & Brenner, Y. S. (1999). Globalization, New Production Concepts and Income Distribution. *International Journal of Social Economics, 26*(4), 559-569.

Martin, D. L., Cheyer, A., & Moran, D. B. (1999). The Open Agent Architecture: A Framework for Building Distributed Software Systems. *Applied Artificial Intelligence - An International Journal, 13*(1-2), 91-128.

Martin, P., & Eklund, P. W. (2000). Knowledge Retrieval and the World Wide Web. *Ieee Intelligent Systems & Their Applications, 15*(3), 18-25.
Marx, K., & Engels, F. (1990). *Capital*. Chicago: Encyclopaedia Britannica.
Maskell, B. (2001). The Age of Agile Manufacturing. *Supply Chain Management: An International Journal, 6*(1), 5-11.
Maslow, A. H. (1943). A Theory of Human Motivation. *Psychological Review, 50*, 370-396.
Maturana, F., Shen, W., & Norrie, D. H. (1999a). MetaMorph: An Adaptive Agent-Based Architecture for Intelligent Manuacturing. *International Journal of Production Research, 1*(3), 159-168.
Maturana, F., Shen, W., & Norrie, D. H. (1999b). MetaMorph: An Adaptive Agent-Based Architecture for Intelligent Manufacturing. *International Journal of Production Research, 37*(10), 2159-2174.
Mayfield, J., Labrou, Y., & Finin, T. (1996). Evaluation of KQML as an Agent Communication Language. In M. Wooldridge & J. P. Muller & M. Tambe (Eds.), *Intelligent Agents Volume II -- Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages* (Vol. LNAI 1037, pp. 347-360). Berlin: Springer-Verlag.
McAfee, R. P., & McMillan, J. (1987). Auctions and Bidding. *Journal of Economic Literature, 25*(June 1987), 699-738.
McIlraith, S., Son, T. C., & Zeng, H. (2001). Semantic Web Services. *IEEE Intelligent Systems, 16*(2), 46-53.
McKendrick, E. (2000). *Contract Law* (Fourth ed.). New York: PALGRAVE.
McNamara, P., & Prakken, H. (1999). *Norms, Logics and Information systems: new studies in deontic logic and computer science*. Amsterdam
Tokyo: IOS ;
Ohmsha.
Mehrabi, M. G., Ulsoy, A. G., & Koren, Y. (2000). Reconfigurable Manufacturing Systems and their Enabling Technologies. *International Journal Manufacturing Technology and Management, 1*(1), 113-130.
Mehrabi, M. G., Ulsoy, A. G., & Koren, Y. (2000). Reconfigurable Manufacturing Systems: Key to Future Manufacturing. *Journal of Intelligent Manufacturing, 11*, 403-419.
MESA. (2002). *Manufacturing Enterprise Solutions Association* [Web Site]. Retrieved Nov 2002, from the World Wide Web: http://www.mesa.org/
Milgrom, P. (1989). Auctions and Bidding: A Primer. *Journal of Economic Perspectives, 3*(3), 3-22.
Miller, G. A. (1995). WORDNET: A Lexical Database for English. *Communications of the ACM, 38*(11), 39-41.
Miller, R. K., & Walker, T. C. (1990). *FMS/CIM Systems Integration Handbook*. Lilburn, Ga. - Englewood Cliffs, N.J.: Fairmont Press;Prentice-Hall.
Milosevic, Z., & Bond, A. (1995, June). *Electronic Commerce on the Internet: What is Still Missing?* In Proceedings of INET'95 - 5th Conference of the Internet Society, Honolulu, Hawai.
Minsky, M. L. (1975). A Framework for Representing Knowledge. In P. H. Winston (Ed.), *The Psychology of Computer Vision* (pp. 211-277). New York: MacGraw Hill.
Minsky, M. L. (1986). *The Society of Mind*. New York, N.Y.: Simon and Schuster.
Mitchell, F. H. (1991). *CIM Systems: an introduction to computer-integrated manufacturing*. Englewood Cliffs, N.J.: Prentice Hall.
Miyagi, P., Camarinha-Matos, L. M., Santos-Filho, Barata, J., & Arakaki, J. (1995). *The Application of Enhanced Mark Flow Graph in Real Time Control Systems.* In Proceedings of LCA'95 - IFAC 4th Symposium on Low Cost Automation, Buenos Aires - Argentina.
Miyagi, P., Camarinha-Matos, L. M., Santos-Filho, Barata, J., & Arakaki, J. (1996). The Application of Enhanced Mark Flow Graph in Real Time Control Systems. In J. Paiuk & J. P. Weisz (Eds.), *Low Cost Automation 1995* (Vol. 1). Oxford - UK: Pergamon - Elsevier.
Morgado, L., & Gaspar, G. (2000). *A Social Reasoning Mechanism Based on a New Approach for Coalition Formation* (di-fcul-tr-00-1). Lisboa: Department of Informatics, University of Lisbon.
Morrison, I., & Schmid, G. (1997). *The Second Curve*: Ballantine Books (Trd Pap).
Muffatto, M. (1999). Evolution of Production Paradigms: The Toyota and Volvo cases. *Integrated Manufacturing Systems, 10*(1), 15-25.
Muller, J. P., & Pischel, M. (1993). *The Agent Architecture InteRRap: Concept and Application* (RR-93-26): DFKI Saarbrucken.
Musen, M. A. (1998). Domain Ontologies in Software Engineering: Use of Protege with the EON Architecture. *Methods of Information in Medicine, 37*, 540-550.
Nagel, R., & Dove, R. (1992). *21st Century Manufacturing Enterprise Strategy*: Iacocca Institute, Lehigh University, USA.
Naisbitt, J. (1982). *Megatrends: ten new directions transforming our lives*. New York: Warner Books.
Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T., & Swartout, W. R. (1991). Enabling Technology for Knowledge Sharing. *AI Magazine, 12*(3), 36-56.
Negroponte, N. (1996). *Being digital*. New York: Vintage.
Niepce, W., & Molleman, E. (1996). Characteristics of Work organisation in Lean Production and Sociotechnical Systems. *International Journal of Operations & Production Management, 16*(2), 77-90.

Nodine, M., Fowler, J., Ksiezyk, T., Perry, B., Taylor, M., & Unruh, A. (2000). Active Information Gathering in InfoSleuth. *International Journal of Cooperative Information Systems, 9*(1-2), 3-28.

Nowacki, H. (1998). The Time is Now - Opening remarks, *Proceedings of the European Conference on Product Data Technology*. Watford: QMS.

Noy, N. F., Fergerson, R. W., & Musen, M. A. (2000). The Knowledge Model of Protege-2000: Combining interoperability and flexibility. In R. Dieng & O. Corby (Eds.), *Knowledge Engineering and Knowledge Management: Methods, Models, and Tools - EKAW 2000* (pp. 17-32). Berlin: Springer-Verlag.

Noy, N. F., & Hafner, C. D. (1997). The State of the Art in Ontology Design - A survey and comparative review. *Ai Magazine, 18*(3), 53-74.

Noy, N. F., & McGuinness, D. L. (2001). *Ontology Development 101: A Guide to Creating Your First Ontology* (KSL Technical Report KSL-01-05): Knowledge Systems Laboratory.

Noy, N. F., Sintek, M., Decker, S., Crubézy, M., Fergerson, R. W., & Musen, M. A. (2001). Creating Semantic Web Contents with Protégé-2000. *IEEE Intelligent Systems, 16*(2), 60-71.

NRC. (1998). *Visionary Manufacturing Challenges for 2020*. Washington: National Academy Press.

Nwana, H. S., Ndumu, D. T., Lee, L. C., & Collis, J. C. (1999). ZEUS: A Toolkit for Building Distributed Multiagent Systems. *Applied Artificial Intelligence - An International Journal, 13*(1-2), 129-185.

OASIS (2003). [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.oasis-open.org

Ohno, T. (1988). *Toyota Production System: beyond large-scale production*. Cambridge, Mass.: Productivity Press.

Olesen, J. (1992). *Concurrent Development in Manufacturing-based on Dispositional Mechanisms.* Unpublished PhD thesis, Technical University of Denmark, Lyngby - Denmark.

Oliver, D. E., Shahar, Y., Musen, M. A., & Shortliffe, E. H. (1999). Representation of Change in Controlled Medical Terminologies. *Artificial Intelligence in Medicine, 15*(1), 53-76.

Olsson, G., & Piani, G. (1992). *Computer Systems for Automation and Control*. New York: Prentice Hall.

Omicini, A., Zambonelli, F., Klusch, M., & Tolksdorf, R. (Eds.). (2001). *Coordination of Internet Agents: Models, Technologies, and Applications*. Berlin: Springer-Verlag.

Onori, M. (1996). *The Robot Motion Module: A Task-Oriented Robot Programming System for FAA Cells.* Unpublished PhD thesis, The Royal Institute of Technology, Stockholm.

Onori, M. (2002a). *Evolvable Assembly Systems - A New Paradigm?* In Proceedings of ISR2002 - 33rd International Symposium on Robotics, Stockholm.

Onori, M. (2002b). Product Design as an Integral Step in Assembly System Development. *Assembly Automation, 22*(3), 203-205.

Onori, M. (2002c). Viewpoint: Product Design as an Integral Step in Assembly System Development. *Assembly Automation Journal, 22*(3), 8-12.

Onori, M., Arnstrom, A., & Erixon, G. (1998). *New Concepts for Optimising FAA Cell Applications.* In Proceedings of 29th Symposium on Robotics, Birmigham - UK.

Onori, M., Barata, J., Lastra, J., & Tichem, M. (2002). *European Precision Assembly Roadmap*: Assembly-Net Project.

Onori, M., Camarinha-Matos, L. M., & Barata, J. (2003a). European Assembly - Status Report. *Assembly Automation, 23*(1), 8-12.

Onori, M., Camarinha-Matos, L. M., & Barata, J. (2003b). *European Assembly: Opportunities or Threats?* In Proceedings of ISATP2003 - 2003 IEEE International Symposium on Assembly and Task Planning, Besançon, France.

Onori, M., Langbeck, B., & Grondahl, P. (1997). The Mark III Flexible Automatic Assembly Cell. *International Journal on Robotics and Computer Integrated Manufacturing, 13*(3), 193-202.

Ontolingua (2002). [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.ksl.stanford.edu/software/ontolingua/

Ontolingua Server (2002). [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www-ksl-svc.stanford.edu:5915/

OPC (2002). [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.opcfoundation.org/

Owen, J. (1993). *STEP: an introduction*. Winchester: Information Geometers.

Padget, J., Shehory, O., Parkes, D., Sadeh, N., & Walsh, W. E. (Eds.). (2002). *Agent-Mediated Electronic Commerce IV: Designing Mechanisms and Systems* (Vol. LNAI 231). Heidelberg: Springer-Verlag.

Panzarasa, P., & Jennings, N. R. (2002). Social Influence, Negotiation and Cognition. *Simulation Modelling Practice and Theory, 10*(417-453).

Panzarasa, P., & Jennings, N. R. (2002). Social Influence, Negotiation, and Cognition. *Simulation Modelling Practice and Theory, 10*, 417-453.

Panzarasa, P., Jennings, N. R., & Norman, T. J. (2001). Social Mental Shaping: Modelling The Impact of Sociality on the Mental States of Autonomous Agents. *Computational Intelligence, 17*(43), 738-782.

Papadopoulos, G. A. (2001). Models and Technologies for the Coordination of Internet Agents: A Survey. In A. Omicini & F. Zambonelli & M. Klusch & R. Tolksdorf (Eds.), *Coordination of Internet Agents: Models, Technologies, and Applications* (pp. 25-56). Berlin: Springer-Verlag.

Parrish, D. J. (1990). *Flexible Manufacturing*. Oxford: Butterworth-Heinemann.

Parsons, S., Sierra, C., & Jennings, N. R. (1998). Agents that Reason and Negotiate by Arguing. *Journal of Logic and Computation, 8*(3), 261-292.

Parunak, H. V. D. (1998). What Can Agents Do in Industry, and Why? An Overview of Industrially-Oriented R&D at CEC. In M. Klusch & G. Weiss (Eds.), *Cooperative Information Agents II. Learning, Mobility and Electronic Commerce for Information Discovery on the Internet* (Vol. LNCS 1435, pp. 1-18). Berlin: Springer-Verlag.

Parunak, H. V. D. (1999). Industrial and Practical Applications of DAI. In G. Weiss (Ed.), *Multiagent Systems : a Modern Approach to Distributed Artificial Intelligence* (pp. 377-421). Cambridge, Masschusetts: MIT Press.

Parunak, H. V. D., Baker, A. D., & Clark, S. J. (1997). *The AARIA Agent Architecture: An Example Requirements-Driven Agent-Based System Design.* In Proceedings of First International Conference on Autonomous Agents.

Patil, R., Fikes, R., Patel-Schneider, P. F., McKay, D., Finin, T., Gruber, T., & Neches, R. (1998). The DARPA Knowledge Sharing Effort: Progress Report. In M. N. Huhns & M. P. Singh (Eds.), *Readings in Agents* (pp. 243-254). San Francisco, CA: Morgan Kaufmann Publishers.

Payne, T., Singh, R., & Sycara, K. (2002). *Facilitating Message Exchange through Middle Agents.* In Proceedings of The First International Joint Conference on Autonomous Agents and Multi-Agent Systems.

Pechoucek, M., Marik, V., & Bárta, J. (2002). A Knowledge-based Approach to Coalition Formation. *IEEE Intelligent Systems, 17*(3), 17-25.

Pfeffer, J., & Sutton, R. I. (2000). *The Knowing-doing Gap: how smart companies turn knowledge into action*. Boston, Mass.: Harvard Business School.

Philips. (1987). *CAM Reference Model* (CFT Report 13/87): Philips CFT.

Pinakis, J. (1991). *A Distributed Typeserver and Protocol for a Linda Tuple Space*: University of Western Australia.

Pine II, B. J. (1993). *Mass Customization : the new frontier in business competition*. Boston, Massachusetts: Harvard Business School Press.

Pinto, H. S., Gómez-Pérez, A., & Martins, J. P. (1999). Some Issues on Ontology Integration. In V. R. Benjamins & B. Chandrasekaran & A. Gómez-Pérez & N. Guarino & M. Uschold (Eds.), *Proceedings of the IJCAI-99 Workshop on Ontologies and Problem Solving Methods* (pp. 7.1-7.12).

Pinto, H. S., Peralta, D. N., & Mamede, N. J. (2002). Using Protégé-2000 in Reuse Processes. In J. Angele & Y. Sure (Eds.), *EON2002 - Evaluation of Ontology-based Tools* (Vol. 62, pp. 15-26). Siguenza, Spain: CEUR-WS.

Piore, M. J., & Sabel, C. F. (1984). *The Second Industrial Divide: possibilities for prosperity*. New York: Basic Books.

Poggi, A., Rimassa, G., & Turci, P. (2002). What Agent Middleware Can (And Should) Do For You. *Applied Artificial Intelligence - An International Journal, 16*(9-10), 677-698.

Porter, M. E. (1990). *The Competitive Advantage of Nations*. New York: Free Press.

Porter, M. E. (1998). Clusters and the New Economics of Competition. *Harvard Business Review, 76*(Nov-Dec), 77-90.

PPDT. (2002). *Kanban for the Shopfloor*: Productivity Inc.

Prakken, H. (1997). *Logical Tools for Modelling Legal Argument: a study of defeasible reasoning in law*. Dordrecht ; London: Kluwer Academic.

Prakken, H. (2001). Modelling defeasibility in law: logic or procedure? *Fundamenta Informaticae, 48*, 253-271.

Prakken, H. (2002). An exercise in formalising teleological case-based reasoning. *Artificial Intelligence and Law, 10*, 113-133.

Prakken, H., & Sartor, G. (1996). A dialectical model of assessing conflicting arguments in legal reasoning. *Artificial Intelligence and Law, 4*, 331-368.

Prakken, H., & Sartor, G. (2001). The Role of Logic in Computational Models of Legal Argument - a critical survey. In A. C. Kakas & F. Sadri (Eds.), *Computational Logic: Logic Programming and Beyond - Essays in Honour of Robert A. Kowalski, Part II* (Vol. LNAI 2408, pp. 342-380). Berlin: Springer-Verlag.

Prakken, H., & Sergot, M. J. (1997). Dyadic Deontic Logic and Contrary-to-duty Obligations. In D. Nute (Ed.), *Defeasible Deontic Logic* (pp. 223-262). Dordrecht: Kluwer Academic Publishers.

Preece, A., Flett, A., Sleeman, D., Curry, D., Meany, N., & Perry, P. (2001). Better Knowledge Management through Knowledge Engineering. *IEEE Intelligent Systems, 16*(1), 36-43.

PROFIBUS. (2002). *Process Field Bus* [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.fieldbus.com

Protégé-2000. (2000). *http://protege.stanford.edu* [web site]. Retrieved Jan 2002, 2002, from the World Wide Web:

PSL. (2002). *Process Specification Language* [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://ats.nist.gov/psl/

Rabelo, R. (1997). *Um Enquadramento para o Desenvolvimento de Sistemas de Escalonamento Ágil de Produção - Uma abordagem multiagente.* Unpublished PhD Thesis, Universidade Nova de Lisboa, Lisboa - Portugal.

Rabelo, R., Camarinha-Matos, L. M., & Afsarmanesh, H. (1999). Multiagent-based Agile Scheduling. *International Journal of Robotics and Autonomous Systems*(27), 15-28.

Rabelo, R. J., Afsarmanesh, H., & Camarinha-Matos, L. M. (2000). Federated Multi-Agent Scheduling in Virtual Enterprises. In L. M. Camarinha-Matos & H. Afsarmanesh & R. J. Rabelo (Eds.), *E-Business and Virtual Enterprises - Managing Business-to-Business Cooperation*. Boston: Kluwer Academic Publishers.

Rabelo, R. J., & Camarinha-Matos, L. M. (1996). Deriving Particular Agile Scheduling Systems using the HOLOS Methodology. *International Journal in Informatics and Control, 5*(2), 89-106.

Raines, P. (2001). *The Cluster Approach and the Dynamics of Regional Policy-Making* (Regional and Industrial Policy Research paper 47). Glasgow: European Policies Research Centre - University of Strathclyde.

Ranky, P. G. (1986). *Computer Integrated Manufacturing: an introduction with case studies*. Englewood Cliffs: Prentice-Hall.

Ranky, P. G. (1990). *Flexible Manufacturing Cells and Systems in CIM: a practical and consistent approach centered around powerful methodologies and technologies leading to the creation of wealth by applying computer integrated manufacturing*. Guildford: CIMware.

Rao, A. S. (1996). AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language. In v. d. Velde & J. W. Perram (Eds.), *Agents Breaking Away* (Vol. LNAI 1038, pp. 42-55). Heidelberg - Germany: Springer-Verlag.

Rao, A. S., & Georgeff, M. P. (1991). Modeling Rational Agents within a BDI Architecture. In J. Allen & R. Fikes & E. Sandewal (Eds.), *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning - KR'91* (pp. 473-484). San Mateo, CA, USA: Morgan Kaufmann publishers Inc.

Rao, A. S., & Georgeff, M. P. (1995, June 1995). *BDI Agents from Theory to Practice.* In Proceedings of Proceedings of the First International Conference on Multiagent Systems - ICMAS'95, San Francisco - USA.

Rapoport, A. (1970). *N-person Game Theory: concepts and applications*. Ann Arbor,.

Redford, A. H. (1989). Error Recovery in Assembly by Robot. *Advanced Manufacturing Engineering, 1*, 109-112.

Reeves, D., Wellman, M., & Grosof, B. (2000). *Automated Negotiation from Declarative Contract Descriptions.* In Proceedings of KBEM'00 - AAAI-2000 Workshop on Knowledge-based Electronic Markets, Menlo Park, CA.

Reeves, D., Wellman, M., & Grosof, B. (2001, May). *Automated Negotiation from Declarative Contract Descriptions.* In Proceedings of Agents'01 - Fifth International Conference on Autonomous Agents, Montreal, Canada.

Rembold, U., Blume, C., & Dillmann, R. (1985). *Computer-Integrated Manufacturing Technology and Systems*. New York: Dekker.

RMI. (2001). *Sun Microsystems. Java Specification. The Remote Method Invocation.* http://java.sun.com [web site]. Retrieved Jan 2002, 2002, from the World Wide Web:

Rosenschein, J. S., & Zlotkin, G. (1994). *Rules of Encounter*: The MIT Press.

Rosenschein, S. J., & Kaelbling, L. P. (1996). A Situated View of Representation and Control. In P. E. Agre & S. J. Rosenschein (Eds.), *Computational Theories of Interaction and Agency* (pp. 515-540). Cambridge, Mass: MIT Press.

Rossi, D., Cabri, G., & Denti, E. (2001). Tuple-based Technologies for Coordination. In A. Omicini & F. Zambonelli & M. Klusch & R. Tolksdorf (Eds.), *Coordination of Internet Agents: Models, Technologies, and Applications* (pp. 83-109). Berlin: Springer-Verlag.

Rowstron, A., Douglas, A., & Wood, A. (1995). A Distributed Linda-like Kernel for PVM. In J. Dongarra & G. M. & B. Tourancheau & X. Vigouroux (Eds.), *EuroPVM'95* (pp. 107-112).

Rubin, D. L., Hewet, M., Oliver, D. E., Klein, T. E., & Altman, R. B. (2001). *Automating Data Acquisition into Ontologies from Pharmacogenetics Relational Data Sources Using Declarative Object Definitions and XML.* In Proceedings of Pacific Symposium on Biocomputing, Kauai, Hawai.

Runge, A. (1998). *The Need for Supporting Electronic Commerce with Electronic Contracting.* In Proceedings of INFORMS - Conference on Information Systems and Technology, Montreal, Canada.

Runge, A., Schopp, B., & Stanoevska-Slabeva, K. (1999). *The Management of Business Transactions through Electronic Contracts.* In Proceedings of DEXA'99 - 10th International Workshop on Database and Expert Systems Applications, Florence, Italy.

Sacerdoti, E. D. (1975). Planning in a Hierarchy of Abstraction Spaces. *Artificial Intelligence, 5*(2), 115-135.

Sacerdoti, E. D. (1977). *A Structure for Plans and Behavior*. New York: Elsevier.

Sachs, M. W., Dan, A., Nguyen, T. N., Kearney, R., Shaikh, H. H., & Dias, D. M. (2000). *Executable Trading-Partner Agreements in Electronic Commerce*. Yorktown Hts, NY: IBM T. J. Watson Research Center.

Sandholm, T. W. (1999). Distributed Rational Decision Making. In G. Weiss (Ed.), *Multiagent Systems : a Modern Approach to Distributed Artificial Intelligence* (pp. 201-258). Cambridge, Masschusetts: MIT Press.

Santos, J. P. O., Ferreira, J. J. P., & Mendonca, J. M. (2000). A modelling language for the design and execution of enterprise models in manufacturing. *International Journal of Computer Integrated Manufacturing, 13*(1), 1-10.

Sartor, G. (2001). Why Agents Comply with Norms, and Why they Should. In R. Conte & C. Dellarocas (Eds.), *Social Order in Multiagent Systems* (pp. 19-43). Boston: Kluwer Academic Publishers.

Scheer, A.-W. (1991). *CIM: Computer Integrated Manufacturing: towards the factory of the future* (2. rev. and enl. ed.). Berlin ; New York: Springer.

Scherer, E. (Ed.). (1998). *Shop Floor Control - A Systems Perspective*. Berlin: Springer-Verlag.

Schmid, B., & Lindemann, M. (1998, 6-9 Jan). *Elements of a Reference Model for Electronic Markets*. In Proceedings of HICCS'98 - Thirty-First Annual Hawaii International Conference on System Sciences, Hawaii.

Schreiber, A. T. (1999). *Knowledge Engineering and Management: the CommonKADS methodology*. Cambridge, Mass.: MIT Press.

Schreiber, A. T., Wielinga, B. J., De Hoog, R., Akkermans, J. M., & Van de Velde, W. (1994). CommonKADS: A Comprehensive Methodology for KBS Development. *Ieee Intelligent Systems & Their Applications, 9*(6), 28-37.

Searle, J. R. (1969). *Speech Acts - An Essay in the Philosophy of Language* (1999 ed.). Cambridge: Cambridge University Press.

Semantic Web (2002). [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.semanticweb.org/

Sergot, M. J., Sadri, F., Kowalski, R., Kriwaczek, F., & Cory, H. T. (1986). The British Nationality Act as a Logic Program. *Communications of the ACM, 29*(5), 370-386.

Shehory, O., & Kraus, S. (1999). Feasible Formation of Stable Coalitions among Autonomous Agents in Non-super-additive Environments. *Computational Intelligence, 15*(3), 218-251.

Shen, W., & Norrie, D. H. (1999). Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey. *Knowledge and Information Systems, an International Journal, 1*(2), 129-156.

Sheth, A. P., & Larson, J. A. (1990). Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys, 22*(3), 183-235.

Shils, E. (1960). Mass Society and Its Culture. *Daedalus, 89*(2).

Shoham, Y., & Tennenholtz, M. (1995). On Social for Artificial Agent Societies: Off-Line Design. *Artificial Intelligence, 73*(1-2), 231-252.

Sichman, J. S. (1998). DEPINT: Dependence-Based Coalition Formation in an Open Multi-Agent Scenario. *Journal of Artificial Societies and Social Simulation, 1*(2), <http://www.soc.surrey.ac.uk/JASSS/1/2/3.html>.

Sichman, J. S., Conte, R., Demazeau, Y., & Castelfranchi, C. (1998). A Social Reasoning Mechanism Based on Dependence Networks. In M. N. Huhns & M. P. Singh (Eds.), *Readings in Agents* (pp. 416-420). San Francisco, CA: Morgan Kaufmann Publishers.

Siegel, J. (2000). *CORBA 3: fundamentals and programming* (2. ed.). New York: Wiley.

Silver, E. A., Pyke, D. F., & Peterson, R. (1998). *Inventory Management and Production Planning and Scheduling* (3. ed.). New York: Wiley.

Sloan, A. P. (1986). *My Years with General Motors*. Harmondsworth, Middlesex: Penguin.

Smith, A. (2000). *The Wealth of Nations*. New York;London: Random House International;Hi Marketing.

Smith, R. G. (1980). The Contract Net Protocol - High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers, 29*(12), 1104-1113.

SOAP. (2003). *Simple Object Access Protocol* [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.w3.org/TR/SOAP/

Soares, A. M. L. S. (1998). *Modelação para Optimização Técnico-Organizacional de Sistemas Integrados de Fabrico*. Unpublished Ph.D., Faculdade de Engenharia da Universidade do Porto, Porto.

Sowa, J. F. (2000). *Knowledge Representation: Logical, Philosophical, and Computational Foundations*: Brooks Cole Publishing Co.

Sowa Ontology (2000). [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://users.bestweb.net/~sowa/ontology/

Spithoven, A. H. G. M. (2001). Lean Production and Disability. *International Journal of Social Economics, 28*(9), 725-741.

Stark, J. (1989). *Handbook of Manufacturing Automation and Integration*. Boston: Auerbach.

Steels, L. (1990). Cooperation Between Distributed Agents Through Self Organisation. In Y. Demazeau & J.-P. Müller (Eds.), *Decentralized A.I.: Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World* (pp. 263). Amsterdam: North-Holland.

Strackbein, R. (1998). *How to Succeed in the Information Age* [Web Site]. Retrieved January 2003, from the World Wide Web: http://www.strackbein.com

Strackbein, R. (2002). *How to Succeed in the Information Age* [Web Site]. Retrieved January 2003, from the World Wide Web:

Strobel, M. (2000). On Auctions as the Negotiation Paradigm of Electronic Markets. *Journal of Electronic Markets, 10*(1), 39-44.

Strobel, M. (2001). Design of Roles and Protocols for Electronic Negotiations. *Electronic Commerce Research Journal, 1*(3), 335-353.

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge Engineering: Principles and Methods. *IEEE Transactions on Data and Knowledge Engineering, 25*(1-2), 161-197.

SUO (2002). [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://grouper.ieee.org/groups/suo/

Susskind, R. E. (1998). *The Future of Law: facing the challenges of information technology*. Oxford: Clarendon.

Susskind, R. E. (2000). *Transforming the Law: essays on technology, justice and the legal marketplace*. Oxford: Oxford University Press.

Svensson, G. (2001). Just-in-Time: the reincarnation of past theory and practice. *Management Decision, 39*(10), 866-879.

Sycara, K. (1989). Multiagent Compromise via Negotiation. In L. Gasser & M. N. Huhns (Eds.), *Distributed Artificial Intelligence* (Vol. 2, pp. 119-138). Los Altos, CA: Morgan Kaufman Publishers.

Sycara, K. (1990). Persuasive Argumentation in Negotiation. *Theory and Decision, 28*(3), 203-242.

Sycara, K. (1992). The PERSUADER. In S. C. Shapiro (Ed.), *Encyclopedia of Artificial Intelligence*. New York: Wiley.

Sycara, K., Decker, K., & Williamson, M. (1997). *Middle-Agents for the Internet.* In Proceedings of IJCAI-97 International Conference on Artificial Intelligence, Nagoya - Japan.

Sycara, K., Paolucci, M., Van Velsen, M., & Giampapa, J. A. (2001). *The RETSINA MAS Infrastructure* (CMU-RI-TR-01-05). Pittsburgh, PA: Robotics Institute, Carnegie Mellon University.

Szegheo, O., & Petersen, S. A. (2000). Extended enterprise engineering - A model-based framework. *Concurrent Engineering-Research and Applications, 8*(1), 32-39.

Tambe, M., Pynadath, D. V., & Chauvat, N. (2000). Building Dynamic Agent Organizations in Cyberspace. *IEEE Internet Computing, 4*(2), 65-73.

Tarasewich, P., & McMullen, P. R. (2002). Swarm Intelligence: Power in Numbers. *Communications of the ACM, 45*(8), 62-67.

Tate, A., Bradshaw, J. M., & Pechoucek, M. (2002). Knowledge System for Coalition Operations. *IEEE Intelligent Systems, 17*(3), 14-16.

Taylor, F. W. (1998). *The Principles of Scientific Management*. Norcross, GA: Engineering & Management Press.

Teich, J., Wallenius, H., & Wallenius, J. (1999). Multiple-issue Auction and Market Algorithms for the World Wide Web. *Decision Support Systems, 26*(1), 49-66.

Teicholz, E., & Orr, J. N. (1987). *Computer Integrated Manufacturing Handbook*. New York: McGraw-Hill.

Teixeira, C. (1996). *Organização do Trabalho e Factor Humano*. Lisboa: IEFP.

Tennison, J., O'Hara, K., & Shadbolt, N. (2002). APECKS: Using and Evaluating a Tool for Ontology Construction with Internal and External KA Support. *International Journal of Human-Computer Studies, 56*(4), 375-422.

Teunis, G., Leitão, P., & Madden, M. (1998). *A New Architecture for Flexible Shop Control Systems.* In Proceedings of Integration in Manufacturing, Gotenborg - Sweden.

Tharumarajah, A., Wells, A. J., & Nemes, L. (1996). Comparison of the bionic, fractal and holonic manufacturing system concepts. *International Journal Computer Integrated Manufacturing, 9*(3), 217-226.

THINKcreative. (2002). *THINKcreative - Thinking network of experts on emerging smart organizations (IST Project)* [Web Site]. Retrieved Nov 2002, from the World Wide Web: http://www.thinkcreative.org

Tichem, M. (2000). *Position Report on Flexible Assembly Automation*. Delft: Delft University of Technology.

Toffler, A. (1981). *The Third Wave* (New ed.). London: Pan in assoc. with Collins.

Tolksdorf, R. (1997). Berlinda: An Object-Orinted Platform for Implementing Coordination Languages in Java. In D. Garlan & D. Le Métayer (Eds.), *Coordination Languages and Models - Proceedings of the 2nd International Conference (COORDINATION'97)* (Vol. 1282 - LNCS, pp. 430-433). Berlin: Springer-Verlag.

Trist, E. L. (1963). *Organizational choice: capabilities of groups at the coal face underr changing technoligies: the loss,rediscovery & transformation of a work tradition*. London: Tavistock.

Tsvetovat, M., Sycara, K., Chen, Y., & Ying, J. (2001). Customer Coalitions in Electronic Markets. In F. Dignum & U. Cortés (Eds.), *Agent-Mediated Electronic Commerce* (Vol. LNAI 2003, pp. 121-138). Berlin: Springer-Verlag.

Tveit, A. (2001). *A Survey of Agent-Oriented Software Engineering.* In Proceedings of First NTNU CSGS Conference.

UBL. (2003). *Universal Business Language* [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl

UDDI. (2003). *Universal Description, Discovery and Integration* [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.uddi.org

UEML. (2003). *Unified Enterprise Modelling Language* [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.ueml.org

UN/CEFACT. (2003). *United Nations Centre for Trade Facilitation and Electronic Business* [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.unece.org/cefact/

UN/EDIFACT. (2003). *United Nations Directories for Electronic Data Interchange for Administration, Commerce and Transport* [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.unece.org/trade/untdid/welcome.htm

UPnP. (2003). *Universal Plug and Play* [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.upnp.org

Uschold, M., & Gruninger, M. (1996). Ontologies: Principles, methods and applications. *Knowledge Engineering Review, 11*(2), 93-136.

Uschold, M., King, M., Moralee, S., & Zorgios, Y. (1998). The Enterprise Ontology. *Knowledge Engineering Review, 13*(1), 31-89.

Valckenaers, P. (1997). *MASCADA - Esprit Project LTR 22728* [Web Site]. Retrieved Feb 2003, 2002, from the World Wide Web: http://www.mech.kuleuven.ac.be/pma/project/mascada/welcome.html

Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., & Peeters, P. (1998). Reference Architecture for Holonic Manufacturing Systems: PROSA. *Computers in Industry, 37*(3), 255-274.

Van der Vet, P. E., Speel, P. H., & Mars, N. J. I. (1993). *The PLINIUS Ontology of Ceramic Materials.* In Proceedings of ECAI'94 - Eleventh European Conference on Artificial Intelligence, Amsterdan, The Netherlands.

Vernadat, F. B. (1996). *Enterprise Modeling and Integration: principles and applications* (1. ed.). London: Chapman & Hall.

Vernadat, F. B. (1999). Research Agenda for Agile Manufacturing. *International Journal of Agile Management Systems, 1*(1), 37-40.

Vickrey, W. (1961). Counterspeculation, Auctions, and Competitive Sealed Tenders. *The Journal of Finance, 16*(1), 8-37.

Victor, B., & Boynton, A. C. (1998). *Invented Here: maximizing your organization's growth and internal profitability*. Boston, Mass.: Harvard Business School.

Vieira, W. (2000). *Agentes Móveis Adaptáveis para Operação Remota.* Unpublished PhD, Universidade Nova de Lisboa, Lisboa.

VOmap. (2002). *VOmap - Roadmap design for collaborative virtual organizations in dynamic business ecosystems (IST Project)* [Web Site]. Retrieved Nov 2002, from the World Wide Web: http://www.vomap.org

Vos, J. A. W. M. (2001). *Module and System Design in Flexible Automated Assembly.* Unpublished PhD Thesis, Delft University Press, Delft.

VOSTER. (2002). *VOSTER - Virtual Organisation cluSTER (IST Project)* [Web Site]. Retrieved Nov 2002, from the World Wide Web: http://cic.vtt.fi/projects/voster/public.html

W3C. (2002). *Web Services Architecture* [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.w3.org/TR/2003/WD-ws-arch-20030808/

Waldner, J.-B. (1992). *CIM: principles of computer integrated manufacturing*. Chichester: Wiley.

Watt, S. N. K. (1996). Artificial societies and psychological agents. *Bt Technology Journal, 14*(4), 89-97.

WebOnto (2002). [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://webonto.open.ac.uk/

Weiss, G. (Ed.). (1999). *Multiagent Systems : a modern approach to distributed artificial intelligence*. Cambridge, Masschusetts: MIT Press.

WFMC. (2002). *Workflow Management Coalition* [web site]. Retrieved October 2002, 2002, from the World Wide Web: http://www.wfmc.org/

Wiederhold, G. (1992). Mediators in the Architecture of Future Information Systems. *IEEE Computer Systems, 25*(3), 38-49.

Williams, T. J. (1994). The Purdue Enterprise Reference Architecture. *Computers in Industry, 24*(2-3), 141-158.

Williams, T. J., Bernus, P., Brosvic, J., Chen, D., Doumeingts, G., Nemes, L., Nevins, J. L., Vallespir, B., Vlietstra, J., & Zoetekouw, D. (1994). Architectures for Integrating Manufacturing Activities and Enterprises. *Computers in Industry, 24*(2-3), 111-139.

Williams, T. J., Bernus, P., & Nemes, L. (1996). *Architecture for Enterprise Integration* (1. ed ed.). London: Chapman & Hall.

Wobbe, W. (1995). Anthropocentric Production Systems: A new Leitbild for an industrial symbiotic work and technology in Europe. In J. Benders & J. De Haan & D. Bennet (Eds.), *The Symbiosis of Work and Technology* (pp. 13-24). London: Taylor & Francis.

Wobbe, W., & Communities, C. o. t. E. (1992). *What are anthropocentric production systems ? Why are they a strategic issue for Europe ?* : Directorate-General Science Research and Development.

Womack, J. P., Jones, D. T., & Roos, D. (1990). *The Machine That Changed the World*. New York: Harper Perennial.

Wong, H. C., & Sycara, K. (2000). *A Taxonomy of Middle-Agents for the Internet.* In Proceedings of Fourth International Conference on MultiAgent Systems.

Wooldridge, M., & Jennings, N. R. (1994). *Agent Theories, Architectures, and Languages: A Survey.* In Proceedings of ECAI-Workshop on Agent Theories, Architectures and Languages, Amsterdam - The Netherlands.

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

Wooldridge, M. J. (2000). *Reasoning about Rational Agents*. Cambridge, Massachusetts; London: MIT Press.

Wooldridge, M. J. (2002). *An Introduction to Multiagent Systems*. New York: J. Wiley.

WSDL - Web Services Description Language (2002). [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.w3.org/TR/wsdl12/

Wurman, P. R., Walsh, W. E., & Wellman, M. P. (1998). Flexible Double Auctions for Electronic Commerce: Theory and Implementation. *Decision Support Systems, 24*(1), 17-27.

Wyckoff, P., McLaughry, S., Lehman, T., & Ford, D. (1998). T Spaces. *IBM Systems Journal, 37*(3), 454-474.

Wyns, J. (1999). *Reference Architecture for Holonic Manufacture - the key to support evolution and reconfiguration.* Unpublished PhD thesis, Katholieke Universiteit Leuven, Leuven.

xCBL. (2003). *XML Common Business Library* [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.xcbl.org

XML. (2000). *XML - eXtensible Markup Language* [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.w3.org/XML/

XML/EDI (2003). [web site]. Retrieved April 2003, 2003, from the World Wide Web: http://www.xmledi-group.org/

Yates, D. (1978). *Exclusion Clauses in Contracts*. London: Sweet&Maxwell.

Zambonelli, F., Jennings, N. R., Omicini, A., & Wooldridge, M. (2001). Agent-Oriented Software Engineering for Internet. In A. Omicini & F. Zambonelli & M. Klusch & R. Tolksdorf (Eds.), *Coordination of Internet Agents: Models, Technologies, and Applications* (pp. 326-346). Berlin: Springer-Verlag.

Zeng, D., & Sycara, K. (1996). How Does an Agent Learn to Negotiate? In J. P. Muller & M. Wooldridge & N. R. Jennings (Eds.), *Intelligent Agents III: Theories, Architectures and Languages* (Vol. LNAI 1193, pp. 233-244). Heidelberg: Springer-Verlag.

Zhang, Q., & Cao, M. (2002). Business Process Reengineering for Flexibility and Innovation in Manufacturing. *Industrial Management & Data Systems, 102*(3), 146-152.

Zurawski, R., & Zhou, M. C. (1994). Petri Nets and Industrial Applications - a Tutorial. *IEEE Transactions on Industrial Electronics, 41*(6), 567-583.

Zwegers, A. (1998). *On Systems Architecting - a study in shop floor control to determine architecting concepts and principles.* Unpublished PhD Thesis, Eindhoven Technical University, Eindhoven - The Netherlands.

Appendix A – External specialists

Specialist 1 - Eng. Luís Flores

Eng. Luís Flores is a specialist from the industry who was asked to test and evaluate the prototype under the scenario introduced in chapter 5. He was also asked to investigate its application to shop floor production systems.

After attaining his degree in Computer Science in 1997 Eng. Luís Flores joined the English line builder Automated System Services UK (ASS), whose core business is the construction of robotised welding stations for the Automotive Industry. During the course of his collaboration with Automated System Services he came by some of the biggest automation projects that occurred in Europe. The most relevant are:

1. Ford Focus – Saarlouis, Germany, Control Systems Engineering for Framing System.
2. Ford Transit – Genk, Belgium, Control Systems Engineering for Framing and Autoroof Systems.
3. Ford Mondeo – Genk, Belgium, Senior Control Systems Engineering for Body Welding.
4. Ford Fiesta – Cologne Germany, Senior Control Systems Engineering for Body Welding.

His activity within line building ceased in 2002 when he started his own company – IntRoSys SA, which is shaped to deliver to the shop floor what scientists believe are the solutions for the problems of building machinery for human operators.

His report is presented below in italics:

*Due to my past and present experience Mr. Barata de Oliveira asked me to evaluate the potential of the **CoBASA** prototype and to investigate its application in Shop Floor production systems. It's with a great sense of honour that I and my company IntRoSys acknowledged Mr. Barata's request.*

Control Systems do not keep up with the mechanical/process complexity of recent manufacturing systems. Programmable Logic Controllers are very good at performing low-level tasks associated with mechanical sequencing but they lack high-level abstraction capability more suited to for variant type handling, production scheduling and machinery changeover. Control Systems are developed in a bottom up fashion, starting from the basic actuator/initiator pair (for e.g. two position valve control) through station control, line control to high level variance management. All of the development stages are implemented at bit-level using ladder logic representation.

After testing CoBASA in the NovaFlex scenario, and have fully understood the principles behind it I was positively impressed by the potential offered by this architecture. The idea of having a tool to support reengineering is very much welcomed and was being felt by shop floor personnel as something missing. In fact, my experience working in different European sites has shown that more and more effort is spent by shop floor personnel in changing or adaptations (shop floor reengineering). I believe that the CoBASA prototype, if properly transformed into a commercial application, can be very useful for manufacturing companies. Therefore, it is my conviction that the ***CoBASA*** *architecture is a breakthrough concept that provides the high-level control architecture that system integrators require as the systems built by them become more complex. One of premises for a fruitful* ***CoBASA*** *integration among today's manufacturing systems is that real time control systems cannot be replaced.* ***CoBASA*** *should be looked at as a tool that will enable it to reconfigure manufacturing systems for product variance at a higher level. Currently these operations are provided by PLCs however, due to their electrical past inheritance, they are not suitable for management and brokering in systems that produce different types of product. Today one third of the time is spent developing the control for the mechanical cycling of the installations while the rest is spent implementing the management of the system at bit-level.*

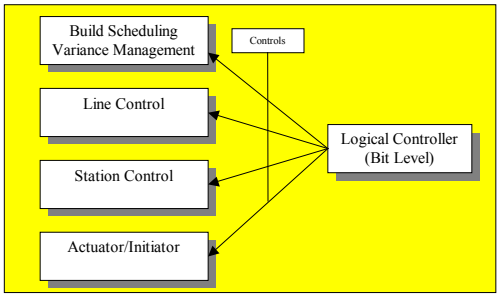

Figure A-51 - Before CoBASA.

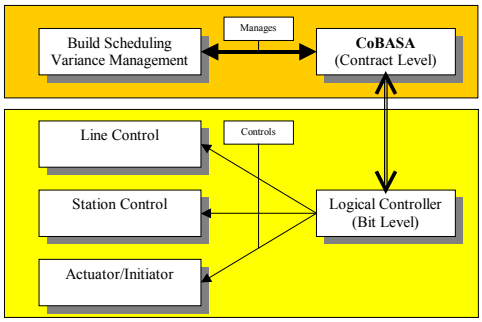

Figure A-52 - With CoBASA.

CoBASA*'s use is obvious as a system that will bridge the gap between production scheduling and machinery control, at the present time this gap is filled by relay logic running on PLCs (Figure A-52).*

Because I am so convinced of the advantages of CoBASA I decided to describe within this brief report a possible application of CoBASA in one real case.

A Possible application for CoBASA: The B256/B226/B257 Project

During the year 2002 the Italian system integrator COMAU deployed a complete tooling system to manufacture B256/B226/B25 in Ford Plants located in Cologne and Valencia. These references represent Ford Fiesta variances, respectively the standard Ford Fiesta, the Ford Fusion, and the Ford Fiesta Van. A single manufacturing system had to accommodate the production of the three variant types. Due to major differences in product engineering, integrating the three variant types in the same production system proved to be a very complex engineering task due to:

1. *Three kinds of geometry tools – one for each model.*
2. *Three independent robot programs – one for each model.*
3. *Highly complex production scheduling management.*

Customer specifications demanded that the integration of a new model should be achieved by installing a new set of geometry tools in the existing systems. This approach implied that variant type management had to be implemented with high flexibility. Nevertheless high-level tooling/type management infrastructure was lacking. Contracts between geometry tools, transport facilities and robots in order to build a subassembly were painfully celebrated by the Control System Engineer at bit level in relay logic representation.

What CoBASA could do for B256/B226/B257

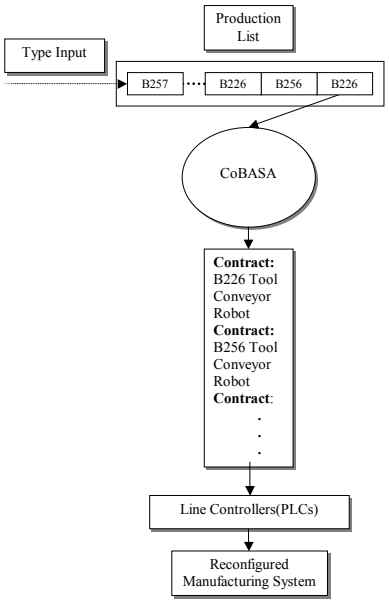

Figure A-53 - Detailing CoBASA operation in the B226/B256/B257 project

The following description should be read taking into account Figure A-53. *Relay logic is good for direct machine control. It is simple, well proven and maintenance friendly. A CoBASA driven*

architecture still requires a hardware level control. It can make a difference if used to celebrate the contracts required to reconfigure machinery to the next variant. For example, the type FIFO from Paint Shop requires 300 B226 while the machine is configured to run B256. On the first stage CoBASA will terminate the contracts established between B256 geometric tools, transport system and robots. On the second stage CoBASA will celebrate a contract between B226 geometric tools, transport system and robots. The end result will be a changeover operation that will accommodate the following build. The contract will reflect the physical reality of a reconfigured line and selection of a different process execution plan for the robots.

Specialist 2 - Eng. Alberto Gavinhos

Eng. Alberto Gavinhos is the maintenance manager of the body shop at AUTOEUROPA, where he has worked since 1993. One of his important tasks is the coordination and management of all the projects involving changes and adaptations to the stamping and body shop floors. Therefore, he is highly aware of how important shop floor reengineering is becoming for a competitive manufacturing company.

His report is presented below in italics:

When I started at Autoeuropa in May 1993 I was faced with a complex production system organized for a common target – TO PRODUCE CARS. From 1993 to 2003 indicators like OEE, MTBF, and MTTR, or expressions like Maintainability & Reliability, Training, Reengineering, Team Work, FMEA, and Breakdown Analyses have been part of my professional life. My goal has been to develop them in a Continuous Improvement Strategy. One of my partners on this Continuous Improvement Strategy was UNINOVA, through Mr. Barata de Oliveira. Due to my background and due to this partnership, I was asked to analyse the potential of CoBASE prototype as well as to suggest a possible application in a Shop Floor production system.

CoBASA (comments / suggestions)

From what I have understood and tested, the philosophy behind CoBASA seems to be a valuable approach with some interesting concepts that can help solve the problem of shop floor adaptation or change. It can be the first step for future projects implementation in the Shop Floor. I foresee three areas where CoBASA concepts can be applied:

1. *Skills Management System - Each technician should be defined as an Agent with certain skills. A contract should be raised with the best combination of Agents to act if anything happens (e.g. a breakdown). The agent credit system should be used to establish the technicians training needs.*

2. *Tool & Die Management System - In a standard Stamping area each Press has several Die Sets to work with. If each Die is characterized as an Agent the correct credit evaluation can be a powerful tool for Tool & Die Management and follow-up. Notice that the Die Set changing in each Press happens in a really dynamic way.*
3. Support for Automation Integrators - In order to provide the best performance, each facility must have "state of the art" engineering. Integrating Geometric Tools, Transport Systems, Robots and Measuring Devices to build an Assembly, is a very hard job for the Control System Engineer. Therefore, CoBASA can be a breakthrough solution for this difficulty. For the future and using CoBASE, the Agent characterization should meet certain skills and credits applicable to equipment/ systems in order to have the best contracts allowing the best integration.