We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

154
Countries delivered to

Our authors are among the

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us? Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

2

The Cobasa Architecture as an Answer to Shop Floor Agility

Jose Barata

1. Introduction

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.

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

Source: Manufacturing the Future, Concepts - Technologies - Visions , ISBN 3-86611-198-3, pp. 908, ARS/plV, Germany, July 2006, Edited by: Kordic, V.; Lazinica, A. & Merdan, M.

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 et al. 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 and Edwards 1999; Koren et al. 1999; Mehrabi et al. 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 altera-

tion projects (Barata and 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.

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 this chapter is focused on the reengineering aspects required by the control/supervision architecture, which covers an important part of any global life cycle support methodology.

The proposed architecture to improve shop floor reengineering (CoBASA) aims at accommodating the following requirements:

- **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 the proposed architecture. The main question being addressed in this chapter and which the CoBASA architecture intends to answer is highlighted below:

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

The hypothesis formulated as a basis for CoBASA to address the previous question is defined below:

Hypothesis
Shop floor control/supervision reengineering agility can be achieved if manufacturing systems are abstracted as compositions of modularised manufacturing components (modular approach) 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 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 are 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 proposed architecture Coalition Based Approach for Shopfloor Agility – CoBASA to answer the question raised above is a multiagent based architecture that supports the reengineering process of shop floor control/supervision architectures. In an innovative way, CoBASA uses contracts to govern the relationships between coalition members (manufacturing agents) and postulates a

new methodological approach in which the reengineering process is included within the life cycle. Since the CoBASA approach is based on the concept of manufacturing modules that might be reused, it requires the manufacturing community to structure and classify the process involved, thus leading to a more systematic or structured methodological approach.

Therefore the CoBASA concept considers modularity and plugability as one of its most important foundations principles. The control system architecture being proposed considers that each basic components are modules of manufacturing components that can be reused and plugged or unplugged with reduced programming effort, supporting in this way the plug & produce metaphor.

CoBASA 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.

2. CoBASA Basic foundations

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 CoBASA assumes that there are similarities between the reengineering process and the formation of consortia regulated by contracts in networked organisations. 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.

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 in-

creasing 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 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.

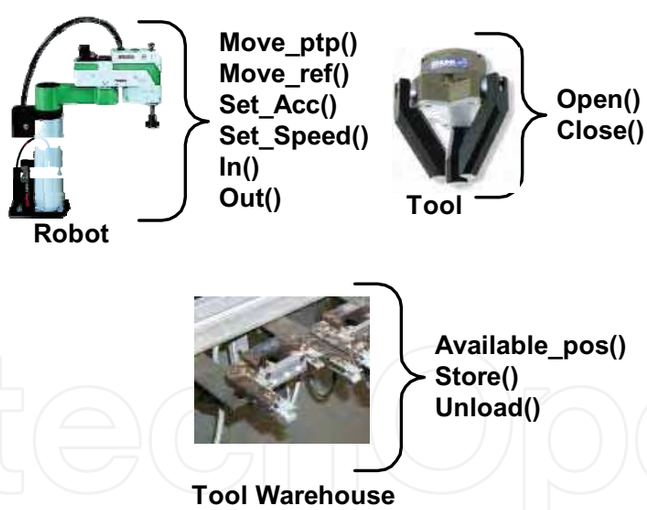

Figure 1. Example of basic manufacturing components and core competencies

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. In the architecture being proposed every manufacturing component e.g. robots, tools, fixing devices, is associated to an agent that represents its behaviour (agentified manufacturing component). When these agents interact or cooperate they can generate aggre-

gated 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 1 - Manufacturing component or module
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 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 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 2).

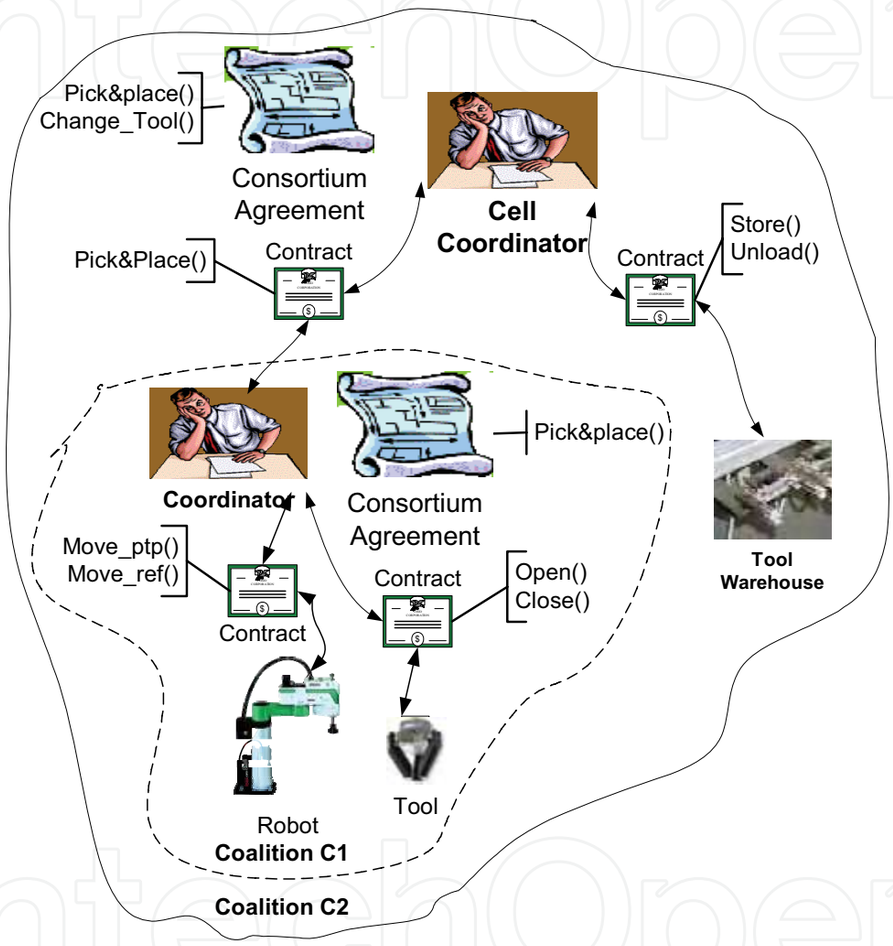

Figure 2. Consortia example

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 - 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 **3** shows how manufacturing agents, cluster, and coalition interrelate. Agentified components in the same "geographical" area of the shop-floor join the same cluster.

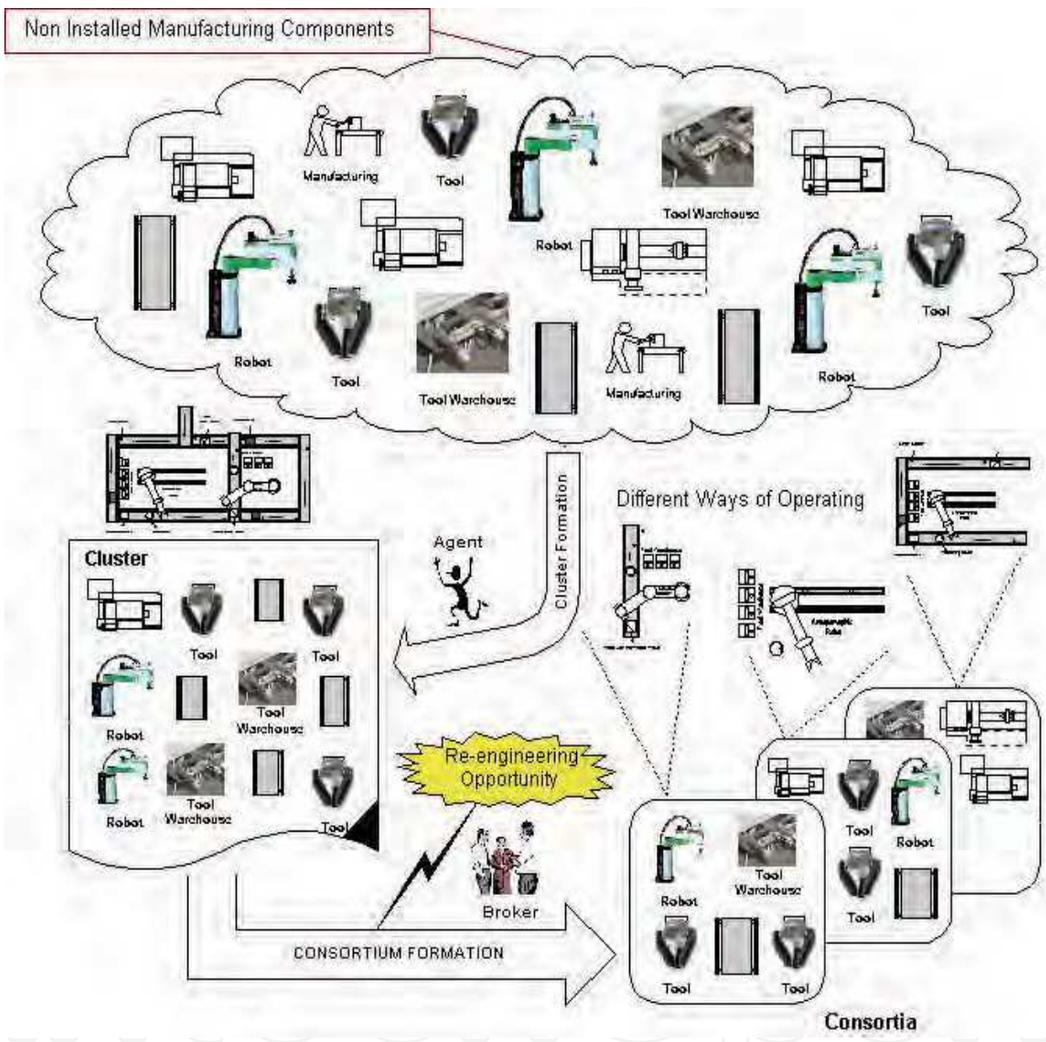

Figure 3. Consortia formation

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 et al. 1997; Camarinha-Matos et al. 1996).

3. CoBASA architecture

The basis 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 and Graesser 1997;

Weiss 1999; Wooldridge and Jennings 1995; Wooldridge 2000; Wooldridge 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 and Dellarocas 2001). 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.

3.1 The components

The basic components of the CoBASA architecture are:

- Manufacturing Resource Agents,
- Coordinating Agent, Broker Agent,
- Cluster Manager Agent,
- and Contract.

Definition 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 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.

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 individu-

alised 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.

Definition 6 – 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.

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.

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.

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). 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*.

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.

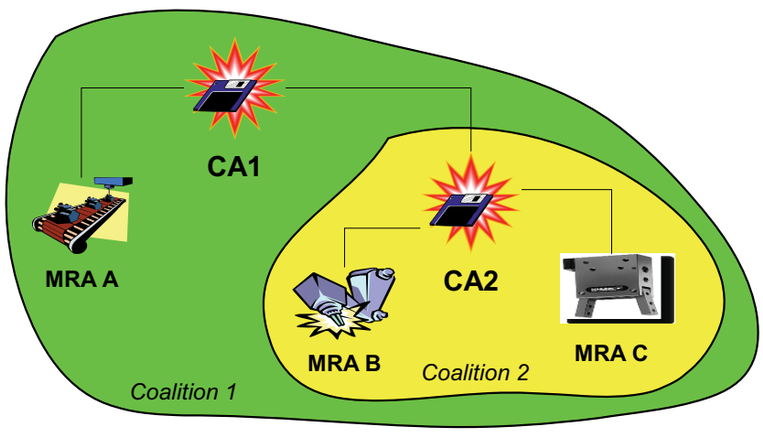

Figure 4. Hierarchy of coalitions/consortia

Definition 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.

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.

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.

Definition 8 – 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.

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 et al. 2000; Klusch and Sycara 2001; Payne et al. 2002; Sycara et al. 1997; Wiederhold 1992; Wong and Sycara 2000).

The broker therefore interacts with the human, the cluster, and the candidate members to the consortium. Coalitions/consortia can be created either automatically or manually. At the current stage only the manual option is considered. The main interactions between the concepts that have been referred to are shown in Figure 5. Contracts are the next important CoBASA mechanism, which is used to regulate the MRAs and CAs interaction with a CMgA as well as the behaviour within the coalition.

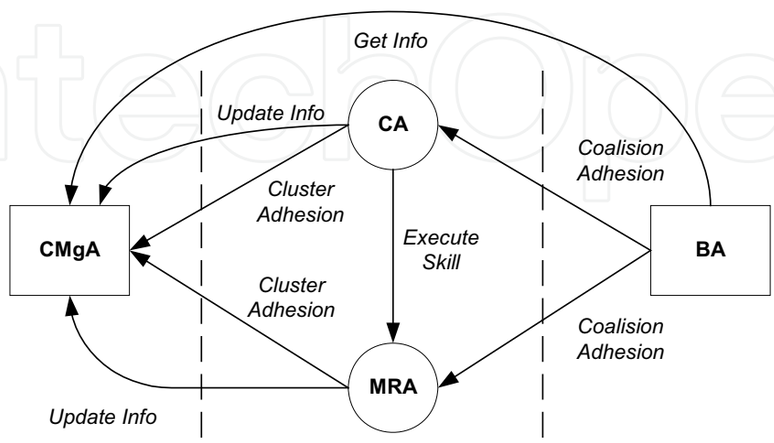

Figure 5. Interactions among the main components

In the CoBASA architecture two type of contracts are considered: **cluster adhesion contract (CAC),** and **multilateral consortium contract (MCC).**

Definition 9 – Cluster Adhesion Contract (CAC)
This contract regulates the behaviour of the MRA when interacting with a cluster. Since the terms imposed by the cluster cannot be negotiable by the MRA the contract type is "adhesion". The CMgA offers cluster services in exchange for services (abilities or skills) from the MRA.

The CAC includes terms such as the ontologies that must be used by the candidate, the duration of the membership, the consideration (a law term that describes what the candidate should give in turn of joining the cluster, usually the skills that the candidate is bringing to the cluster).

Definition 10 – Multilateral Coalition/consortium Contract (MCC)
This contract regulates the behaviour of the coalition by imposing rights and duties to the coalition members. The contract identifies all members and must be signed by them to be effective. The coalition leader (CA) is identified as well as its members. The members are entitled to a kind of award (credit) in exchange for their skills.

The important terms of this type of contract other the usual ones like duration, names of the members, penalties, etc., are the consideration and the individual skills that each member brings to the contract. Note that the skills involved in a specific consortium contract may be a subset of the skills offered by the involved agent when it joins the cluster. The importance of contracts as a mechanism to create/change flexible and agile control structures (consortia) lays on the fact that the generic behaviours exhibited by generic agents are constrained by the contract that each agent has signed. This calls forth that different consortium behaviours can be achieved by just changing the terms of the consortium contract, namely the skills brought to the consortium.

MCCs represent simultaneously a coordination mechanism and a mean to facilitate coalitions/consortia dynamics. Since a coalition/consortium is created, changed, and terminated mainly through contract operations, the task of grouping manufacturing components able to perform certain tasks (coalition) is facilitated. In addition, the introduction of new components to this group involves only contract configurations. Agility is thus achieved since moving components from one organisational form to another involves only configuration instead of programming effort.

3.2 Coalition dynamics

Since CAs are able to generate new skills from the set of skills brought in by its members, coalitions enable the creation of completely different control structures. This could not ever be achieved using a traditional control architecture because of its rigidity. Traditional 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 considering also that coalitions can be composed of other coalitions, the next question is what impact a change on a coalition has on the whole structure. This impact might happen because after a change on 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 they are 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 missed skills were offered to any other higher-level coalition. If this happens a renegotiation process must be started with the higher-level one, which should then verify the impact and if necessary renego-

tiate with its own higher-level coalition(s). This process is expanded through the whole levels until reaching the upper one. As a conclusion it can be claimed that the removal (or addition) of a manufacturing component (MRA) (its skills) provokes the automatic updating of the higher-level skills that could be directly or indirectly dependent on the ones that were removed (added).
It is important to retain that the skills offered to the coalitions at a higher-level can be a subset of the skills possessed by the CA member agent.
The skills brought to a coalition j led by CAi are the union of the skills brought by all MRAs that belong to the coalition j plus all the skills offered by the various coalitions that might be participating in coalition j. This means that a complex skill can be dependent on another complex one. To understand the next steps of CoBASA operation the following definitions are necessary:

S_{CA_i}	The set of skills of CAi in coalition/consortium i
$S_{MRA_{i,j}}$	The set of skills of MRA i in coalition/consortium j
$S_{CAmembers_i}$	The set of skills brought to the coalition/consortium i, led by CAi, by its members
$S_{CAgenerated_i}$	The set of skills generated by CA i in coalition/consortium i
$S_{CAoffered_{i,j}}$	The set of skills the coalition/consortium i, led by CA i, offers to the coalition/consortium j

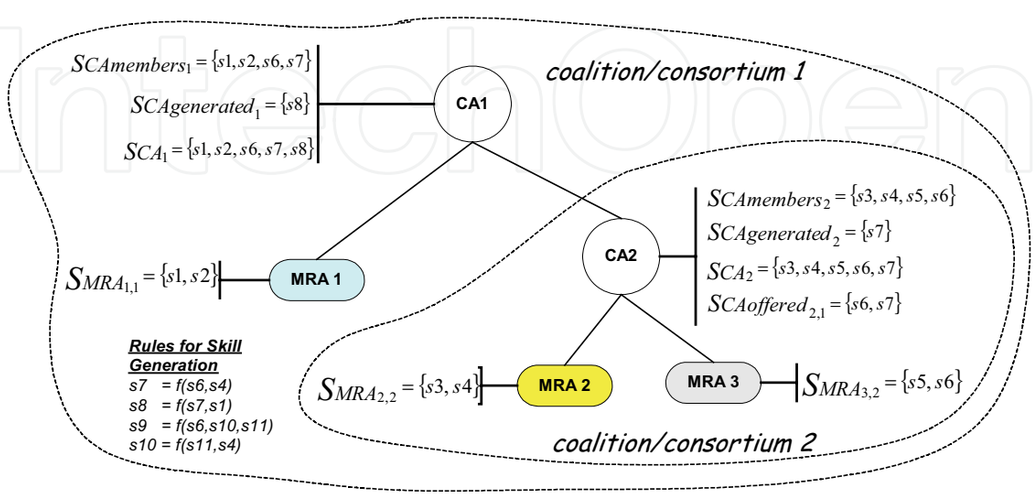

Figure 6. Coalition in its initial situation

Figure 7 shows that the skills offered by the coalition 2 are a subset of the skills the coalition possesses, which is perfectly valid. The skills to be offered are chosen during the coalition creation by the broker. The generation of skills is based on a set of rules that belong to the CoBASA knowledge base. For instance in coalition/consortium 1, according to the rules illustrated in Figure 3 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.

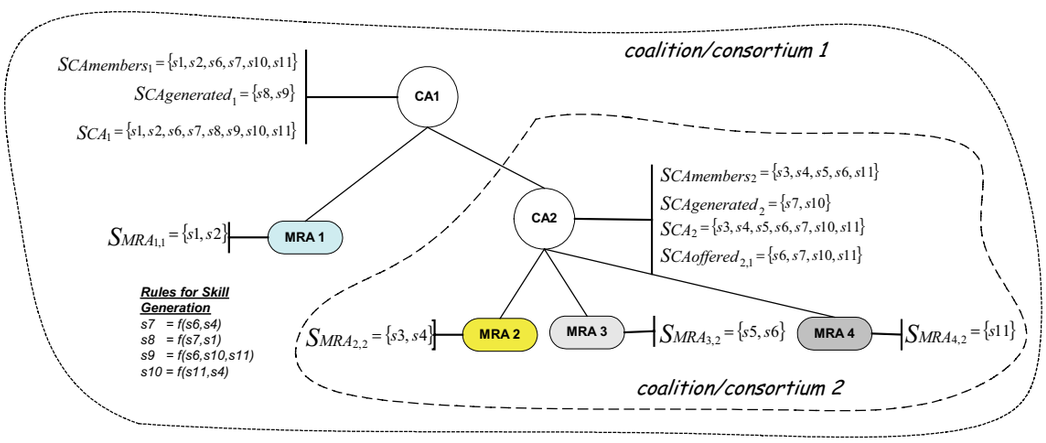

Figure 7. Hierarchy of coalitions after introducing a new element MRA 4

The effect of coalitions dynamics in CoBASA, can be verified by analysing what happens when a new component is added, for instance to coalition 2 (Figure 7). The introduction of MRA 4, which brings in new skill s11 causes an alteration on the set of skills CA2 can handle. It can be seen that the set of skills for the coalition 1 were increased. The update is almost automatic because it has only to do with the generation of complex skills and renegotiation between coalition leaders.

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 8).

From this discussion it is now possible to better understand why the CoBASA architecture can be considered a complex adaptive system. In effect coalitions are just an expression of the interaction that occur among coalition/consortium members. The skills owned by the coalition/consortium leader represent the

behaviour that results from its members' interactions. It can be identified a "movement" of low level skills to higher level ones, which allow us to claim that this architecture displays a kind of emergent behaviour (Johnson 2001).
A coalition member must execute all the operations promised by it in the consortium contract, when requested by the coalition coordinator. On the other hand, the coordinator (CA) can create complex operations (services) by aggregation of the individual operations of the members.
Let us now have a first look at the contracts that regulate the behaviour of coalitions and their members.

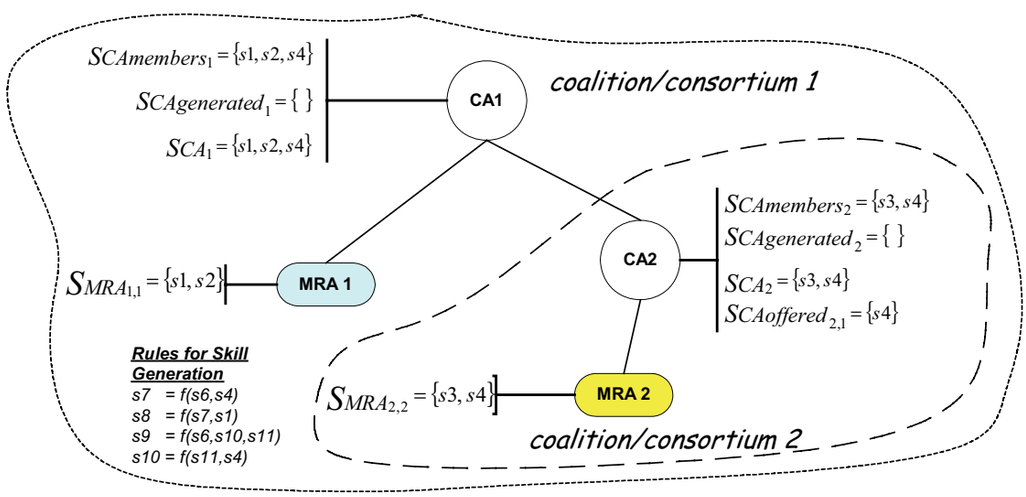

Figure 8. Hierarchy of coalitions after removing MRA 3

Figure 9 shows a hierarchy of two coalitions/consortia 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 consortium/coalition. However, each member of a consortium/coalition must have a copy of the contract that regulates the coalition's operation, since the members' behaviour is regulated by that contract. This means that in the case of figure 6 CA2 behaviour 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 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.
Skills descriptions help the creation of manufacturing coalitions. However this is not their only role, since they are also very important when the coalition is being operated (operational phase). This is so because skills represent also the commands to be used among coalitions/MRAs (services). The important question here is how the CA reacts when it receives a request to perform a certain task according to the skills it offered.

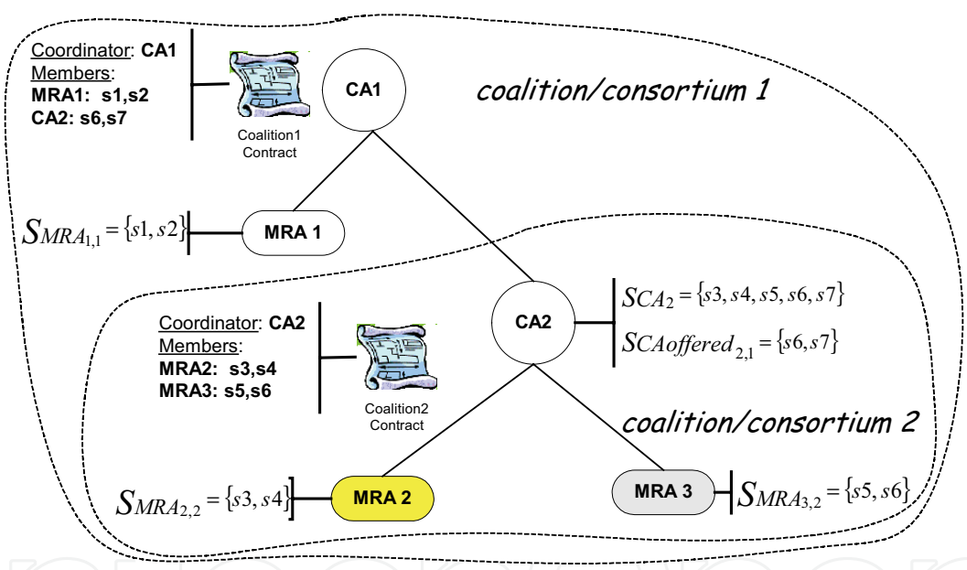

Figure 9. Coalitions contracts

When the CA is requested to perform some task associated to one of its skills, it behaves differently according to the skill type. If the skill was not generated by this CA (simple skill) the action consists simply in redirecting the request to the member of the coalition that has brought it. On the other hand, if the skill is generated by this CA then the procedure is more complex. This is so because the skill is now a composition of the skills brought to the coalition by its members, and this composition can be complex. This means that a model is needed to describe this composition and it should allow the modelling of complex command structures, which are needed to represent those skills that have complex structures. The CA must then execute the model by sending lower

level commands (skills) according to the model structure of the complex skill being executed. This is to conclude that a model is required to represent the structure of the composed skill and then an execution machine is needed as part of the CA to execute the model properly.
If each CA embeds a generic execution machine, like a Petri Net (Zurawski and Zhou 1994) executor, or even a workflow engine (WFMC 2002), able to execute Petri Nets or Workflow models than the CA is transformed into a kind of generic machine that can work with different types of skills.

3.3 Contracts

According to the law of contracts (Almeida 2000; McKendrick 2000), 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 2000) 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 constrains 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 as introduced in previous section: generic **multilateral contracts** and **adhesion contracts**. The main difference between them is the process of formation, which in the case of the adhesion contracts is via standardised forms. The contract offered by the cluster manager agent to the candidate member agents is a typical contract of adhesion, in the sense that the cluster imposes its terms. The only thing an agent can do is accepting or refusing it. Part of the terms of this adhesion contract, namely the "consideration" of the candidate agent, is left open to be filled in by the candidate, when accepting the offer. In terms of the human law systems **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". In most of the law systems in order to **create a contract** at least two sequential statements are required: 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 offer and the acceptance might not be the first and second action but they will be surely the last but one, and the last. Offers may set certain conditions on acceptance and to these, the acceptor is bound. The

acceptance validates and gives life to the contract. The contract starts at the moment the acceptance reaches the offeror.

The cluster manager, and the candidate agents when negotiating the cluster contract will use the offeror-acceptance protocol of real life contracts with some adaptations.

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). In the approach being followed an offer is made without specifying a deadline. 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 to reach an agreement other than the offer-acceptance protocol is using joint contractual terms, which express the agreements of the parts in only one text. This modality is specially 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 kind of communicative acts in a preliminary phase. Afterwards, the final contract is put on a written form (final agreement) and finally all the partners must subscribe the contract. The contract turns effective when the last partner subscribes the document.

The formation of the coalition contract used in the proposed architecture uses this modality with some adaptations. The human user interacting with the broker will prepare the agreement on the terms of the contract (preliminary phase). It is this user that chooses the skills that each agent will bring to the contract (this user is just configuring the system). The broker agent then sends the final *text* to all partners to be subscribed. When the last agent finally subscribes it, the contract is considered as valid.

3.3.1 Cluster Adhesion Contract - CAC

The cluster adhesion contract is defined externally to the cluster and modelled using a knowledge representation system – Protégé 2000 (Protégé-2000 2000). The cluster manager agent can interact with this system to have access to the contract representation. Whenever it needs to offer an adhesion contract to an agent it just uses the form, waiting afterwards for its acceptance or refusal.

The formation of the contract starts when the cluster manager sends a message to the candidate agent containing an instance of an adhesion contract. The "accept" message from the candidate contains the complete adhesion contract, now filled in with the terms of the candidate (its skills), and when received by the cluster manager the contract turns to be valid. The cluster manager only agrees to negotiate with the candidate agent if it is not on the black list of the cluster. The cluster manager agent then checks for the credits of the candidate, which represents a kind of curriculum vitae. A credit is, for instance, the number of hours working properly, or a number that qualifies the global performance of the agent when working on consortia. Those agents with lower level qualification can sometimes not be accepted as members of the cluster. This is to guarantee that consortia created out of a cluster have a certain level of qualification (Barata and Camarinha-Matos 2002). When the candidate (MRA/CA) does not have sufficient credits, the cluster manager replies with a FAILURE command message (left part of Figure 13). If the credits are accepted, the cluster manager fills in all the cluster adhesion contract (CAC) terms except the skills that will be brought in by the candidate, which should be filled in by the candidate. Then the cluster manager sends a REQUEST message to the candidate asking it to accept the contract. This corresponds to an offer in contract law 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. Then a FAILURE message is sent to the candidate stating that the cluster manager did not accept its REQUEST to join the cluster. If, on the other hand the MRA/CA, after evaluating the offer decides for its acceptance, sends an INFORM message stating its acceptance. The cluster manager sends then a final INFORM message to the candidate stating that its initial REQUEST has been accepted (right part of Figure 13).

The commands exchanged between the candidate and the cluster manager follows the FIPA protocols (FIPA 2002).

There is a tight connection between the CAC and credits (agent's curriculum). If credits are regarded as a kind of performance measure it is quite natural that at the end of a contract credits must be 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 it means the contractee agent (MRA/CA) has accomplished all its promises. Therefore it is natural that this agent could add some good points to its curriculum. On the other hand, if an abnormal termi-

nation is considered, it is normal that a kind of curriculum 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 are the penalisations or rewards they get when the contract is terminated.

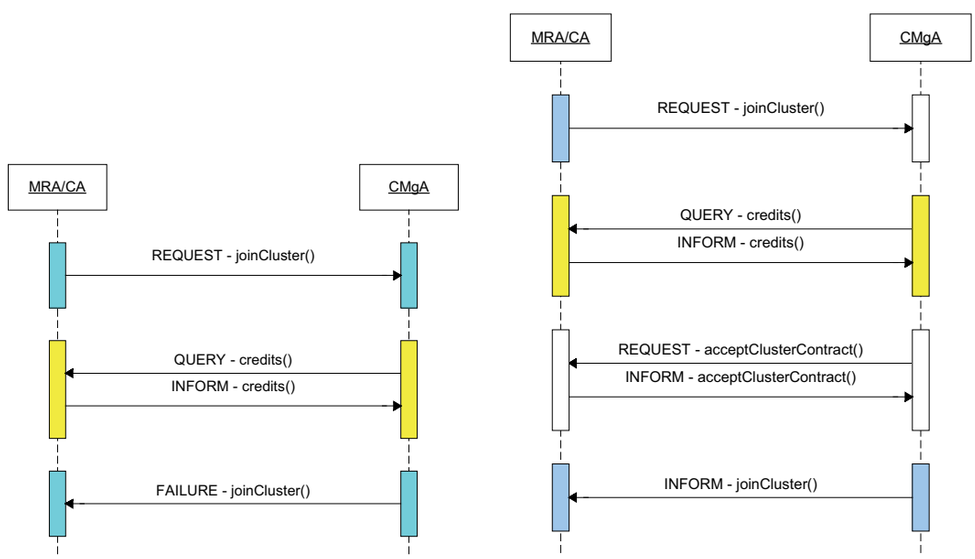

Figure 10. Unsuccessful and successful cluster joining

3.3.2 Coalition Contract - MCC

The broker agent, with the help of a human expert, creates the coalition contract (MCC). The model of this type of contract has many similarities with the previous one but has also some slight differences because it is a multilateral contract instead of a bilateral contract. To support various members and one contractor the contract has one common part dedicated to the contractor (the agent playing the co-ordination role), and another part dedicated to each of the other members. The *members* part of the contract is composed of several *individualConsortia* elements that in turn describe the individual contractual terms of each member of the coalition. The **promise** (declaration or manifestation of an intention in a contract) brought to the contract by each member is a set of manufacturing skills.

The broker creates the contract when a coalition is created. The user configures the different parts of the contract based on the requirements needed by the coalition. For each member the individual part is fulfilled namely by choosing which skills the members bring to the coalition.

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 asked.

At the end of the contract the CA awards each coalition member with a number that represents the quality of the handed out service. This award or penalisation, if added to the agent credits, can be used to improve (or even reduce) its qualification, and is important for the future participation of the agent on consortia. This mechanism is similar to the one mentioned when CACs have been discussed. Similarly there are three different ways of terminating a MCC: by performance, by frustration, and by breach.

The "good" way of terminating a contract is by performance. In this situation the CA (coordinator) verifies if the participation of any member is within the valid date. If not, the CA asks that member to terminate its participation. Based on the value stored 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 an abnormal way, and consequently the breaking agent may incur in some penalisations. The request to break the contract by frustration is always initialised by the coalition member that detected the frustration. When this happens the member collects the penalisation stored in the contract. Three reasons can lead a coalition member to request to terminate 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.

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 of the contract 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 in penalisations. However when it is "guilty", i.e., the coordinator detected some misbehaviour, it gets penalised. A member shows bad behaviour whenever it does not answers a request from its coordinator to execute one of the promised skills. Likewise 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. 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.

4. CoBASA main interactions

The most important functionalities related to CoBASA coalitions are:

1. Creating new coalitions
2. Changing coalitions
3. Coalition dissolution
4. Service execution

4.1 Creating new coalitions

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.

Fifure 11 shows the interactions that happen between the different actors involved in creating a coalition.

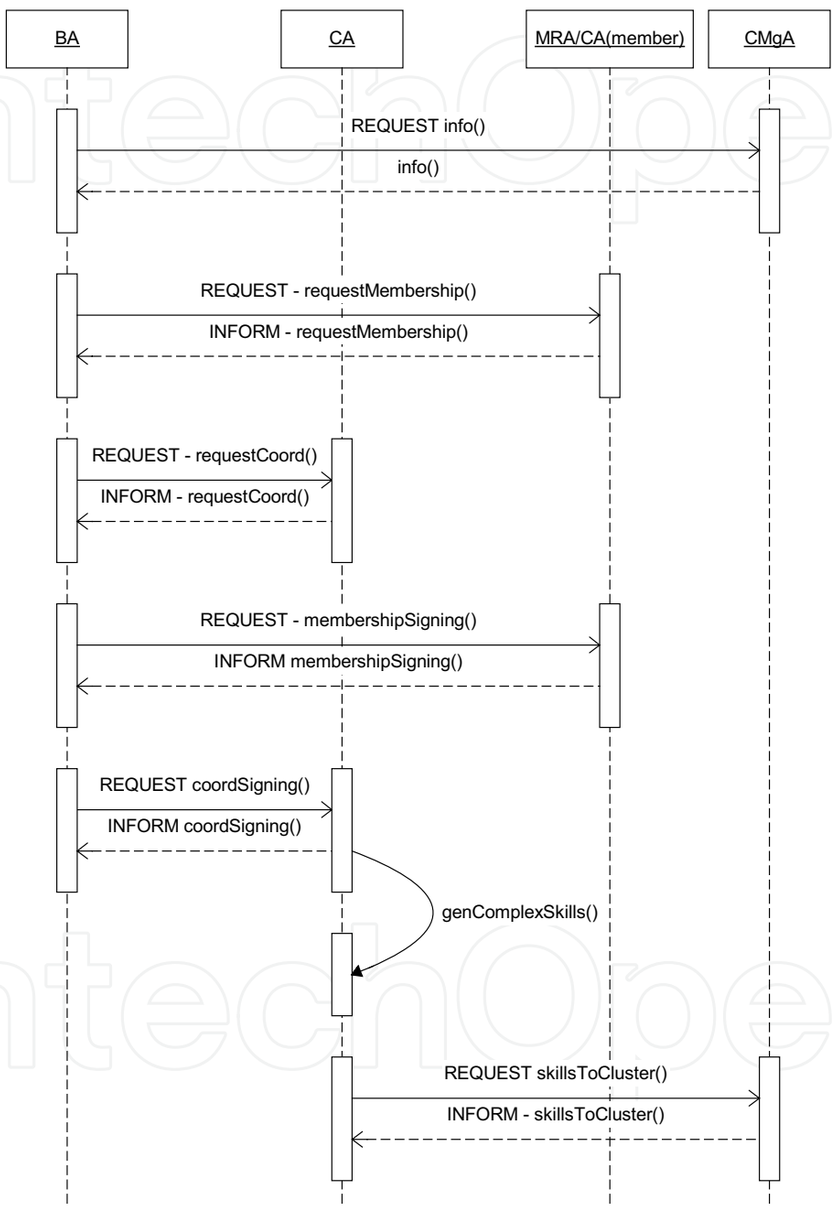

Figure 11. Interactions when creating a coalition

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 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 <u>new member is added</u> to an existing coalition are shown in Figure 15. 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 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.

4.2 Changing coalitions

Changing a coalition corresponds to changing the way the manufacturing (Figure 15). 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*).

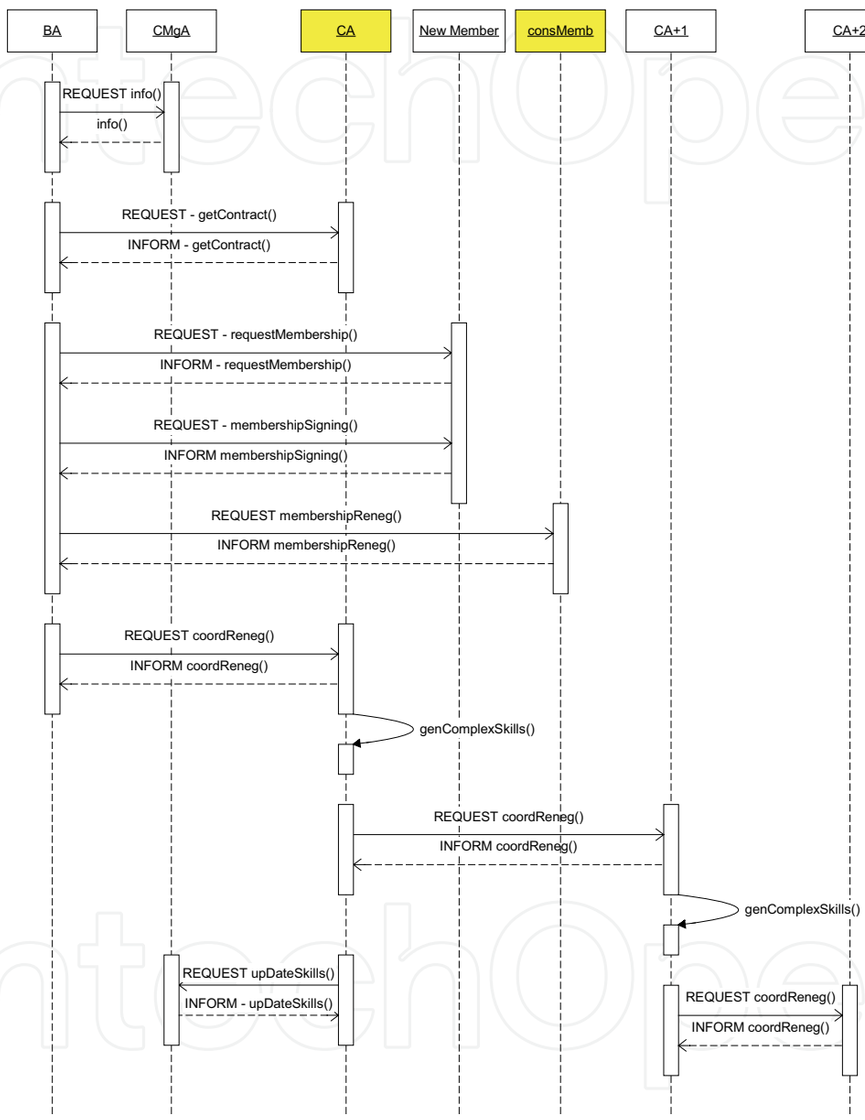

Figure 12. Adding an element to an existing coalition

Figure 12 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).
The removal of one element is not shown because it follows a similar negotiation pattern.

4.3 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 17 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*).

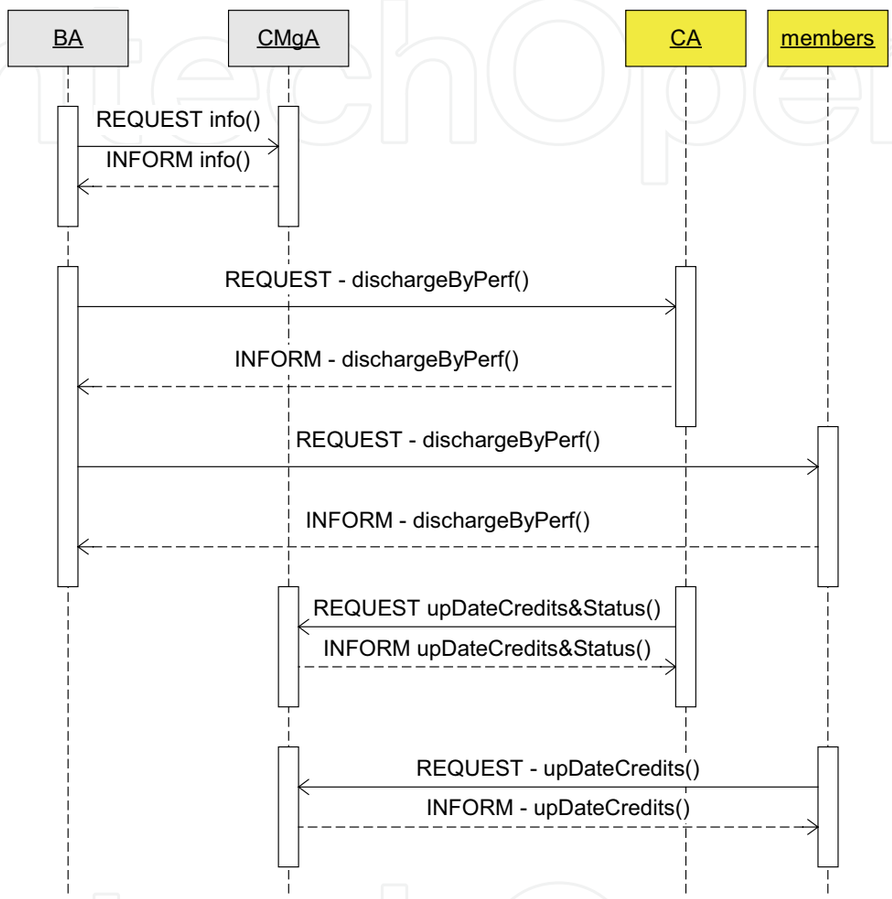

Figure 13. Coalition dissolution

4.4 Service execution

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.

Figure 14shows 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.

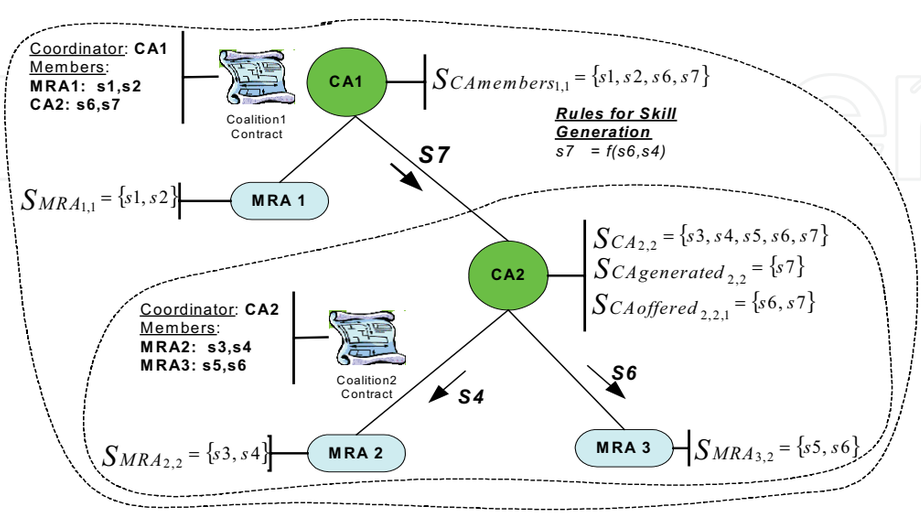

Figure 14. Skills requests in a hierarchy of coalitions

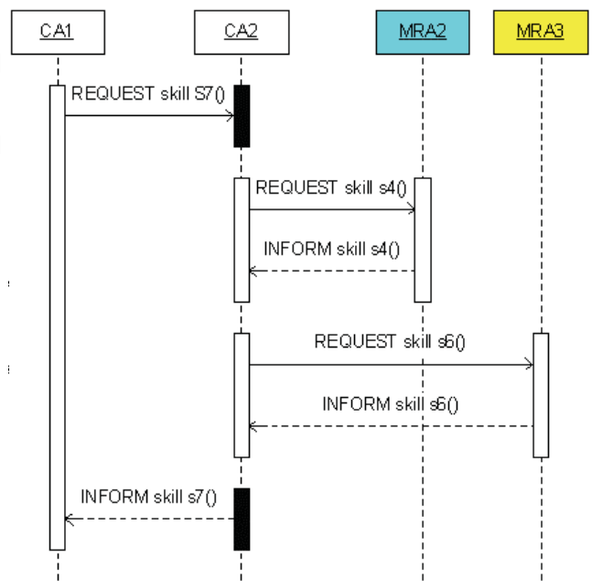

Figure 14. (Continued.) Skills requests in a hierarchy of coalitions

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.

5. Practical Implementation

The CoBASA architecture was validated in the NovaFlex pilot assembly cell (Figure 15), which is composed of two robot cells, one automatic warehouse and various conveyors connecting the two robot cells.

Figure 15. Célula NovaFlex

5.1 Development platform and CoBASA prototype

The JADE – Java Agent Development framework (Bellifemine et al. 2001; JADE 2001) was chosen for the experimental work mainly because it is an open source FIPA compliant platform, provides good documentation and support, and it is also recommended by the experience of other research groups with whom the authors have close relationship. Its use of *Behaviours*, and the easy connection to JESS rule processing engine (Jess 2000) helps in reducing the programming effort. Moreover JADE, implements the FIPA-ACL agent communication language. Another interesting feature of JADE is the functionalities

provided to manage the community of agents. It includes a *Remote Monitoring Agent* (RMA) tool, which is used to control the life cycle of the agent platform, and an agent for *white pages* and life cycle services (Agent Management Service - AMS).

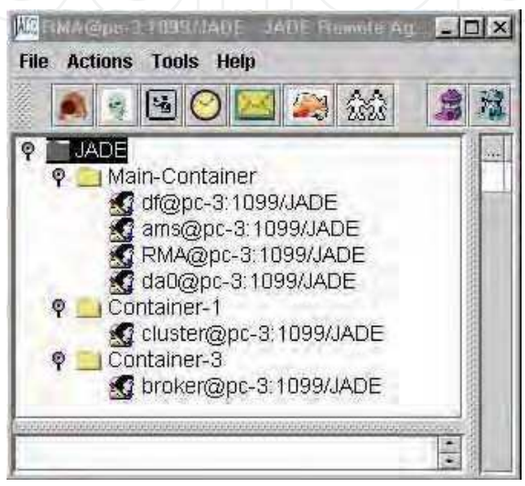

Figure 16. JADE Monitoring tool and messages between the Cluster and the Generic Agent

In Figure 16 (left hand side) the JADE monitoring tool shows the three example agents of the architecture. The agent address is da0@pc-3:1099/JADE. Although all agents were running in the same platform pc-3, this is not at all mandatory. The right hand side of Figure 16 shows the sequence of messages

between the cluster manager (CMgA) and a CA/MRA. This specific case shows the registering sequence in the cluster of two MRAs.

Figure 17 shows the main user interface of the agent (CA/MRA) (left part). The right part shows the window that is opened when the user clicks the *cluster* button. In this window the user verifies the cluster adhesion contract (Figure 18), asks the cluster manager to update the agent's credits and skills, and can terminate the agent's participation in the cluster (*dischargeByFrustration* button).

The agent's interface lets the user access other windows related to its participation in coalitions as well as its execution phase.

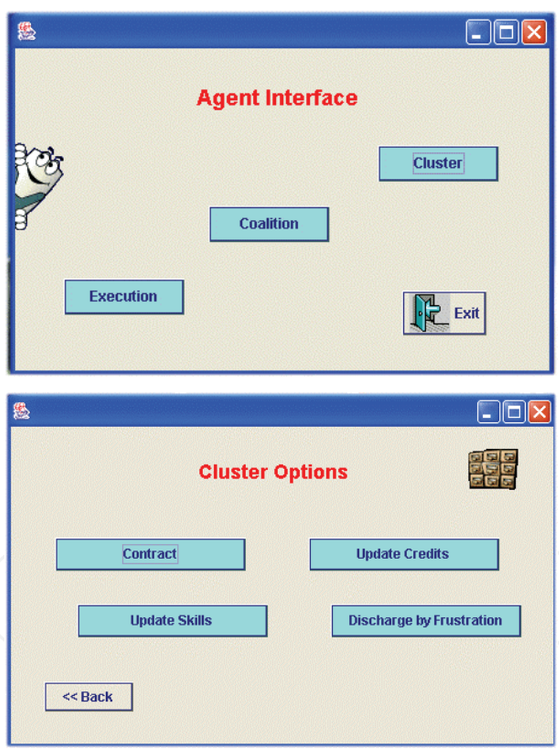

Figure 17. Agent interface and cluster options window

Figure 19 is the basic GUI of the broker. When the user chooses a candidate by selecting it (left column of available members), the broker asks the cluster manager for information about the selected agent. The figure shows that the cluster has five types of manufacturing components: robots, grippers, feeders, fixers, and coordinators (the tabs). When the user clicks on the "tabs" (options)

the members of that type existing in the cluster appear, and when the name is clicked the skills appear in the small window. The right part of the window shows the agents that have been chosen. In this case agents of type robot, feeder, gripper, and a CA, were chosen. When the user clicks on one type, the specific agent names appear in the middle column. In addition if the names in the middle column are selected the skills that were chosen to be brought in to the coalition are shown.

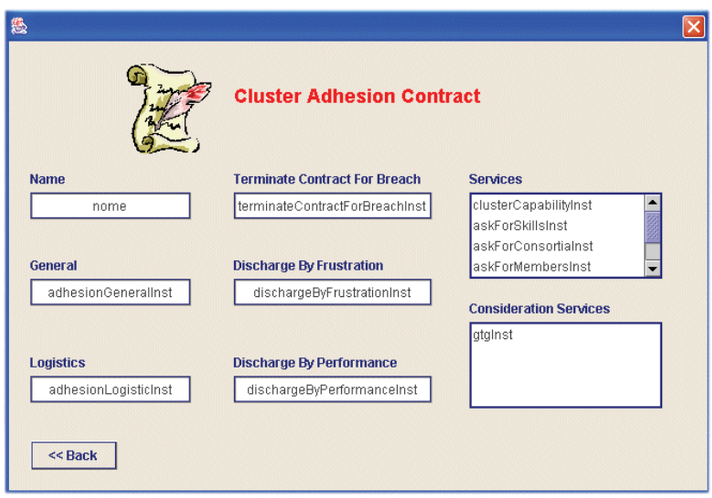

Figure 18. Cluster adhesion contract window

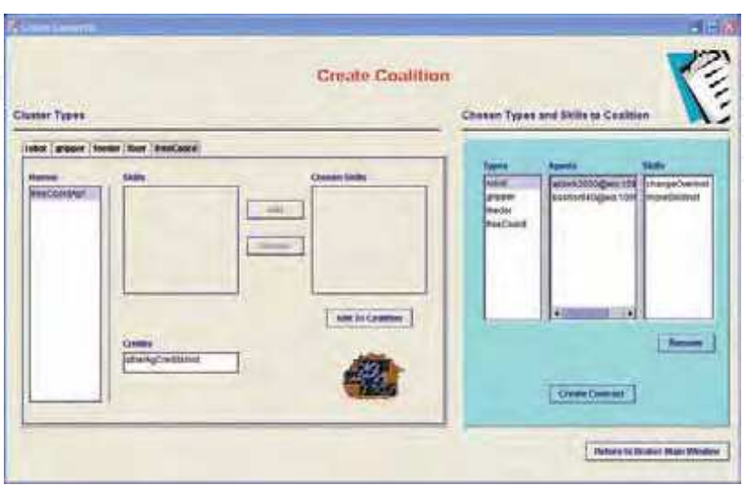

Figure 19. Create coalition/consortium in the broker

5.2 Agentification

Connecting the physical controller to the AMI could be an easy task if every physical component was controlled directly by its own agent. However, outdated legacy controllers with closed architectures control most of existing physical components. 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.

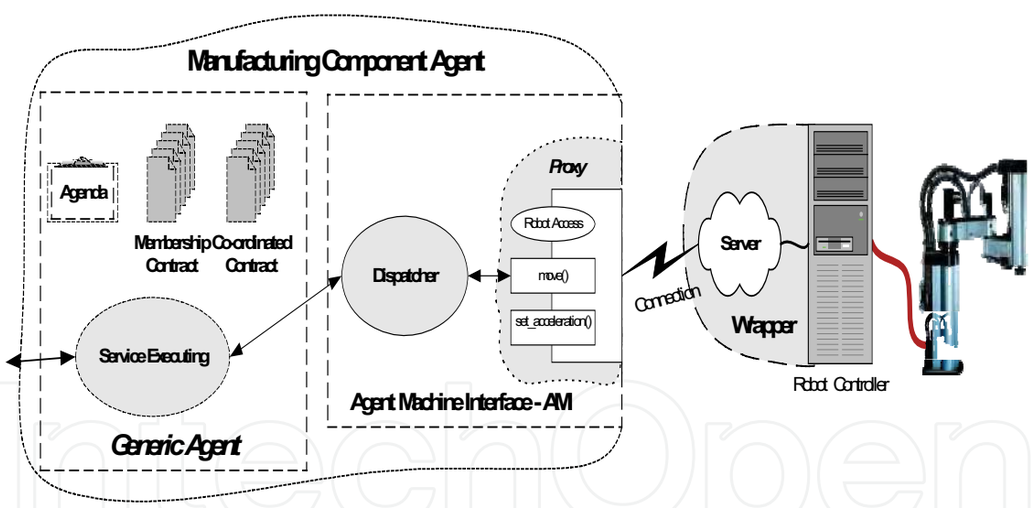

Figure 20. Physical component integration

Figure 20 shows a high level representation of an operative agent indicating how the wrapper integrates a manufacturing component (robot).

In previous works, the wrapper used to integrate physical components during the agentification process has been successfully implemented using two-tier client-server architecture (Barata et al. 1996; Camarinha-Matos et al. 1997; Camarinha-Matos et al. 1996). Recently, the wrappers for our NovaFlex manufacturing system, which is described in (Barata and Camarinha-Matos

1994), were developed using DCOM.

The Agent Machine Interface implementation is generic, i.e. an AMI can be connected to different distributed components (proxy) just by configuring what are the services of that proxy and the name/address of the component.

The generic agent tied to the AMI behaves in a slightly different way from other agents, at the initialisation phase. In this situation the GA reads from a contract representation file an instance of a consortium contract between itself and the AMI, and establishes a coalition. The member promise part (AMI) of the contract contains all 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 to create (generate) any type of manufacturing agent just by configuring an AMI and the consortium contract between the agent and the AMI. The only part of the system that is dependent of the physical component is of course the wrapper.

6. Conclusions

The CoBASA system offers an approach to introduce agility at the shop floor control level. The prototype proved the feasibility of the proposed approach, which seems able to provide a solution to rapid reengineering of shop-floor systems. Based on the concept of generic agent and its various behaviours regulated by contracts, it is possible to change the behaviour of a complex shop floor through the definition of new contracts (configuration) without the need to reprogram the control system. Current developments are devoted to assess the level of agility of the solution and to partially automate the brokerage activities.

7. References

Almeida, C. F. (2000). *Contratos I - Conceitos; Fontes; Formação*, Almedina, Coimbra.

Barata, J., and Camarinha-Matos, L. M. (1994). "Development of a FMS/FAS System." *Studies in Informatics and Control*, 3(2-3), 231-239.

Barata, J., and Camarinha-Matos, L. M. (2000). "Shopfloor Reengineering To Support Agility in Virtual Enterprise Environments." E-Business and Virtual Enterprises, L. M. Camarinha-Matos, H. Afsarmanesh, and R. Rabelo, eds., Kluwer Academic Publishers, London, 287-291.

Barata, J., and Camarinha-Matos, L. M. (2002). "Contract Management in Agile Manufacturing Systems." Collaborative Business Ecosystems and Virtual Enterprises, L. M. Camarinha-Matos, ed., Kluwer Academic Publishers, New York, 109-122.

Barata, J., Vieira, W., and Camarinha-Matos, L. M. "Integration and MultiAgent Supervision of Flexible Manufacturing Systems." *Mechatronics'96 - The 5th UK Mechatronics Forum International Conference*, Guimarães - Portugal, 185-190.

Bellifemine, F., Poggi, A., and Rimassa, G. (2001). "Developing Multi-Agent Systems with a FIPA-Compliant Agent Framework." *Software-Practice & Experience*, 31(2), 103-128.

Camarinha-Matos, L. M., Barata, J., and Flores, L. (1997). "Shopfloor Integration and MultiAgent Supervision." I. Rudas, ed., 457-462.

Camarinha-Matos, L. M., Seabra Lopes, L., and 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.

Conte, R., and Dellarocas, C. (2001). "Social Order in Multiagent Systems." Multiagent systems, artificial societies, and simulated organizations, Kluwer Academic Publishers, Boston, ix, 239.

Ferber, J. (1999). *Multi-Agent Systems : an Introduction to Distributed Artificial Intelligence*, Addison-Wesley, Harlow.

FIPA. (2001). "FIPA Request Interaction Protocol Specification." *XC00026F*, FIPA - Foundation for Intelligent Physical Agents, Geneve.

FIPA. (2002). "The Foundation for Intelligent Physical Agents."

Franklin, S., and Graesser, A. (1997). "Is it an Agent or Just a Program? A Taxonomy for Autonomous Agents." Intelligent Agents III - Agent Theories, Architectures, and Languages, J. P. Muller, M. Wooldridge, and N. R. Jennings, eds., Springer-Verlag, Berlin, 21-35.

Giampapa, J. A., Paolucci, M., and Sycara, K. "Agent Interoperation Across Multagent System Boundaries." *Fourth International Conference on Autonomous Agents (Agents 2000)*, Barcelona - Spain.

Goldman, S. L., Nagel, R. N., and Preiss, K. (1995). *Agile competitors and virtual organizations: strategies for enriching the customer*, Van Nostrand Reinhold, New York.

Gullander, P. (1999). "On Reference Architectures for Development of Flexible Cell Control Systems," PhD Thesis, Gotenborg University, Sweden.

Huff, B. L., and Edwards, C. R. (1999). "Layered Supervisory Control Architecture for Reconfigurable Automation." *Production Planning & Control*, 10(7), 659-670.

JADE. (2001). "http://sharon.cselt.it/projects/jade/."

Jess. (2000). "http://herzberg.ca.sandia.gov/jess/."

Johnson, S. (2001). *Emergence*, Penguin group, London.

Klusch, M., and Sycara, K. (2001). "Brokering and Matchmaking for Coordination of Agent Societies: A Survey." Coordination of Internet Agents: Models, Technologies, and Applications, A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf, eds., Springer-Verlag, Berlin, xxvii, 523.

Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritchow, G., Ulsoy, A. G., and Van Brussel, H. (1999). "Reconfigurable Manufacturing Systems." *CIRP Annals*, 48(2).

McKendrick, E. (2000). *Contract Law*, PALGRAVE, New York.

Mehrabi, M. G., Ulsoy, A. G., and Koren, Y. (2000). "Reconfigurable Manufacturing Systems: Key to Future Manufacturing." *Journal of Intelligent Manufacturing*, 11, 403-419.

Onori, M. (1996). "The Robot Motion Module: A Task-Oriented Robot Programming System for FAA Cells," PhD thesis, The Royal Institute of Technology, Stockholm.

Payne, T., Singh, R., and Sycara, K. "Facilitating Message Exchange through Middle Agents." *The First International Joint Conference on Autonomous Agents and Multi-Agent Systems*.

Protégé-2000. (2000). "http://protege.stanford.edu."

Sycara, K., Decker, K., and Williamson, M. "Middle-Agents for the Internet." *IJCAI-97 International Conference on Artificial Intelligence*, Nagoya - Japan.

Vos, J. A. W. M. (2001). "Module and System Design in Flexible Automated Assembly," PhD Thesis, Delft University Press, Delft.

Weiss, G. (1999). "Multiagent Systems : a modern approach to distributed artificial intelligence." MIT Press, Cambridge, Masschusetts, xxiii, 619.

WFMC. (2002). "Workflow Management Coalition."

Wiederhold, G. (1992). "Mediators in the Architecture of Future Information Systems." *IEEE Computer Systems*, 25(3), 38-49.

Wong, H. C., and Sycara, K. "A Taxonomy of Middle-Agents for the Internet." *Fourth International Conference on MultiAgent Systems*, 465-466.

Wooldridge, M., and Jennings, N. R. (1995). "Intelligent Agents - Theory and Practice." *Knowledge Engineering Review*, 10(2), 115-152.

Wooldridge, M. J. (2000). *Reasoning about Rational Agents*, MIT Press, Cambridge, Massachusetts; London.

Wooldridge, M. J. (2002). *An Introduction to Multiagent Systems*, J. Wiley, New York.

Zurawski, R., and 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," PhD Thesis, Eindhoven Technical University, Eindhoven - The Netherlands.

Manufacturing the Future

Edited by Vedran Kordic, Aleksandar Lazinica and Munir Merdan

ISBN 3-86611-198-3

Hard cover, 908 pages

Publisher Pro Literatur Verlag, Germany / ARS, Austria

Published online 01, July, 2006

Published in print edition July, 2006

The primary goal of this book is to cover the state-of-the-art development and future directions in modern manufacturing systems. This interdisciplinary and comprehensive volume, consisting of 30 chapters, covers a survey of trends in distributed manufacturing, modern manufacturing equipment, product design process, rapid prototyping, quality assurance, from technological and organisational point of view and aspects of supply chain management.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Jose Barata (2006). The Cobasa Architecture as an Answer to Shop Floor Agility, Manufacturing the Future, Vedran Kordic, Aleksandar Lazinica and Munir Merdan (Ed.), ISBN: 3-86611-198-3, InTech, Available from: http://www.intechopen.com/books/manufacturing_the_future/the_cobasa_architecture_as_an_answer_to_shop_floor_agility

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821