Coalitions of manufacturing components for shop floor agility - The CoBaSA architecture

José Barata and Luis M. Camarinha-Matos
New University of Lisbon and Uninova
Quinta da Torre – 2829-516 Caparica, PORTUGAL
TF: +351212948535 E-mail: jab@uninova.pt, cam@uninova.pt

Abstract: In this article an agent-based architecture in which cooperation is regulated by contracts is proposed as a flexible approach to dynamic shop floor re-engineering. It describes the dynamic and flexible co-operation of manufacturing agents, representing manufacturing resources, and how they can be created from a generic agent template. Agents behaviour and contract types and structure are discussed. First experimental results are also introduced.

Keywords: Virtual organisation, contracts, agile manufacturing, multiagent systems.


Biographical notes: José Barata is a teaching assistant at the Electrical Engineering Department of the New University of Lisbon and a researcher at the UNINOVA institute. His current research interests are agile manufacturing, shopfloor integrating infrastructures, shop floor reengineering, and virtual organisations.

Luis M. Camarinha-Matos is associate professor at the Electrical Engineering Department of the New University of Lisbon where he leads the group of Robotics and Integrated Manufacturing. He is also a researcher in the UNINOVA institute where he leads the group of Robotics and CIM. His research interests include virtual organizations, remote supervision systems, multiagent systems in industry, systems integration and intelligent robotics.

1. Introduction

The capability to rapidly change the shop floor infrastructure is a condition to allow participation of manufacturing enterprises in dynamic cooperative networks. The evolution in the market conditions, the environment and working conditions regulations, the improved standards for quality, fast technological mutation, and changes of the production paradigm itself, require agility and new ways of cooperation from the manufacturing companies.
Networked enterprise associations, virtual enterprises, advanced supply chains, etc. are examples of cooperative structures created to cope with the mentioned aspects [1]. 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, while lean essentially means producing without waste [2]. On the other hand, agility corresponds to operating efficiently but in a competitive environment dominated by change and uncertainty [3]. 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. The processes of change (re-engineering/adaptation) have been addressed mostly at the level of business process re-engineering 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. In fact, despite the efforts put in the creation of agile organizational structures little attention has been devoted to the agility of the shop floor, even if many research works have been focused on flexible assembly/manufacturing systems [4-7]. 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 much manufacturing companies demand for agile shop-floors is the increasing number of shop floor alteration projects [8]. As long as people in the shop floor are faced with the need to often change (adapt) their production systems, the need to have tools and methods to cope with such challenge increases significantly.

A particularly critical element in a re-engineering process is the control system. Current control/supervision architectures 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, what increases the programming effort and the potential for side-effect errors. It is therefore urgent to develop approaches to eliminate or reduce these problems, making the process of change (re-engineering) faster and more flexible, focusing on configuration instead of codification.

The work described in this paper assumes that there is a similarity between this 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 organization 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. Section 2 describes the architecture of a system to support dynamic and flexible cooperation of manufacturing agents, and how contracts are used to regulate cooperation in that architecture. The system is called Coalition Based Approach for Shop floor Agility – CoBASA, which adopts a contract-based multi-agent architecture designed to support an agile shop floor evolution. It is a multi-agent system because its components are agents in the sense defined in the Distributed Artificial Intelligence (DAI) / Multi-Agent areas [9-14]. The proposed approach has some similarities with the MetaMorph architecture [15]. This approach is contract-based because the behaviour of the agents is determined by contract arrangements, which are detailed in section 3. Multi-agent communities, where manufacturing equipment is represented by software agents have been identified in a number of other works [16, 17], however the use of contracts to govern their behaviour is an
innovative approach. The coalition formation process is described in section 4. Finally section 5 introduces current developments and experimental results.

2. CoBASA system architecture and components

An agent-based architecture in which cooperation is regulated by contracts is proposed as the basis of a flexible approach to dynamic shop floor reengineering. The basic components of the proposed architecture are Manufacturing Components, Manufacturing Resource Agents, Coordinating Agents, Clusters, Coalitions/consortia, Broker, and Contracts.

2.1 Basic concepts

<table>
<thead>
<tr>
<th>Definition 1 - Manufacturing Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>A manufacturing component is a physical equipment that can perform a set of specific functions in the shop floor. It is able to execute one or more basic production actions, e.g. moving, transforming, fixing or grabbing.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Definition 2 - Manufacturing Resource Agents (MRA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>The MRA is an agentified manufacturing component, i.e. a manufacturing component extended with agent skills like negotiation, contracting, and servicing, able to participate in coalitions/consortia.</td>
</tr>
</tbody>
</table>

As it could be expected there are several types of MRAs, one for each manufacturing component type. Therefore it is expected to find robot MRAs, gripper MRAs, tool warehouse MRAs, etc. Each MRA is individualised by its basic skills and attributes. In the CoBASA society the basic members are not the manufacturing components but the MRAs. Each manufacturing component thus needs to be agentified (transformed into an MRA) before it can participate in the CoBASA society [8, 18]. Since skills represent a very important characteristic of a manufacturing component in the CoBASA context, and since these skills are implemented by manufacturing controllers, a MRA is effectively the agentification of a manufacturing controller.

<table>
<thead>
<tr>
<th>Definition 3 – Coalition/Consortium</th>
</tr>
</thead>
<tbody>
<tr>
<td>A coalition/consortium is an aggregated group of agentified manufacturing components (MRAs), 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.</td>
</tr>
</tbody>
</table>

A coalition is usually regarded as an organisational structure that gathers groups of agents cooperating to satisfy a common goal [19]. Agents are often self-interested entities that see cooperation as a way to benefit their individual interests. On the other hand, the term consortium comes from the business area where it is defined as an association of companies for some definite purpose. Comparing both definitions it can be seen that they are quite similar because in both definitions there is the notion of a group of entities cooperating towards a common goal. This common definition is adapted to the CoBASA context. A basic coalition/consortium, besides being composed of MRAs, includes an agent that leads the coalition – Coordinating Agent (CA). In addition it can include as members other
coalitions/consortia. The coordinator of a consortium is able to execute complex operations that are composed of simpler operations offered by the consortium members.

**Definition 4 – Coordinating Agent (CA)**

A CA is a pure software agent (not directly connected to any manufacturing component) specialised in coordinating the activities of the coalition, i.e. that represents the coalition.

As members of coalitions/consortia, MRAs can only play the member role while CAs can play both the coordinator role and member role. A simple manufacturing coalition/consortium is composed of some MRAs and one CA. However, a coalition/consortium can be composed of other consortia creating in this way a hierarchy of coalitions/consortia. Therefore a CA can simultaneously coordinate MRAs and others CAs. In figure 1, for instance, 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`.

**Figure 1** Hierarchy of coalition/consortia

The better the dynamics of the consortia the better the agility of the manufacturing systems they represent. If agility is seen as the capability to easily change the behaviour of a manufacturing system as a reaction to a change in the environment, then an easy way to create and change consortia is an important support to give agility to a manufacturing system. Before addressing how coalitions/consortia are organised it is worthwhile to emphasise an intuitive aspect: the fact that the set of skills offered by a coalition is composed of not only the basic skills brought in by its members but also more high level skills that result from a composition of those simpler skills. Therefore, some kind of skill composition is needed to generate new skills.

When forming a coalition/consortium there are no limitations on the type of agents that can be involved in but there is an important restriction on MRAs, which limits their cooperation capability – their spatial relationship. Manufacturing agents that are not spatially related cannot cooperate, as it is the case, for instance of a robot and a tool. If the tool is not within the reachability space of the robot it will be impossible to create a cooperation 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 consortium it is mandatory to know what are the available, and “willing” to participate, agents. It would be important that these agents could be grouped by their spatial relationships (or any other relevant relationship e.g. technological compatibility), i.e. manufacturing agents that could establish consortia should be grouped together because they share something when they are candidates to consortia. This is
analogue to the long term collaborative alliances of enterprises called industry clusters [20]. An industry cluster is a group of enterprises and other organisations, in many cases located in the same geographical region, that are willing to cooperate and establish a long term cooperation agreement and proper common business practices/rules and support infrastructures. The objective of these clusters is to facilitate the creation of temporary consortia to respond to business opportunities. Similarly, in the case of CoBASA there is a need for a structure (cluster) that groups the agentified manufacturing components (MRAs) willing/able to cooperate.

**Definition 5 – Shop Floor Cluster**

A shop floor cluster is a group of agentified manufacturing components (MRAs) committed to participate in coalitions/consortia and sharing some relationships, like belonging to the same manufacturing structure and possessing technological compatibility.

A shop floor cluster includes a kind of directory where the agents willing to participate in coalitions/consortia can register. This directory acts as the place where those agents become known and where they publish their skills. A special agent – the cluster manager agent (CMgA) – is responsible for keeping the directory and supporting the adhesion/withdrawal of agents.

Every MRA should be able to: 1) adhere to a cluster, 2) participate in consortia/coalitions, and 3) perform the manufacturing operations associated to the skills it represents.

The cluster entity is also important as a facilitator of the adaptation process of its members to participation in coalitions/consortia. Namely by informing the candidates about some aspects that need to be followed and which will be important for the operation of future coalitions/consortia. 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/consortium. By doing this, the CMgA improves the process of creating coalitions/consortia.

To summarise 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/consortia.

These advantages contribute to a faster process when creating or changing coalitions/consortia. However, the formation of a coalition/consortium is not done by the cluster manager but rather by a specialised agent called broker.

**Definition 6 – Broker**

A broker is an agent that is responsible for the creation of coalitions/consortia in interaction with an external user. The broker agent gathers information from the cluster and, based on the user preferences, supervises/assists the process of creating the coalition/consortium.

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.

Figure 2 shows how manufacturing agents can be related to the cluster and coalition/consortium structures. Agentified components in the same “geographical” area of the shop floor and possessing technological compatibility may join the same cluster. From a
cluster it is possible to create coalitions/consortia that are nothing more than different forms of operating a cell.

2.2 Contracts

Contracts are the next important CoBASA mechanism, which is used to regulate the agent’s interaction with a cluster as well as its behaviour within coalitions/consortia.

**Figure 2** Consortia Formation

![Diagram showing consortia formation and different ways of operating](image)

**Definition 6 – Contract, according to the law [21]**

"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."

In the CoBASA architecture two type of contracts are considered: cluster adhesion contract (CAC), and multilateral consortium contract (MCC).
Definition 7 – 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 8 – 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 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 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 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.

2.3 Coalition Dynamics

As 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 be ever 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 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 lost skills were offered to any other higher-level coalition. If this happens a
The renegotiation process must be started with the higher-level one, which should then verify the impact and, if necessary, renegotiate with its own higher-level coalition(s). This process is expanded through the various 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 depending 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 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. To understand the next steps of CoBASA operation the following definitions are necessary:

- \( SCA_i \) - The set of skills of CAi in consortium i
- \( SMRA_{ij} \) - The set of skills of MRAi in consortium/consortium j
- \( SCA_{membersi} \) - The set of skills brought to the consortium/coalition i, led by CAi, by its members
- \( SCA_{generatedi} \) - The set of skills generated by CAi in consortium/consortium i
- \( SCA_{offeredi,j} \) - The set of skills the coalition/consortium i, led by CAi, offers to the coalition/consortium j

**Figure 3** Example of a coalition hierarchy in its initial situation

Figure 3 shows that the skills offered by the coalition/consortium 2 are a subset of the skills the coalition/consortium 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/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.

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 4). The introduction of MRA 4, which brings in the 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 is increased. The update is almost automatic because it has only to do with the generation of complex skills and renegotiation between coalition leaders.

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

Considering now the removal of a component (MRA 3, for instance), it causes a reduction of skills both in coalition/consortium 1 and coalition/consortium 2 (Figure 5).

Figure 5  Hierarchy of coalitions after removing MRA 3

From this discussion it is now possible to better understand why the CoBASA architecture can be considered a complex adaptive system. The skills owned by the coalition/consortium leader represent the behaviour that results from its members’ interactions. A “movement” of low level skills to higher level ones can be identified, which allows us to claim that this architecture displays a kind of emergent behaviour [22].

A coalition/consortium 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/consortia and their members.

Figure 6 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 the behaviour of CA2 is conditioned, in fact, by two contracts instead of one: 1) the contract of coalition/consortium 1, where CA2 is a member, and 2) the contract of coalition/consortium 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 classification, the structure of the contracts is the same, since both types are multilateral consortium contract - MCC.

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 very important when the coalition is being operated (operational phase). This is so because skills also represent 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 6  Coalitions/Consortia 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 an aggregated one, generated by this CA, then the procedure is more complex. This is so because the skill is now a composition of skills brought to the coalition by its members. A model is needed to describe this composition, allowing 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.
skill execution requests) according to the model structure of the complex skill being executed. 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 [23] executor, or even a workflow engine [24], 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.

3. Contracts

3.1 Background

According to the usual laws of contracts [25, 26], 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 [26] 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 types 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 actions 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 [26]. 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 consortium 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.

3.2 The Cluster Adhesion Contract

Figure 7  UML diagram of the Cluster Adhesion Contract (CAC)

The model of the cluster adhesion contract is depicted in figure 7, and it shows the different parts involved in the contract:

(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 (cluster) and from the offeree (agent) are described in this part. The agent consideration is the set of skills that it is interested in bringing to the cluster, and that might be used by future consortia. Although the consideration from the cluster side could be just to let the agent belonging to the cluster (implicit consideration) it was decided to define an explicit consideration – cluster skills are represented by some clustering functionalities.

(iii) Contract Logistics. An important parameter defined here is the clusterWorkingPlace that defines the physical area covered by the cluster. A candidate agent to 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 to be used 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 breached (interrupted) by three important reasons: 1) breached by frustration of the contractor (offeror), 2) breached by frustration of the contractee (offeree), and 3) breached by bad behaviour of the offeree. 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. In the contract modelled here the behaviours that might be used for each of the cases as well as some remedies that might be claimed by the offended partner are described.

It must be pointed out that the non-existence of a breach by bad behaviour of the offeror (the cluster) just occurs from the fact that the cluster is not expected to have bad behaviour, because when behaving improperly it is really impairing itself. On the other hand, from a law point of view the dominant partner can impose an adhesion contract where it cannot be liable for its bad behaviour.

The cluster adhesion contract is defined externally to the cluster and modelled using a knowledge representation system – Protégé 2000 [27]. 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.

**Figure 8** Unsuccessful and successful cluster joining

The establishment 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 sometimes cannot be accepted as members of the cluster. This is to guarantee that consortia created out of a cluster have a certain level of qualification [1]. When the candidate (MRA/CA) does not have sufficient credits, the cluster manager replies with a FAILURE command message (left part of figure 8). If the credits are accepted, the cluster manager fills in all the cluster adhesion contract (CAC) terms except the term offereeConsideration that refers to 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. 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. 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 8).

The commands exchanged between the candidate and the cluster manager follows the FIPA protocols [28].

There is a tight 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 it means 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, if an abnormal termination 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 because this is registered in the slots dischargeByPerf, DischargeByFrust, and termContForBreach of the contract. When the contract is terminated by performance, the contractee (MRA/CA) 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.

3.3 The Multilateral Coalition Contract

A model of the multilateral coalition/consortium contract (MCC) can be seen in figure 9. The broker agent, with the help of a human expert, creates this contract. It should be stressed that the human user currently conducts the negotiation process during a preliminary phase of this contracting process.

The model of this type of contract has many similarities with the previous one but 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 attribute is composed of several individualConsortia...
elements that in turn describe the individual contractual terms of each member of the consortium. The consortiumGeneral and consortiumLogistics parts are similar to the adhesion contract, but applied to the co-ordinator. The promise (declaration or manifestation of an intention in a contract) brought to the contract by each member is a set of manufacturing skills that are represented by the class manCompSkills.

Figure 9  UML diagram of the coalition/consortium contract

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.

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

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/consortium is operating. Only promised skills can be asked.

At the end of the contract the CA awards each coalition/consortium 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
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 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 dischargeByFrustration. Three reasons can lead a coalition/consortium member to request 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.

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. Therefore, a member starting a breach does not incur in penalisations. 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. 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.

4. CoBASA main interactions

The main global behaviours of CoBASA can be summarised as: 1) cluster registering, 2) creating a new coalition/consortium, 3) coalition/consortium changing, 4) coalition operation, 5) coalition dissolution, and 6) service execution. Cluster registering is the global behaviour of registering an agent in the cluster. This has been already discussed when the cluster contract was mentioned. The service execution behaviour is the action related to the execution of requests from higher levels (see hierarchy of coalitions) and how they can be decomposed into lower level requests. Coalition/consortium dissolution is similar to contract termination, which has been already discussed. For lack of space these two behaviours will not be described here. Coalition/consortium creation and changing involves the process of creating a new coalition/consortium or the changes that can be done to it, either by removing or adding members. These behaviours will be the focus of this section.

4.1 Coalition Creation

Creating a coalition/consortium corresponds either to create a new system, if this happens just after the first time the manufacturing components have been installed, or to change the system, if this is done in order to create a new logical structure of an already installed system which has previously dissolved all its coalitions/consortia.
The main actor in creating coalitions/consortia is the broker agent (BA). The other important actor is the cluster manager that provides information about available members. Furthermore other agents are needed to create a coalition:

1. A CA not currently engaged in any consortium (available to lead a coalition/consortium).
2. Various candidate members. These members are MRAs if the coalition contains only manufacturing components. If the coalition includes also other coalitions then some of the members are the CAs that represent those coalitions.

Figure 10 shows the interactions that happen among the different actors involved in creating a coalition/consortium. An AUML [29] sequence diagram is used to represent the agent interactions. AUML is an agent based unified modelling language similar to UML. Please note that the figure does not consider the interactions that occurred when the CA/MRA adhered to the cluster since these steps are not considered as part of the coalition creation phase. They belong instead to cluster adhesion. 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...
The broker asks for information about candidate members present in the cluster by sending a REQUEST command. After getting the information from the cluster manager (CMgA), the broker shows all the available members as well as their individual information and let the user compose the coalition and create the contract that regulates the coalition/consortium.

The broker then asks each member to verify if they accept the contract, which is done by sending a REQUEST to be member command. This step is to guarantee that each individual agent has the opportunity to detect those contractual conditions it cannot achieve, which the user interacting with the broker could not think of. An example of one such condition is timeToAnswer. This phase corresponds to ask the agent if it is interested in participating in the coalition under those conditions. The same is done with the coordinator.

After all members and the coordinator have replied they are interested in participating in the coalition, the broker starts the process of signing the contract, what in CoBASA is made by sending a REQUEST to sign command. It should be pointed out that there is no complex formalism in this process. The idea is more to indicate the coalition members that the contract is now effective. After requesting the coordinator to sign the contract the coalition is operating from the broker point of view.

However there are yet some other tasks before the creation is effectively finished. In effect the CA must now generate its new skills (genComplexSkills). This makes sense because the coordinator has just received a set of new skills, which it has got from its coalition members. It is worthwhile to remember that this step is crucial for the agility of the system, because the coalition is now generating automatically its skills based on the members that compose it.

Once 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 Coalition Changing

The process of changing a consortium is very similar to the creation of a new one. The only difference is found in the renegotiation behaviour that must run in the coordinator of the consortium that is going to be changed. Only after that the coordinator is ready to accept service requests. This renegotiation is important because in the case of a removal of an agent the consortium can be in a situation that it is no longer capable of honouring the promises made to the higher-level consortium it is a member of. To keep the system working properly it is necessary to update the promise on the higher-level consortium contract. A change can occur either when a new member is added or when a member is removed. Due to space restrictions only the situation when a new member is added to an existing coalition/consortium is considered.

Changing a coalition/consortium corresponds to changing the way the manufacturing components are organised, i.e. changing the system’s logical control structure. Therefore this phase is directly connected to the reengineering phase of the production system.

The interactions involved when a new member is added to an existing coalition/consortium are shown in figure 11. As in the previous case, the broker and the cluster manager (CMgA) agents are important players because it is through the broker that the coalition is altered while the CMgA provides the necessary information. Furthermore, the coalition leader and its members, the member to be added, and the coordinators of the coalitions, where hypothetically the coalition being changed is participating in, are the other actors.
Figure 11 Adding an element to an existing coalition

The process starts with the BA asking the CMgA to provide information about its members. It must be recalled that coalitions/consortia are also registered in the cluster. Hence, the user, via the broker, chooses which is the coalition/consortium to be changed. By doing this the BA asks then the coordinator of that coalition/consortium to send it its MCC (REQUEST \texttt{getContract}). The contract is requested since the user, besides configuring the individual part of the MCC with data from the new member, might also change other parts of the MCC. 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 needs now to renegotiate with the other coalition members the new terms of the contract to let these members having a word on them (REQUEST \texttt{membershipReneg}). The normal situation is acceptance of the changed contract by these agents. Please note that it is not shown what happens if one or more members refuse to participate. This is so to keep the
figure simple. 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. However, the CA has some other tasks to do before the whole process is concluded. First the CA 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/consortium as well as if it has got new skills. If yes in both cases it renegotiates with the leader (CA+1) of that coalition/consortium to change the skills it is bringing in (REQUEST coordReneg). In the particular case of an addition it is highly likely that new complex skills were generated besides the ones brought in by the new member. The CA just wants to change the skills, nothing else. Finally, after the successful renegotiation, the CA updates its skills. These skills correspond to the ones registered by this coalition in the cluster (REQUEST updateSkills).

Figure 11 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 other coalitions, 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 through all the coalitions that are directly and indirectly related with it.

5. Experimental Developments

5.1 Development platform and CoBASA prototype

The JADE – Java Agent Development framework [31, 32] 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. The use of Behaviours supplied by JADE, and the easy connection to JESS rule processing engine [33] helped 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).

In figure 12 (left 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 side of figure 12 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. The informative of the message can also be seen and this sequence can be compared with the protocol of figure 8.

Figure 13 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 contract (figure 14), asks the cluster manager to update the agent’s credits and skills, and can terminate the agent’s participation in the cluster (dischargeByFrustration button).
Figure 12 JADE Monitoring tool and messages between the Cluster and the Generic Agent

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

Figure 13 Agent interface and cluster options window

Figure 14 Cluster adhesion contract window

Figure 15 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 four 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 bring to the coalition are chosen.

Figure 15 Create coalition/consortium in the broker

5.3 Agentification

The agentification process corresponds to the transformation of a manufacturing component into a MRA. This process could be done by developing an agent that could both interact with the controller and participate in consortia, which is very demanding in terms of contractual negotiation and composition of skills. However, the requirements imposed by the interaction with the controller (short response time, interaction protocol specificities, ...) advise for the use of a dedicated agent to interact with the controller. Therefore the adopted approach is to separate the functionalities and to have one dedicated agent to interact with the controller – Agent Machine Interface (AMI) and a generic agent specialised on contractual negotiation and skills composition – Generic Agent (GA).

The Agent Machine Interface (AMI) is therefore the agent that will be 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. However, only one type of GA is required to build different MRAs types. The AMI is the agent wrapper of a manufacturing component that exports the functionalities existing in its physical controller.

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 16). The set of potential skills (operations) of a consortium led by a GA is composed of the basic operations brought in by consortium members plus the generated complex operations that can be composed using the basic ones. When the GA is leading a MRA the basic operations are those supported by the AMI. In this case the GA coordinates the AMI. When leading a complex consortium the basic operations come from the set of operations brought to the
consortium by all its members. In this case the high-level GA coordinates each of the other lower-level GAs that are leading other consortia.

**Figure 16** Hierarchy of Consortia

Connecting the physical controller to the AMI is an important process to be detailed. 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 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 17 shows a high level representation of an operative agent indicating how the wrapper integrates a manufacturing component (robot).

**Figure 17** Physical component integration

In previous works, the wrapper used to integrate physical components during the agentification process has been successfully implemented using two-tier client-server architecture [34-36]. Nevertheless, this architecture can have an important drawback, which is its limited flexibility in moving program functionality from one server to another [37]. Three-tier architectures have emerged as a solution to this limitation, namely three-tier with ORB (Object Request Broker) architecture. The current big players in this arena are the CORBA (Common Object Request Broker Architecture), DCOM (Distributed Component Object Model), and the JRMI (Java Remote Method Invocation). Recently the wrappers for our NovaFlex manufacturing system [38] were developed using DCOM because (1) all the computers available to control the NovaFlex are running Microsoft operating systems (Windows95, 98, and NT), (2) C++ Builder, the used development language, has good tools to develop DCOM applications, and (3) developers were better trained on the Microsoft environment.
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/consortium. 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

This paper introduced a new multi-agent-based control architecture founded on the concepts of cluster and consortium regulated by contracts. The advantages of the approach to support the shop floor life cycle can be summarised in the following:

1. **Flexibility/Agility** – Separating the aspects purely related to agent competence (skills) from the aspects related to co-operation (regulated by contracts) and co-ordination facilitates the development of much more agile structures. It is preferable to have an entity (agent) with generic capabilities (skills) and whose tasks are regulated by a contract than an entity with fixed behaviour (pre-programmed). Consortia and contracts allow creating different structures with different objectives, with the same agents.

2. **Adaptability** – Changes made to the capability of an agent or consortium may be propagated to the rest of the architecture almost automatically or via simple re-configuration of contract clauses.

3. **Openness/Scalability** – Adding new agents to the architecture becomes easier because there is no need for reprogramming the other components.

4. **Reconfigurability** – Physical changes are essentially made based on configurations done in clusters, consortia and/or contracts.

In spite of these advantages there remains a difficult task when adding a new resource to this environment – the agentification process that requires the development of a specific wrapper.

The main behaviours associated to the cluster, broker, and generic agent concepts were described and explained. The importance of the generic agent to create manufacturing agents as well as co-operating consortia was also stressed. A first implementation of contract models and the various types of agents shows promising results towards a highly re-configurable control infrastructure within a real complex manufacturing environment. Further work is necessary on cluster management services, dynamic consortium formation and modification, error recovery, and contract configuration / specification tools.

**Acknowledgements**

This work was supported in part by the European Commission through the Growth Assembly-Net project.
References