# Aspects of co-operation in Distributed Manufacturing Systems

**Paulo SOUSA, Nuno SILVA**
Instituto Superior de Engenharia do Instituto Politécnico do Porto, Portugal
Email: {psousa | nsilva}@dei.isep.ipp.pt

**Tapio HEIKKILA**
VTT automation, Oulu, Finland
Email: Tapio.Heikkila@vtt.fi

**Martin KOLLINGBAUM**
University of Cambridge, UK
Email: mjk27@eng.cam.ac.uk

**Paul VALCKENAERS**
Katholieke Universiteit Leuven, Belgium
Email: Paul.Valckenaers@mech.kuleuven.ac.be

## Abstract
It's current practice in manufacturing enterprises the policy of directly control all the phases of business processes, leading to an overwhelming amount of knowledge and competencies to maintain. From the Information System's point of view, these corporations had monolithic software architectures (e.g. Computer Integrated Manufacturing) with rigid control structures.
Some recent trends in manufacturing in particular, and business in general, lead to new approaches regarding the organisation and software architecture, mainly adopting distributed solutions. The paper presents several issues related to this new concept of Distributed Manufacturing Systems, its properties (e.g., autonomy) and behaviours (e.g., co-operation), as well as three different organisational models: Bionic Manufacturing; Fractal Factories; and Holonic Manufacturing Systems. A survey of existing work is presented and compared.

## 1 Introduction

Manufacturing comprises all phases needed to provide a product or service from order booking through design, production, and marketing. The majority of the manufacturing enterprises had the usual policy of directly controlling all the phases of business processes. Furthermore, there was also the idea that all these phases should be done inside the enterprise with internal staff.

Analysing different opinions (Silva, 1998), three common trends are observed in today's manufacturing context:

- Increased product variety over time;
- Increased technological complexity;
- Market globalisation.

These new trends suggests the need for different approaches, namely concerning Workforce Flexibility; Knowledge Supply Chains; Rapid Product/Process Realisation; Next-Generation Manufacturing Processes & Equipment; Pervasive Modelling & Simulation; Adaptive, Responsive Information Systems; Extended Enterprise Collaboration; Enterprise Integration (Agility Forum, URLa).

Computer Supported Manufacturing systems (e.g. CIM) had been characterised by monolithic structure, centralised co-ordination, low flexibility and adaptability, but also efficiency under intensive and repetitive processes. Original Computer Integrated Manufacturing (CIM) concept lacks

features such as distribution and decentralisation. Distribution and decentralisation have proven its value in computer science fields such as databases, file systems, etc. and become increasingly usual with the advent of the Internet. So, therefore, it is natural that these concepts, techniques and methodologies are applied to manufacturing as well. A distributed decentralised architecture is a natural way of modelling a manufacturing enterprise since the manufacturing system is comprised of several entities such as resources, tasks, tools, etc. Hence, Distributed Manufacturing Systems (DMS) are proposed as the next evolutionary step from traditional CIM architectures.

The paper is organised as follows: In section 2 an introduction to Distributed Manufacturing Systems is given, presenting its reason of existence, characteristics and behaviours. This section also presents three different approaches to model Distributed Manufacturing Systems. Section 3 introduces the field of Agent-based computing as a technology for building DMSs. In section 4 some related work is presented and compared from the point of view of co-operation. Final remarks can be found in section 5.

# 2 Distributed Manufacturing Systems

In today's global and highly competitive market, enterprises must be aware of momentary market opportunities, and quickly and properly react to customers' demands. Analysing the above mentioned trends, distribution offers suitable solutions:

- *Increasing product diversity over time* expands associated risks and costs, which are sometimes prohibitive. However, distributing responsibilities by multiple entities, risks and costs became acceptable and market opportunity can be achieved;
- *Increasing technological complexity* enforces enterprise to acquire knowledge in non-fundamental domains, which implies increased time-to-market periods. However, distributing competencies by different enterprises, each one maintains its core competency while achieving the market opportunity;
- *Market globalisation* virtually increases both enterprise opportunities and risks. Each enterprise has to operate in the global market with globally based enterprises supplying global products. However, developing relations and partnerships with such enterprises, decreases challenges and risks while benefiting from a wider market.

Different management approaches have been adopted, related to different levels of partnership, trust and dependency between enterprises (Silva, 1998):

- *Supply Chain management*, characterised by rudimentary relationship between supplied and supplier, tasks and technological competencies distribution, but centralising strategies and risks;
- *Extended Enterprise*, where entities develop more durable, coupled and mutual intervening relation, sharing technological and strategic efforts. Yet, supplied entity maintains a dominant position over suppliers;
- *Virtual Enterprise*, is a very dynamic and restructuring organisation, where supplier and supplied are undifferentiated and no dominant position exists.

Although previous description relates to inter-enterprise context, the same characteristics and behaviours (distribution, decentralisation, autonomy and dependency) are also suggested in an intra-enterprise context. Intra-enterprise workgroups emphasise self-competencies while combining efforts for a global response to external requirements.

DMS is an abstract concept (i.e., a class of systems) characterised by a set of common features and behaviours, with several specific characterisations (i.e., instantiations), named organisational paradigms.

In section 2.1 and 2.2 respectively, DMS properties and behaviours are described and in section 2.3 three organisational paradigms are presented.

## 2.1 DMS' Properties

DMS are characterised by several properties and behaviours. Such features relate both to the overall system and each composing entity. Basic properties include:

- *Autonomy* – An entity is said to be autonomous if it has the ability to operate independently of

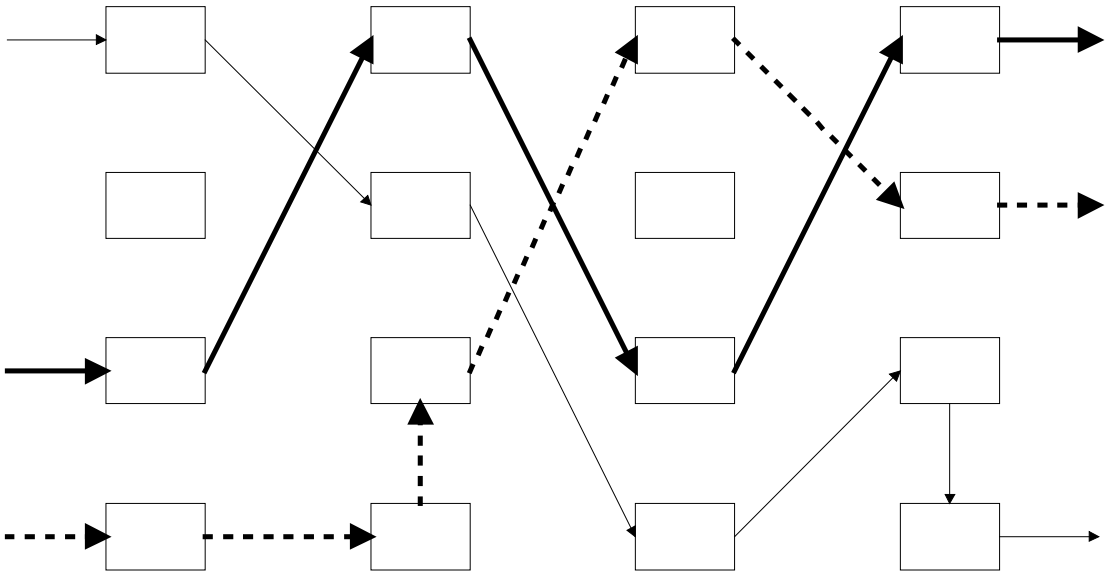

Figure 1 – Flexible Manufacturing System

the rest of the system and it possesses some kind of control over its actions and internal state (Castefranchi, 1995). I.e. autonomy is the ability of an entity to create and control the execution of its own plans and/or strategies (Valckeneaers *et al.*, 1994), instead of being commanded by other entity (e.g., a master/slave relationship);

- *Distribution* – A system is said to be distributed if different entities operate in the system;
- *Decentralisation* – Decentralisation means that an operation/competency can be carried out by multiple entities. One single system can be simultaneously centralised and decentralised. I.e. (de) centralisation refers to operations not to the system itself;
- *Dynamism* – Refers to changes in manufacturing system's structure and behaviour during operation. This express different competencies, responsibilities and relations between entities;
- *Reaction* – An entity is said to be reactive if it adjust its plans according to its perceptions;

Other properties deserve a deeper description and as such will be presented in the following subsections.

### 2.1.1 Flexibility

Flexibility is the ability the system exhibits during operation that allows it to change processes easily and rapidly in a predefined set of possibilities (Silva, 1998) each one specified as a routine procedure, defined ahead of time so that the needs to manage it are in place (Agility Forum, URLa).

In manufacturing, Flexibility is related to physical flexible machinery. Flexible in the sense that the machines (or cells) are able to execute several operations. In addition, they can quickly change among different production plans according to the part's type to manufacture at a given point in time. The concept of Flexible Manufacturing System (FMS) is very popular for companies who produce small lots of each product, mixing different lots in the production flow. One of the main problems in achieving flexibility is related to transportation. Since a product will need to pass through several workstations in order to be manufactured and different products will have different routes, the transport links between workstations should be as "free" as possible. Figure 1 shows the routing of production lots according to its type (different lines), among different workstations of a shop floor.

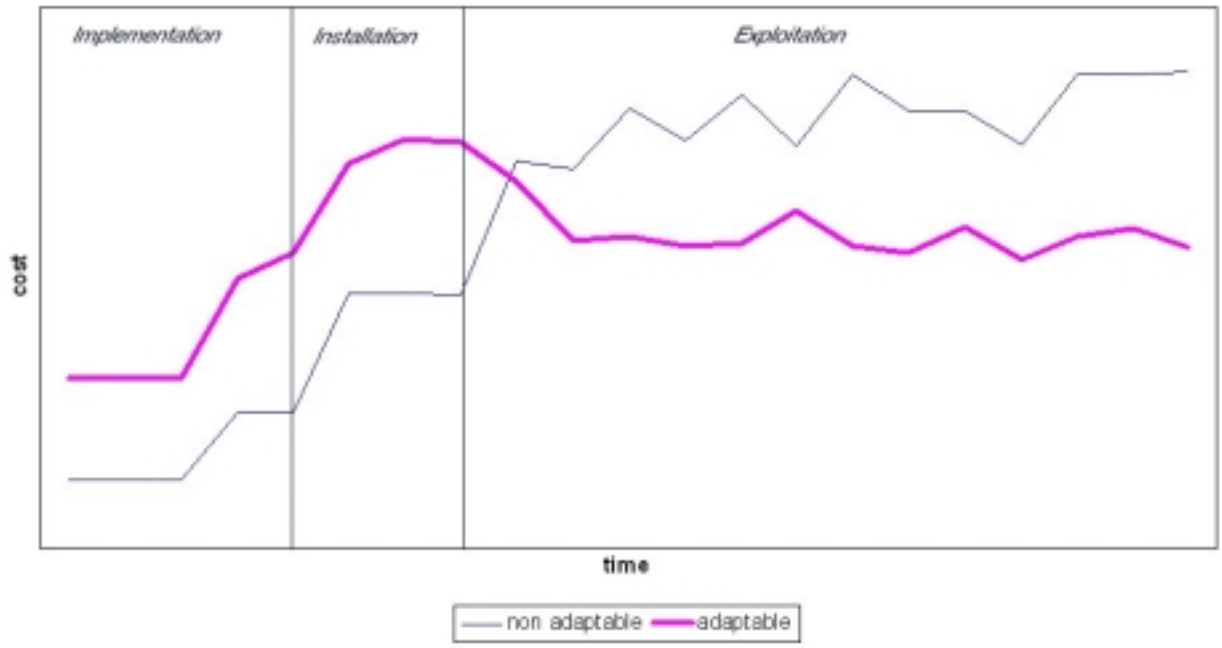


Figure 2 – Expected Cost/Time plot for adaptable and non-adaptable systems

### 2.1.2 Adaptability

Adaptability is the manufacturing system ability to be maintained easily and rapidly, in order to respond to manufacturing requirements, based on its shop-floor constraints. Adaptability refers to production facilities reconfiguration and scalability, workforce that has the incentive and flexibility to respond creatively to customer needs (Agility Forum, URLa) and thus requires flexibility. Ill-specification is a well-known synonym.

A system is said to be adaptable if it can continue to operate in the face of disturbances changing its structure, properties and behaviours accordingly to new situations it encounters during its "life-time". A disturbance is any event not previously and formerly specified (e.g. machine breakdown or a new type of product to manufacture). However, it is very hard to predict every disturbance that may happen.

Figure 2 shows the expected cost for developing, installing and running an adaptable (bold line) and a "non-adaptable" system (thin line) – values are empirical and intuitive. The first sector (from the left to the first vertical bar) represents the development phase, here the cost of developing an adaptable system is higher than a rigid one. The main reason for this is that the programming effort for fault–taulerance, reconfiguration, etc. is greater than the one that is necessary if the problem is simplified by not adding this functionality. On the second sector (installation) the cost of the adaptable system is still higher than the non-adaptable system. The effort for configuring all the components, and extra-coding necessary for additional information feedback from the hardware is the main reason for this higher cost. On the third stage (exploitation), it is expected that the adaptable system will give rise to lower costs by handling disturbances with minimum human intervention, without stopping the production or causing great delays and long waiting queues. Since this is the longest phase (the system is supposed to be "up-and-running" for several years), the higher costs of the initial phases are attenuated.

Adaptability is an important issue for a manufacturing system, especially when considering the downtime – idle production units – which as in automotive industry is around 5.000 USD per minute (Parunak, 1999).

An important functionality of an adaptable system, related to maintenance and extensibility, is some form of "plug & play". It should be possible

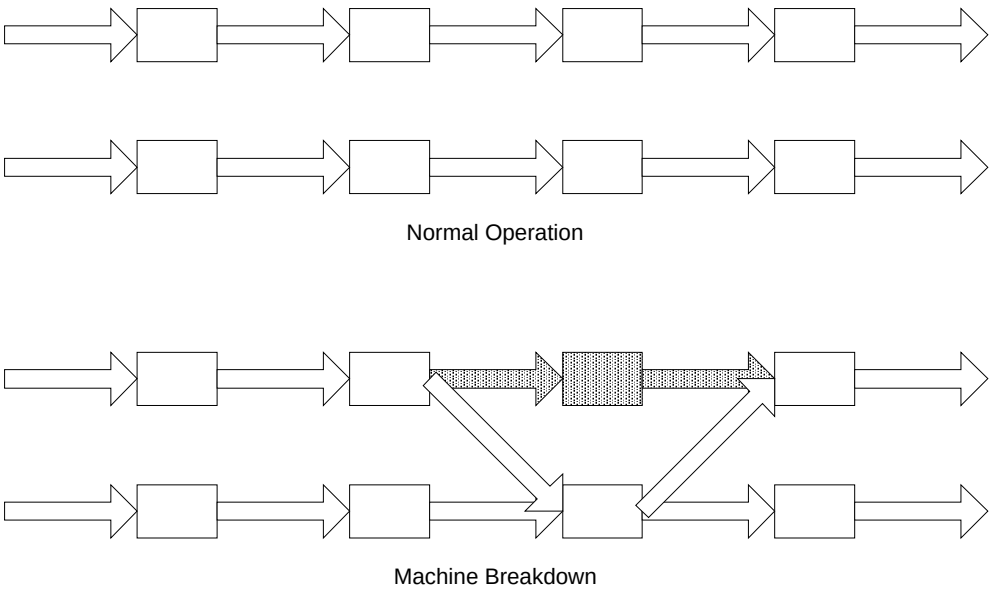


Figure 3 – Adaptability needs Flexibility

to add or replace hardware and the corresponding software modules on the fly, without stopping the system. The system should also be able to reconfigure itself for new replicated resources, or unavailability of resources, by setting new routing courses for products. This can also have a side effect of load balancing. In order to adapt, the system needs flexibility. Some disturbances can be handled without any kind of change in the manufacturing structure (e.g., slight delay of one lot). However, cases exist where is mandatory to change the physical layout (or routing) of the shop floor (e.g., a machine breakdown).

Figure 3 shows a scenario of two identical production lines. When one machine breakdowns the system can only adapt to this situation if the physical system is flexible enough to re-route the production flow. If this flexibility does not exist, the system can only adapt by not accepting anything on production line 1 – certainly not a very good solution.

### 2.1.3 Agility

Agility is understood (Agility Forum, URLa) as a management paradigm, consisting of different approaches in multiple organisational domains. However, Agility presumes system empowerment, achieved by continuous observation searching for new market opportunities. Agile manufacturing enterprise continually evolves adapting itself and pursuing strategic partnerships in order to prosper in an extremely dynamic and exigent economy.

Agility needs co-operation. According to (Agility Forum, URLb) “To bridge "gaps between visible a feasible" it is necessary to develop co-operative thinking and working. Unpredictability has to be limited by a mutual consulting, by mutually agreed scenarios, by a common sense and language.”

### 2.1.4 Agility, Adaptability and Flexibility

Figure 4 represents the Agility, Adaptability and Flexibility properties, specifying its purposes and relations in the system. The three properties relate to Change Management needed to adjust manufacturing system, in order to solve business disturbances.

Flexibility is the simplest approach and relates directly to the shop floor. It allows the shop floor to react accordingly in a predefined set of possibilities to meet primary disturbances in production. Feedback from the shop floor comes mainly with the identification of the current lot, which will serve as a base of decision for the download of the correct production plan.

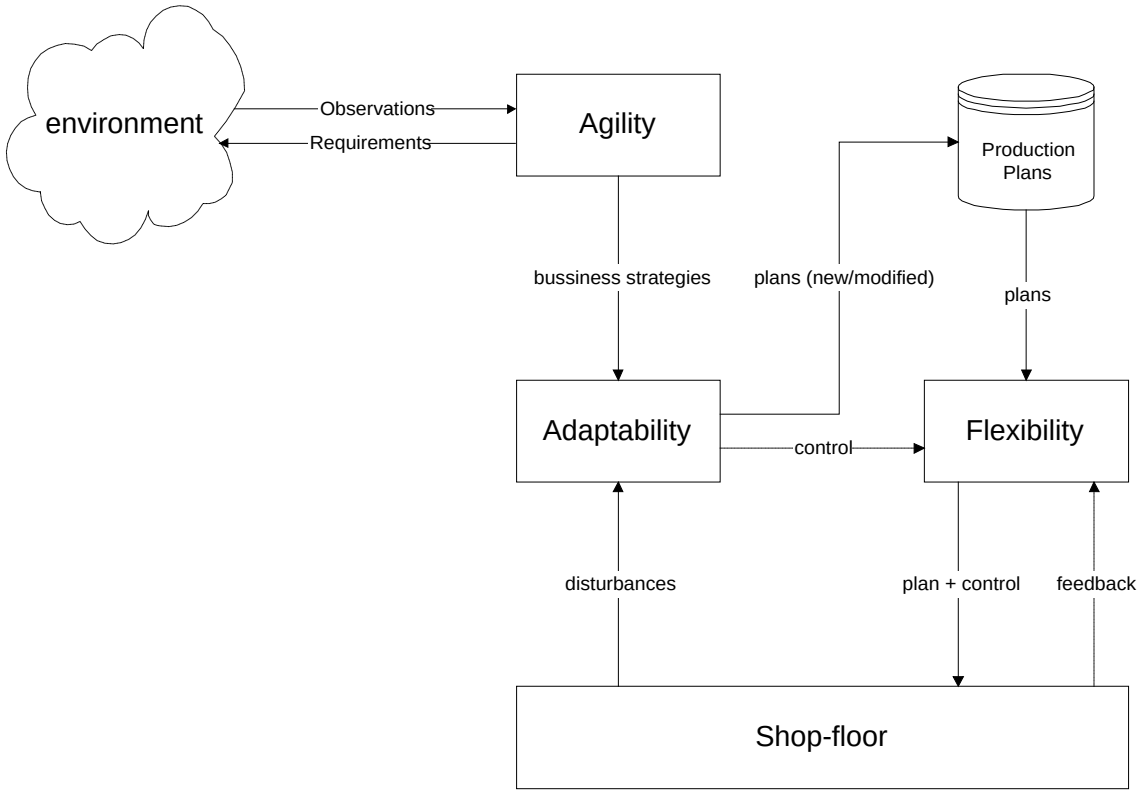


Figure 4 – Agility, Adaptability & Flexibility

On the contrary, Adaptability is based on sub-specification; i.e. the system is not completely defined, which allows run-time specification and change according to momentary requirements. Adaptability must be able to understand the disturbances in the shop floor, generating new production plans for the current situation if necessary.

Agility is the uppermost concept and relates to strategic options. Perceiving its business environment, the enterprise has to continuously evolve, adapting internally and pursuing external strategic partnerships that complement its own competencies.

Adaptability also plays an important role, by understanding the business strategies generated by the "agility module". Using these strategies, new production plans or modified ones are added to the production plans database to be used at the shop floor. These new plans may be alternative ways of doing current products, or plans for the production a new products.

## 2.2 DMS Behaviours

A Distributed Manufacturing System may exhibit several behaviours, the most important one, for the scope of this paper, is co-operation. However, the goal of any system (not only in manufacturing) is to behave coherently. *Coherence* refers to how well a set of entities behaves as a whole (Sycara, 1989). A coherent system will minimise or avoid conflicting and redundant efforts among entities (Nwana *et al.*, 1996).

The real question is: *How to achieve a coherent state?* In order to reach a coherent state, agents in a system may engage in one or more of the following processes with each other:

- *Co-operation* – a process whereby a set of entities develops mutually acceptable plans and executes these plans (Valckeneaers *et al.*, 1994). These entities explicitly agree to try to achieve a goal (or several goals) with the partial contribution of each participant. The goal needs not to be the same for each participant, but

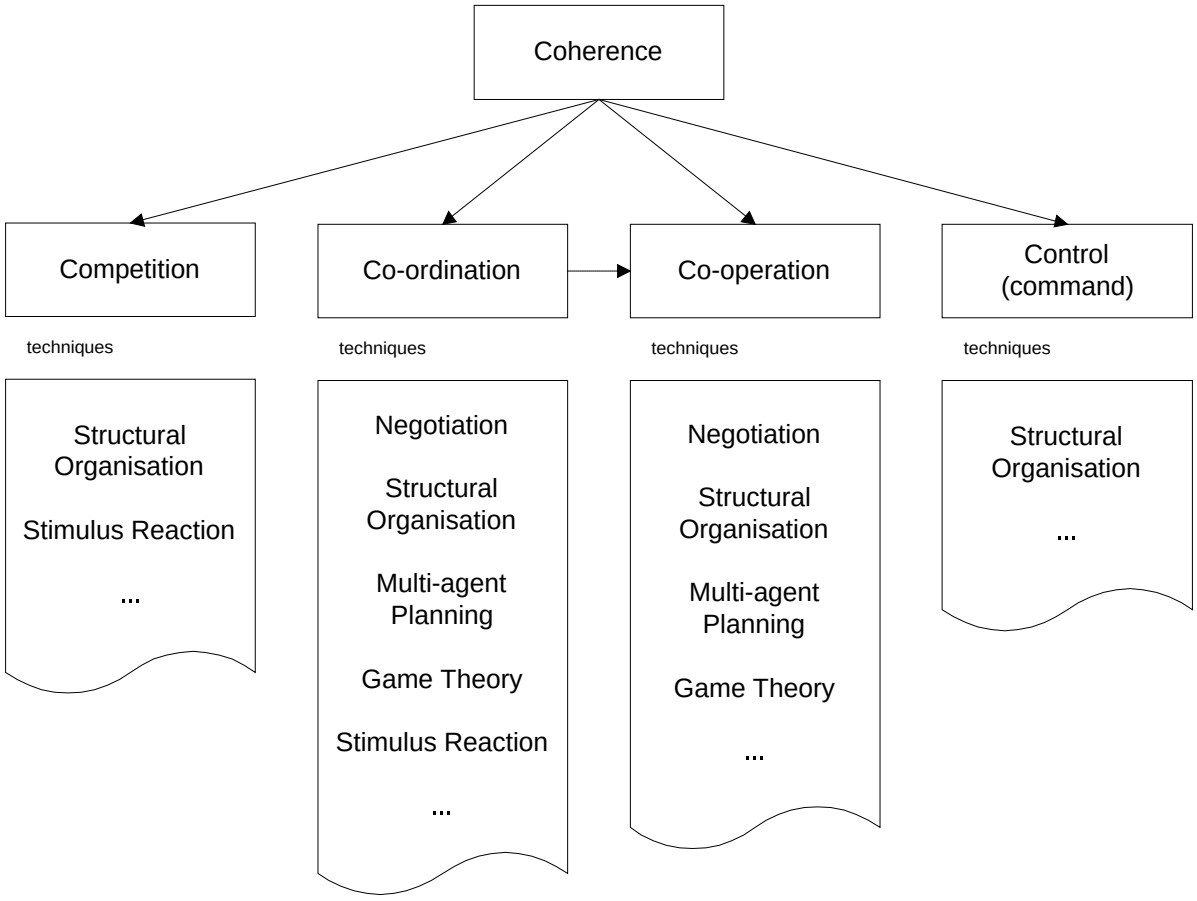


Figure 5 – Relation between Coherence, Co-ordination, Co-operation, Competition and Control

every participant expects to benefice from the co-operation process.

- *Co-ordination* – is the process of managing interdependencies between activities (Malone and Croston, 1994). I.e., "the harmonious functioning of parts for effective results" (MW, URLb).
- *Competition* – is a process whereby several entities independently try to achieve the same goal (with or without the knowledge of the other participants – explicit or implicit competition). I.e., "to strive consciously or unconsciously for an objective" (MW, URLa).
- *Control or command* – is a process where one entity rules the actions of another (limiting its autonomy).

During the system's execution it's only natural that all of these behaviours are observed. For instance, a Distributed Problem Solver will exhibit both co-operation and co-ordination. The several solvers co-operate by sharing effort among them. They also co-ordinate their activities' dependencies. E.g. on a finite element analysis, each solver operates only on a small set of data, requiring it to exchange some data with its neighbours. They co-ordinate their activities so that each one's output is valid, and co-operate by dividing the work load.

Figure 5 shows the relation among the above mentioned behaviours. This picture shows that coherence is the overall goal (i.e., whatever actions taken in the system its state is always coherent), thus imposing limitations to competition, co-ordination and co-operation. It also shows that Co-ordination may require co-operation (as shown in the Finite Element Analysis example above) but may also exist *per se*. E.g., if a person is running towards

another, and the other gets out of his way, they have co-ordinated their actions, however, they have not entered into co-operation (no explicit agreement was made).

Several techniques exist for each of these behaviours, and only some are shown in Figure 5. Examples and definitions are given next:

- *Structural organisation* – exploits the *a priori* organisation of the system, because the organisation defines implicitly the agent's responsibilities, capabilities, connectivity and control flow (Nwana *et al.*, 1996). E.g. master/slave; client/server.
- *Stimulus reaction* – allows an agent to change its plans and actions according to its perceptions. This technique involves no communication (at least direct and explicit) between entities, instead is based on the observations one entities makes from the environment and other entities.
- *Negotiation* – is the effort made by two or more entities to achieve an agreement benefiting themselves; negotiation is a technique to achieve co-operation and/or co-ordination.
- *Multi-agent planning* – avoids inconsistent or conflicting actions and interactions, by building a multi-agent plan that details all their (agents) future actions and interactions required to achieve their goals.
- *Game theory* – helps to understand the interactions of decision-makers. The basic assumptions that underlie the theory are that decision-makers pursue well-defined objectives (they are rational) and take into account their knowledge or expectations of other decision-makers' behaviour (they reason strategically) (Osborne and Rubinstein, 1994). E.g. prisoner's dilemma.

For a detailed description of co-ordination issues (and its relation to co-operation, as well as co-ordination techniques) please refer to (Nwana *et al.*, 1996).

### 2.2.1 Co-operation

One popular technique used in many DMSs is the Contract Net Protocol (Davis and Smith, 1983). When an agent poses a request in the system, several agents able to provide the requested service bid their offers, thus competing with each other. When the consumer agent reaches an agreement with one of the provider agents, they began to co-operate. Also, refined models of the Contract Net may allow for the proposal and counter-proposal exchange between agents negotiating the best deal for everybody.

As shown above this system exhibits several behaviours (even for a simple model as the original Contract Net). However, the one that lasts longer is co-operation, thus the fact that this kind of system is also called co-operative system.

Co-operation in manufacturing is intuitively proven necessary when adopting a distributed solution:

- A product needs several operations, and probably these operations are performed on several resources. The job (i.e., the manufacturing order) must establish a contract with the needed resources in order to be executed, thus entering a co-operation process.
- The resources must co-operate to manufacture a product and must co-ordinate their actions since the dependencies between operations must be observed.
- When looking at the multi-enterprise level, each company involved establishes plans accepted by the other partners for the fulfilment of the global objectives. They co-operate to manufacture a product (e.g. providing different assembled parts) and/or to provide some kind of necessary service (e.g. accounting, distribution, quality testing, etc.).

The theory of neo-liberal institutionalism (Axelrod, 1984) claims that co-operation rather than competition becomes more advantageous in any given scenario, following the hypothesis that the main goal of an entity can be represented by its individual payoff rather than its relative gain. Under this hypothesis, provided that its payoff is positive, an entity does not care if another gets a higher payoff (implying a negative relative gain) (Pontrandolfo and Okogbaa, 1999).

There are also social and political trends towards co-operation. Through Human history, tribes with individualist behaviour gave rise to ancient civilisations (Romans, Egyptians) with imperialistic motivations, which continued to exist in the modern civilisations. Nowadays, through international co-operation protocols, one can usually see a richer country helping a poorer one. The poor country's objective is to grow its economy, while the richer country is seeking for political advantages for future

times, as well as economic advantages for its enterprises who decide to invest in the other country. They have different individual objectives, however the concerted actions of both give then benefices and thus they have co-operated.

However, co-operation is nothing new to the world. The old exchange mechanism used before the invention of currency is a form of co-operation. People traded goods they had in excess for lacking ones. The usual commerce people make in everyday life, buying and selling items is a form of co-operation. There are two participants that explicitly interact with benefits for both, and that is the essence of co-operation.

## 2.3 DMS organisational paradigms

Distributed Manufacturing has been adopted for a long time but its organisational structures were only formalised and proposed as an essential solution in recent years.

In order to achieve the above mentioned characteristics and behaviours, several organisational paradigms were proposed, namely the Fractal Factory (Warneke, 1993) (Sihn, 1997), Bionic Manufacturing Systems (Okino, 1993) and Holonic Manufacturing Systems (IMS, 94), which are comprised in a broader term designated Open Hierarchical Systems. These paradigms suggest the idea that manufacturing systems will continue to need a hierarchical structure besides the increased autonomy assigned to individual entities. They also advise the hierarchy is needed in order to guarantee the inter-entities conflict resolution and maintain the overall system coherence and objectivity resulting from the individual and autonomous attitude of the entities.

### 2.3.1 Fractal Factory

Fractal Factory paradigm is based on mathematical fractal concept and the associated theory of chaos. The fractal is characterised by constant evolution respecting to its partners and environment (Tharumarajah *et al.*, 1996). (Warneke, 1993) defends the factory of the future will be substantially different, specially concerning its dynamic organisational structure. It suggests the enterprise will abandon the function oriented organisation and will adopt the project orientation management (Sihn, 1997). This approach implies the organisational structure will encapsulate the process and the technology, therefore forming a cybernetic structure.

Thus, the factory will not have a predefined organisation, but a more or less restricted set of resources with static capabilities, and a very dynamic set of projects (tasks). Although the resource is restricted in terms of quantity and capabilities, combining multiple alternatives, the enterprise reacts timely and efficiently to business requirements. In addition, the enterprise is naturally apt to combines its competencies with external entities, increasing its ability to satisfy market opportunities.

Conceptual behaviour is observed when a new project is introduced into the system, initiating a very dynamic process, responsible for resource/task negotiation, leading to constant changes in the enterprise structure and organisation. Each resource notices the new project event and concurs to the negotiation (it can even compete) with remainder resources. Considering those different project request different competencies, resources are dynamically associated to projects; thus, each resource can belongs to multiple projects yet maintaining its core competencies.

This process is very dynamic and efficient, as the resource can (well) decide about its own behaviour and does not rely on a higher entity to do so.

The fractal factory must be understood as a non-linear system (Sihn, 1997), structurally reactive and adaptive to the dynamic context. Additionally, the concept easily models different enterprise dimension (e.g. strategic, social, cultural, informational and technologic) thus easily models enterprise reality in different perspectives.

### 2.3.2 Bionic Manufacturing System

(Okino, 1993) introduced the Bionic Manufacturing Systems and the *Biological-oriented* expressions, to present the manufacturing system concept based on structures and behaviours observed in biology. In his analyses, Okino mention the fact that from the simplest through the most complex living form, inside of certain hierarchically ordered relations, all manifest autonomy, spontaneous behaviour and social harmony. In biologic systems, autonomy and spontaneous behaviour concerns to the responsibility for its activities and self-division

according to a genetic code named DNA[1]. Furthermore, consistency and goal orientation are conceptually supported by the genetic inheritance, in which the genetic code of the entity (be a cell or a complete living being) is inherited.

BMS applies to manufacturing systems the structure and organisation behaviour of the living beings, defining a parallelism between biologic systems and manufacturing systems.

The cell, organ or living being is modelled in BMS by the modelon concept, which is composed by others modelons, forming a hierarchical structure. Each modelon has a very static set of properties and behaviours, which can be combined with others, forming distinct entities also designated modelons. The notion of DNA inheritance is translated to manufacturing context by the properties and behaviours that are passed intrinsically to developed modelons.

The information and communication systems in manufacturing systems arise from the surrounding environment existent in biological systems. In addition, the notion of enzymes and its role in the living beings is modelled in manufacturing systems by entities called supervisors. These entities are very important since they are responsible for the regulation and control of the system. Furthermore, the supervisors also play an organisational and structural role, influencing the modelons relations, imposing self-division or aggregation, in order to meet requirements imposed by the environment. The self-division and aggregation are specific mechanisms implemented in BMS to fit the DMS conceptual behaviour of co-operation.

### 2.3.3 Holonic Manufacturing Systems

The Holonic paradigm arises from Herbert Simon and Arthur Koestler studies about biological society evolution and organisation. Simon observed that complex systems are hierarchical systems formed by intermediate stable forms. Later, analysing Simon theory and comparing it with its own observations, Koestler perceived that each system and its intermediate forms do not exist as auto-sufficient and non-interactive elements. On the contrary, they are simultaneously a part and a whole, a container and a contained, a controller and a controlled. In order to represent this hybrid nature, Koestler proposed the terms 'Holon' and Holarchy. Holon is a combination of Greek word *holos* (whole) with the suffix *on* which, as in prot*on* or neutr*on*, suggests a particle or part (Koestler, 1967). Holarchy is "a hierarchy of self-regulating holons, in supra-ordination to their parts, in sub-ordination to the higher levels and in co-ordination with environment" (Koestler, 1967). Additionally, the IMS – HMS group defined a set of properties related to the manufacturing systems based on the holonic paradigm:

- The holonic manufacturing system entities are autonomous and co-operative;
- Holon has information about itself and the environment;
- Each holon is composed by other holons and thus each holon is a also a holarchy;
- Each holon can dynamically belong to multiple holarchies;
- The holarchy has fixed rules and directives (the *canon* (Tharumarajah *et al.*, 1996)).

In a holarchy, plans are partially specified in higher level holons; travels down the holarchy being progressively completed. Task-results travel up the holarchy, generating control attitudes by upper levels down to sub-ordinated holons.

The holon is an autonomous entity, includes operational features, individual goals and ability to define its own tasks and execution plans. However, combines its set of competencies with its lateral partners, with whom co-operate in order to achieve both individual and system goals.

### 2.3.4 Comparison

The three paradigms have different origins, which impose different approaches. Nevertheless, hierarchical structure and co-operation are essential properties of these paradigms, forming very dynamic systems while maintaining overall goal orientation and coherence.

Based in (Tharumarajah *et al.*, 1996), Table 1 resumes the most important structural concepts, while Table 2 refers to the operational behaviours.

The Creation and Mouldation parameters describe respectively the principles behind the creation and definition of each basic unit. In FF and HMS, any fractal or holon represents a specific

[1] The DNA is contained in chromosomes, *stores the genetic information about the individual to further transmission to its descendants, during the cell division. Each species has its own DNA composition and even each individual has its own DNA composition that makes it unique.*

| | | FF | BMS | HMS |
|---|---|---|---|---|
| **Unit** | **Creation** | Predefined | Genesis & Dynamic | Predefined |
| | **Mouldation** | Multi-dimension | Multi-functional (DNA based) | Physical & Functional |
| **Group** | **Creation** | (Project oriented) | (DNA & enzymes) Predefined & Dynamic | Predefined & Dynamic |
| | **Process** | Regroup | Unit division & DNA combination | Regroup |
| | **Goals** | Projects & Optimisation | Functional (DNA based) | Functional |

Table 1 - Structural aspects of FF, BMS and HMS

resource characterised by its competencies, capabilities and other particular elements, thus the unit and its set is perfectly defined at design time. In BMS on the contrary, the modelon models the biologic entity, characterised by its birth, living (including reproduction) and death. At the beginning of the system it exist a set of predefined modelons, but during existence they reproduce themselves generating child modelons and other complex structures, constituting a very dynamic concept.

The Creation and Process parameters concerning group structure, describes respectively the purposes behind the group creation and the applied procedures. FF grouping is project oriented, which means the system applies it to deal with project challenges but no new units are generated during the operation. The BMS follows the biological approach, whereas unit respects co-ordinators command and its division code included in DNA. In HMS, the group creation is based in predefined characteristics, thus design specific.

Concerning the Goals parameters, the FF orients its goals to project execution and optimisation. Again, the BMS relates to DNA code to specify its goals, while the HMS holon has its goals specified by its base and predefined functionality.

Resuming, Fractal Factory, due to its multi-dimensional specification and project orientation, perfectly fits negotiation and dynamism requirements. The BMS approach relies on DNA and enzymes concepts, to command division and reproduction by DNA combination. The HMS paradigm uses a functional predefined approach, which makes it the most traditional, however the easily understandable and structural adaptable.

Table 2 resumes the dynamics properties related to the paradigms. The Autonomy parameter represents the entity faculty to pursue its own goals. Fractal entity is created and defined multi-dimensional, including its goals and capabilities. The Fractal Factory main singularity is Vitality, which means the entity continually attempts to achieve better results. The holonic structure is characterised by hierarchical rules called the *cannon*, which imply in the entity a limited autonomy. The modelon, besides the fact is created with predefined goals, it is very dependant on the environment contents and operators (enzymes) commands.

Individual Planning is related to Autonomy parameter, since the ability to define its own plans largely depends on the Autonomy faculty. Fractal Factory is project oriented, which means the system restructure itself to meet project needs. Each fractal engages in negotiation in order to participate in project execution and thus improves its own goals and performance. In BMS approach, modelon individual planning largely depends on environment and operators command. In HMS, holon individual planning is hierarchy dependent, but it allows the holon to dynamically refine sub-plans.

| | | Fractal Factory | BMS | HMS |
|---|---|---|---|---|
| **Autonomy** | | Individual goals & Vitality | Answer to changes in the environment and operators | Limited by hierarchical rules |
| **Individual Planning** | | Continuous search for goals and optimisation | Minimal, most based on reaction to environment and operators | Some defined hierarchically, most top-down refinement |
| **Individual Control** | | Continuous control by state comparison | Reactive to changes in the environment | Reactive and based in intermediate stable forms |
| **Hierarchical Control** | | Reactive conflict resolution between adjacent levels, through negotiation | Hierarchical tasks, and results injected into the environment | Bottom-up results and subsequent hierarchical control. |
| **Co-operation** | **Event** | Project arrival | Environment contents & Operators commands | Functionality |
| | **Process** | Negotiations through regroup | Reproduction and DNA combination | Negotiation through regroup |

Table 2 - Dynamic properties of Fractal Factory, BMS and HMS

Concerning Individual control, Fractal entity, based on vitality characteristic, always pursue optimisation by comparing previous, momentary and intended states. Modelon reacts to changes in the environment. The holonic paradigm relies on the concept of intermediate stable forms, which settle the holon (simple entity or an holarchy) as an auto-sufficient and auto-stable entity, reacting to needs in the operation context.

Hierarchical Control represents the procedures applied in order to maintain overall system operability, integrity and coherence. In FF, hierarchical control uses dynamic and reactive negotiations between adjacent levels. This characteristic strongly enforces the need for advanced state/goals representation and process. In BMS, tasks and operation results are injected into the environment, and will influence proportionally the rest of the entities and its tasks. In HMS, the operation results flows bottom-up in the hierarchy, which in turn will control the subhierarchies based in these results.

Respecting to Co-operation, two sub-parameters are considered, the event that triggers the co-operation and the process whereby it is accept between entities. In FF, since the main idea of the paradigm is the project orientation, as a new project arrives into the system, all the entities knows the fact and can engage in negotiations to co-operate. The BMS paradigm relies in the environment contents and operators commands in order to trigger co-operation that is achieved by reproduction and DNA combination between modelons. The HMS structure uses top-down approach to define tasks and plans, thus, as soon as the entity receives the order, its individual characteristics and states can impose the need for co-operation. At that moment, the entity starts negotiation for co-execute (co-operate) sub-tasks.

Resuming, the BMS paradigm suggests some properties intimately similar to modern enterprises, specially relating to the environment: full of information and chances to improve business, which impose awareness and perception. However, managing so much information may cause negative consequences due to stabilisation and productivity, if no co-ordination and hierarchical competencies are perfectly defined.

The Fractal Factory paradigm is the most modern approach, in the sense that it relies on individual entities autonomy and vitality to maintain and increase system dynamics and performance. Further, it is based on mathematical formalisms that makes it the preferred approach to design and specification.

However, its application tends to be complex especially respecting to implementing navigation and co-ordination mechanisms. The HMS paradigm is the most traditional, due to its structure, organisation and functional orientation. Also, is the most stable due to the statically defined hierarchical rules: the *canon*.

Apart the differences between these and other organisational paradigms, it is suggested that conceptually different systems can (co-) operate simultaneously, whereas the system is a complete functional or a single isolated entity. I.e., characteristics and behaviours related to different paradigms can be combined into a single entity. In order to achieve it, entities must implement different behaviours and interfaces, but mould internal coherent behaviours and states.

# 3 Agent Technology

Aspects of distributed manufacturing systems have been described, which are important and have an impact on the performance of such systems. Overall behaviour of such distributed systems depends on the interaction and co-ordination of the distributed elements. Agent technology provides means to implement such distributed systems as a set of agents, which are autonomous acting software entities with capabilities to co-ordinate activities to create a desired overall system behaviour. Agent technology is especially attractive to application domains like manufacturing environments, because "agents are best suited for applications that modular, decentralised, changeable, ill-structured, and complex." (Parunak, 1998).

In the following, a short overview of important aspects of this technology will be given. For more detailed information refer to (Wooldridge and Jennings, 1995) and (Nwana, 1996) among many other valuable sources of information. Starting with a description of agent architectures, the importance of interaction and communication abilities of agents to enable co-ordination and co-operation is outlined.

## 3.1 Agent Architectures

A lot of work has been done to answer questions about necessary architectural issues of agents. Two main types of architectural concepts have emerged over time, "deliberative" and "reactive" architectures. Agents with a *deliberative* architecture maintain an internal symbolic representation of the world, have planing and reasoning capabilities to pursue specific goals, and are able to communicate and negotiate with other agents to achieve some form of co-ordination. A famous deliberative architecture is the so-called BDI-architecture (Belief, Desire, Intention) (Rao and Georgeff, 1995). This concept describes the internal processing state and behaviour of an agent in terms of mental categories like beliefs, desires, and intentions. Beliefs are the agent's current view or expectations of the current state of the world. Desires specify preferences or goals over future world states or preferences for actions, which can produce such a desired world state. An intention represents a commitment of an agent to perform a specific plan or course of action to satisfy a specific desire or goal. Therefore, an implementation of this architecture will provide data structures, which allow to manage facts, goals, and plans to represent beliefs, desires, and intentions. The agent stores beliefs or facts about the world, about other agents, and the agent's own state. Goals represent desires (actually the "consistent subset of desires, that an agent shall pursue, because – according to the "property of realism" – an agent must belief, that its goals are achieve-able), and plans implement intentions to satisfy goals.

*Reactive* agent architectures allow to create agents, which have a close connection of sensors to actuators, which provides them with a kind of "perception-driven" reactivity. Events from the environment of such an agent trigger pre-wired patterns of behaviour. This behaviour is described as situation-action rules. These agents have no shared symbolic representation of the world and no explicit reasoning capabilities. Brook's concept, the "Sub-sumption Architecture" (Brooks, 1986), is an example, how to implement purely reactive agent architectures. Without any internal representation of the world, a stimulus-response schema works well enough to allow an agent to act in quite complex environments. Brooks implemented a couple of successful applications in the robotics area, based on this architecture.

An important aspect of agent technology is the ability of agents to co-operate in problem solving and to co-ordinate their activities to generate co-operation in an agent application. Co-operation is a concept of deliberative architectures, where agents are able to maintain beliefs about other agents and

where these agent architectures provide communication facilities to enable a knowledge exchange between agents. Aspects of interaction, communication, co-ordination and implementation of negotiation schemata are described in the following.

## 3.2 Agent Interaction

Multi-agent systems are distributed applications where single agents try to achieve co-ordination of their activities with other agents to create an intended overall system behaviour. Co-ordination of activities can be achieved by a specific form of interaction between agents. Agents, which co-ordinate themselves to pursue a specific shared or global goal, are "co-operating" in this activity.

Co-operation and co-ordinated behaviour of agents depends on how single agents choose their course of actions. This decision process is based on information, which is available to an agent. It is very popular to describe agents exchanging messages and updating their internal models or beliefs about the world and other agents. But, as outlined above, there are different forms of agent architectures, especially those, which implement simple reactive agents without any symbolic representation of the world. Such agents respond to events with predefined responses, triggered from state changes in the environment, in which these agents are embedded. In particular, the notion of "indirect interaction" between agents resembles behaviour, which can be found in biological systems like ant colonies. Such systems are based on concepts like pheromone spreading. According to this concept, agents do not interact directly, but with their environment, posing or broadcast information into their environment, from where it can be picked up by other agents. We therefore distinguish two forms of interaction:

- *Direct interaction* – where agents directly exchange some form of messages;
- *Indirect interaction* – where agents have no direct communication with each other, but somehow interact with their environment, where on the one hand they pose information without specifying a specific recipient, and on the other hand just collect information without knowing about the sender.

### 3.2.1 Direct Agent Interaction

In terms of "direct interaction," agents are described as software entities, which exchange messages to communicate state information, goals etc. to maintain their models of the world. Agents "co-operate" by co-ordinating their activities. Using concepts of negotiation, tasks are shared between agents to allow a concerted problem solving. Co-operation is an important concept in agent systems. The process of co-operation can be roughly described to take place in following three phases:

(i) *Negotiation*: agents negotiate how to allocate tasks to agents within an agent community; this process is called "task sharing." A famous example for negotiation is the previously referred Contract Net Protocol;
(ii) *Execution*: agents perform the assigned tasks and share partial results; this is called "result sharing";
(iii) *Result Reporting*: agents report success/failure of their activities to the community.

The two concepts "task sharing" and "result sharing" therefore determine co-operation. For implementing task sharing between agents, a specific interaction schema must be established between agents. Usually one agent will be determined to take the role as a "master" or "manager," which distributes tasks or collects results. In detail, task sharing itself can be implemented in following flavours:

- *Static allocation*: here a specific agent is available in the agent community, which knows how to decompose a specific problem into a task hierarchy and how to allocate these tasks to agents;
- *Predetermined dynamic allocation*: this is implemented by the Contract Net Protocol (Davis and Smith, 1983); agents dynamically get the role of a master or contractor (a contractor itself can in turn act as a master for further sub-tasking), but there is still one top-level predetermined manager in the agent system;
- *General dynamic allocation*: no top-level predetermined master exists, problems are posed to all agents, each agent tries to come up with an own solution; the problem here is to find the "best" solution.

The Contract Net Protocol is a well-known interaction schema and widely accepted implementation of negotiation between agents. As already outlined, it allows task sharing between agents. Agents negotiate contracts for performing specific tasks among each other. During this negotiation process, agents can take over one of two roles, manager or contractor. The manager tries to create subtasks for it's own task and start the negotiation process to find contractors. In particular, the contract net protocol works in the following way:

(i) A task is assigned to an agent for execution;
(ii) If the agent cannot execute this task, other agents must be found for help; the agent is now a manager, decomposing its unsolvable task into subtasks;
(iii) A negotiation process is started with other agents to find contractors;
(iv) Potential contractors send bids (expressing their capability to manage the contract) to the manager;
(v) The manager uses the bids to choose a contractor.

### 3.2.2 Agent Communication

Direct interaction of agents is based on their ability to communicate and exchange messages. Speech Act theory (Austin, 1962) (Cohen, 1988) (Searle, 1969) was originally developed to model verbal communication between humans and is used to describe the communication behaviour of agents. In that sense, messages, which are exchanged by agents, are so-called speech acts. A speech act is determined by the three aspects "Locution" (physical utterance), "Illocution" (the transfer of the intention of the sender to the receiver), and "Per-locution" (the reaction of the receiver to the illocution). Agent communication has to do with these aspects. Agent communication languages like KQML (Finn et al., 1997) have been developed, which provide message types, so-called "performatives", to implement speech-acts. These performatives implement speech acts. The message type expresses the "Illocution" or intention of the message. KQML provides message types like "ask" or "tell," allowing agents to ask other agents for specific knowledge and giving possibilities of response. KQML assumes that each agent contains a kind of "virtual knowledge base," on which queries can be performed. KQML only provides message types, which express the intention of the sender, the actual content of the message must be formulated in a different language. Here KIF (Knowledge Interchange Format) (Genesereth, 1992) is often used.

Communication between agents can only take place if there is a common understanding between agents about the concepts communicated in messages' content.

### 3.2.3 Agent Understanding

Ontologies are a means to create shared understanding between communicating agents, describing concepts and their relationships. (Gruber, 1993) points out that ontologies describe a common vocabulary that models a specific application domain. Each ontology defines mainly a set of classes, their relationships and constraints for their interpretation. An ontology therefore contains descriptions of domain concepts and their relationships. It is comparable to a data dictionary. These concepts can be used by agents to interpret the content of exchanged messages.

The modelling of ontologies resembles the data modelling process known from database schema modelling and object-oriented design. According to that, Entity-Relationship diagrams or object-oriented concepts like UML (Pooley, 1999) can be used to model ontologies. A special modelling language is described by IDEF5 (IDEF URL) providing a method determined to create ontologies.

# 4 Applications and examples on co-operation in distributed manufacturing systems

In this chapter examples about applications of agent based technologies in manufacturing will be presented. Common to all these is at least inherent co-operation between the system entities. At the end of this chapter a short comparison on the features and characteristics of these systems, from the point of view of establishing co-operation, can be found.

## 4.1 Supply chain management

### 4.1.1 Agent based manufacturing enterprise integration

In their MetaMorph I system (Shen and Norrie, 1998) provide a framework for building an agent based system. MetaMorph I is a federation organisation, where intelligent agents can link with mediator agents to find other agents in the environment. The mediator agents assume the role of system co-ordinators by promoting co-operation among intelligent agents and learning from the agents' behaviour. Mediator agents are able to expand their capabilities to include mediation behaviours, which may be focused upon high level policies to break the decision deadlocks (cf. the 'staff' agent in (Van Brussel et al., 1998). Mediator agents can use brokering and recruiting communication mechanisms to find related agents for establishing collaborative subsystems ('co-ordination clusters' or 'virtual clusters'; cf. 'co-operation domains' in HMS (Rannanjärvi and Heikkilä, 1998)).

In MetaMorph II (Shen and Norrie, 1998) have applied MetaMorph I framework in the enterprise function level. A Design Mediator does integration of design and manufacturing. Marketing functions are integrated by easy-to-use interfaces to request product information (performance, price, etc.) and material supply and management by a material mediator which co-ordinates material management (material handling, stock, etc.) sub-system. In addition there are simulation mediators for production simulation and forecasting, and execution mediators co-ordinate execution of machines, AGV's, etc. Dynamic scheduling and rescheduling is established by the mediation mechanism and Contract Net bidding mechanism. The Machine Mediator selects a machine after receiving bids/propositions from the machines to the request. Non-contracted machines are memorised as alternatives for rescheduling within failures/machine breakdowns. A prototype implementation has been reported with four mediators: enterprise (enterprise administration centre), design (integrates a functional design system), resource (co-ordinates an agent based manufacturing scheduling subsystem), and marketing (integrates customer services).

### 4.1.2 Integration of a group of agented manufacturing capabilities

AARIA (Autonomous Agents at Rock Island Arsenal) by (Parunak *et al.*, 1998) aims to derive actual control decisions from schedules, which are created in a distributed manner by agents. Manufacturing capabilities (people, machines, and parts) are encapsulated as autonomous agents.

AARIA is expected to be suitable for manufacturing control at a significant distance from equipment control itself (MASCADA, URL), including a multi-site co-ordination and control. In fact, AARIA has been lately extended to cover the whole supply chain management from customer to manufacturing, especially trying to demonstrate, that this can be facilitated based on internet technologies. Goods will be bought over the Internet through a direct dialog with the distributed manufacturing capabilities needed to make and deliver orders. This will increase the accuracy of commitments to the customers, reduce lead time and cost of final delivery, and find the best compromise between the customers' desires and current manufacturing capabilities. AARIA provides fundamental integrated ERP and MES functionality. The MES functionality includes basic 'what-if' simulation, finite capacity scheduling, and intelligent shop floor interfaces. The ERP functionality includes basic planning, order entry, purchasing, bill-of-materials management, inventory management, resource management, personnel management, integrated financials, and reporting.

Currently AARIA uses a simple protocol, whereby a customer requests a specific product, the manufacturing system responds with a bid of costs vs. delivery time, and the customer chooses a delivery time and cost that satisfies him. Further on, each agent in the internal supply chain then makes and maintains a commitment to perform his part of the job. Within this scenario, later stages of production become customers to earlier stages. When a customer requests a product, requests for bids propagate down the supply chain from part broker agent to unit process agent and to resource agents. From the resource agent the same continues to other part broker agent etc. Resource broker agents and part broker agents that sell raw material stocks, issue bids, which propagate up the supply chain, getting folded with the production costs at each stage, until a final bid is presented to the

customer. After the customer has chosen a cost and delivery date, purchase orders propagate back down the supply chain establishing commitments for the individual agents,

#### 4.1.3 Intelligent assistants to manufacturing planners and schedulers

RedPepper (PeopleSof, 1997) originates from the work of Monte Zweben in NASA Ames Research Centre in 80's in fundamental work on 'constraint based scheduling' on Ground Processing Scheduling System, for use in scheduling of Shuttle Orbiter refurbishing. It is currently the core of PeopleSoft's ERP software identified as 'intelligent assistants to manufacturing planners and schedulers'. RedPepper is based on 'Response Agent' (Production Response Agent; Enterprise Response Agent) technologies, which provide real-time planning and scheduling to respond and adapt to changes. The target is for supply chain optimisation and real-time order promising. RedPepper allocates, manages and tracks of cost for material, machine and labour usage. The considered constraints include promise dates, request dates, inventory shortages, aggregate capacity, safety stocks, excess stocks, raw material shortages. The resulting supply chain plans and schedules are based on up-to-the minute transactional or event information, on a real-time 'net-change' basis. One of the applications/functions is sequencing and scheduling, which builds schedules and dispatch lists for each resource. The algorithmic solution is interactive (operator assisted) constraint satisfaction with scorecard reports on the number of times each constraint has been violated. In addition, a plan optimiser improves schedules. RedPepper works on-line for adjusting resources without disturbing on-going operations. Change management tracks Bill-Of-Material's, costs, prototypes, cost simulations, the transfer of design to production, and engineering change orders integrating these into working applications.

RedPepper views a single model on organisation's supply chain as one logical model, and splits it into several physical representations across a number of servers which is very efficient for one production line, but can lead to difficulties if dealing with multiple production sites. In principle RedPepper treats global performance by observing production overhead and proposing improving minor actions.

### 4.2 Distributed Scheduling Systems

#### 4.2.1 Distributed scheduler

Wishes behind the distributed scheduling system ReDS (Hadavi, 1994) were not so interested on optimality of schedules, but more in having control of lot movements and some realistic prediction on the near future (hours, weeks) schedules. ReDS distributes scheduling criteria over agents. In developing ReDS it was notified, that in order to design a complex control systems easier it is essential to subdivide the problem into independently smaller and less complex problems; however, by doing so one loses some measures for global optimality.

ReDS considers main shop floor events: new orders entered, machine breakdowns, reworks, lot splitting, lots on hold, and bottleneck areas. The purpose of REDS is to find a structure that can be used for designing production control systems. The proposed structure is composed of recursively defined autonomous modules, referred as planning agents. The architecture is based on recursively defined autonomous modules that have a predictive element and a reactive element. The in-flows of an agent are consisting of goals/tasks from higher layer; overriding commands from 'higher' agents, events from shop floor, and guidance for performance from statistician. No centralised co-ordination module exists. Out-flows are consisting of operation instructions to the shop floor (operators). The commands are handled hierarchically within the layers: higher layers overcomes lower layers, and layers deal with specific time horizon (shortening while going down). In principle this makes a top-down goal decomposition mechanism, with bottom up disturbance handling ('shock absorption'). The agent layers are as follows: sequencing & dispatching, detailed scheduling, feasibility analysis for resource & material availability, order release for minimum waiting times and a global statistician (monitoring data, detecting trends, interpreting events to forward to proper listeners). Communication between agents is done by contract net protocol with broadcasting.

ReDS introduces flexibility in two ways. It treats intentional changes flexibly in a top down manner by goal decomposition and it treats disturbances flexibly, in a locally reactive manner by bottom up 'shock absorption'. Although it does not provide a global optimum, it treats global performance by the statistician.

#### 4.2.2 Micro-opportunistic scheduling

MICRO-BOSS micro-opportunistic scheduling system (Sadeh, 1994) treats scheduling problem as a constraint satisfaction problem. Each job has an earliest start time and due time, and identification of the need for backtracking quickly is done by aggregated demand profiles. Although in principal it targets to find feasible schedules, it is possible to define due dates and take decisions so that the tardiness is minimised. MICRO-BOSS defines aggregate demand profiles for each resource. Using probabilistic models as a resource inspiration, it is possible to identify which machines are most bottleneck in which time periods, and for which operations. Once the decision to be taken is identified ('the variable ordering is done' or 'the operation to be scheduled is chosen'), the value evaluation is to be done ('the variable instantiated'). One could look then at the (local) 'optimal' value for that variable (greedy heuristics), but Sadeh also looks at the survivability of these decisions, i.e., the chance the decision will not have to be backtracked.

In the CORTES project (Sycara et al., 1991), Sycara extends the concept of micro-opportunistic scheduling to distributed scheduling. Agents construct aggregated demand profiles, maintain them and exchange updates of them in a regular basis. As agents take scheduling decisions, new constraints are added. Different agents solve sub-problems of the scheduling problem, for a subset of the orders and a subset of the resources. Agents exchange the demand profiles to represent the agents' intended resource usage for different time intervals. This means agents have information about behaviour of other agents, but it is incomplete. Although the demand profiles are changing during the optimisation process, and even though there may be considerable communication lags, the problem has turned out to be not so big. The demand profiles provide fairly accurate predictions of the agents' behaviour, if communicated at the beginning and if the constraints do not change externally. The compromise between greedy and altruistic value ordering heuristics may have to be chosen more to the safe side (altruistic) for a distributed implementation.

#### 4.2.3 Scheduling by an asynchronous team of agents

A-Team (Asynchronous Team) was developed at Carnegie Mellon University (Chen et al., 1997). An A-Team is a network of software agents collaborating and exchanging information over shared memory areas. There can be many agents and many shared memory areas, resembling a blackboard architecture. Agents create and manipulate information residing in shared memory areas.

A-Team has been applied in different application domains. In scheduling domain it results to an asynchronous agent-based solver for static job shop scheduling problems based on optimisation heuristics. It focuses on tardiness and inventory costs and a set of agents work together to produce a schedule. Seed agents produce initial complete non-delay schedules employing one of the following dispatch heuristics: shortest processing time, weighted shortest processing time, earliest due date, most work remaining, slack over remaining processing time, weighted cost of time, and apparent tardiness costs. Modification agents swap start times of two adjacent orders or sequences on a machine to create better scheduling solutions. Transfer agents move information among shared memories, the deconstruction agent creates partial schedules, the reconstruction agent rebuild a full schedule. Destroyer agents remove weak schedules.

### 4.3 Agent based manufacturing control

#### 4.3.1 Integrated flow control framework

The integrated flow control framework of Lin & Sohlberg follows the data flow model by Lewis (Lewis et. al., 1987). Under the data flow model, machines select jobs according to a simple dispatching rule (first-in-first-out); jobs are routed to the first machine available to complete the next task to be performed on them. No supervisory control is applied. The integrated flow control framework was established to address highly uncertain and changing

computer controlled manufacturing environment. It employs a market-like system model, where a generic bid construction mechanism based on combination of price and objective mechanism is used, and the negotiation obeys a multiple-way and multiple step negotiation metaphor. It is part-centred, and tries to find critical resources based in dynamic resource unification scheme.

### 4.3.2 Holonic Assembly System

Holonic Assembly System (Silva *et al.*, 1998), (Silva and Ramos, 1999) provides scheduling & resource allocation to assembly tasks through negotiation. The system is composed by different basic entities: Task Manager, Task, Resource and Product. Each of these is potentially composed by multiple entities of its own type. Additionally, these entities are aggregated in functional entities, like the scheduling or the process planning system. The holonic approach is used, which implies a hierarchical organisation while maintaining individual entity autonomy.

As a new task arrives to the system, the Task Manager launches a new Task holon that will be responsible for between resource negotiation and choosing process. The tasks/resources negotiation protocols are fixed and based in constraint propagation. During negotiation a market-based approach is used in order to facilitate and ameliorate the choosing process. The system provides infrastructures for establishing and instantiating a co-operative assembly system and running it flexibly, namely blackboard, broker and pooling service.

### 4.3.3 Holonic Shot Blasting System

A Holonic Shot Blasting System (Rannanjärvi and Heikkilä, 1998), (Heikkilä et. al., 1999) specifies a multi-robot application for surface treatment applications, mainly for shot blasting. It is based on autonomous and co-operative units, i.e. shot blasting robot holons, which are integrated into product

| APPLICATION | Principal subject & topology of communication | Principle of co-ordination | Strategy for co-operation |
|---|---|---|---|
| Agent based manufacturing enterprise integration / MetaMorph II | orders, goals; agent ↔ mediator | Contract negotiation | Finding feasible resources |
| Integration of group of agented manufacturing capabilities / AARIA. | orders, goals; agent ↔ broker | Contract negotiation | Market-based with cost profiles |
| Intelligent assistants to manufacturing. Planners and schedulers / RedPepper | order constraints; agent ↔ user | Constraint propagation | Finding feasible resources |
| Distributed scheduler / ReDS | goals, status; agent ↔ agent | Contract negotiation | Statistics for decision support |
| Micro-opportunistic scheduling / MICRO-BOSS, CORTES | load profiles; agent ↔ agent | Constraint propagation | Local load balance with global consequences |
| Scheduling by an asynchronous team of agents / A-Team | (revised) schedule; agent ↔ agent | Blackboard | Gradual improvement by specialised agents |
| Integrated flow control framework | orders; agent ↔ agent | Contract negotiation | Simple (dispatching) heuristics |
| Holonic Assembly System. | goals; agent ↔ agent | Hierarchy, Negotiation and Contract | Constraint propagation and market based |
| Holonic Assembly System. | goals; agent ↔ agent | Contract negotiation | Market based |
| Holonic Shot Blasting System. | goals, tasks; agent ↔coop. domain | Contract negotiation or blackboard | Gradual composition of mutual plans |
| Handling production changes and disturbances / Mascada. | Pheromones; Agent ↔environment | Distributed blackboard | Emergent social behaviour |

Table 3 – comparison of related work

models. The planning activities of the robots is supported by product models. Task sharing for the robots is carried out through blackboard or contract net based negotiation, including task allocation, detailed motion planning (in time & space) to prevent collision, and sensing planning for accurate locating of the work pieces. The robots are supporting each other by providing sensor information to locate work pieces and by sharing the work space by gradually constructing the motion plans for the shot blasting paths. Human integration is considered of key importance, i.e., human prepares the 'missing elements' to product models; corrects or fine tunes the product models or even the instantiated task descriptions, i.e., paths of the robots motions. The system also supports instantiation and maintenance of the system by flexible management of the system resources (creating and maintaining co-operation domains, holons, etc.).

#### 4.3.4 Handling Production Change and Disturbances

Mascada (MASCADA, URL) is heading to production systems that are robust against changes and disturbances in production as well as in the production systems. The goals of Mascada include building manufacturing control systems out of agents having expertise only on what they are and let functionality emerge, and on the other hand, safeguard performance, e.g., throughput, when disturbances nullify the properties of ordinary schedules.

In the Mascada approach co-operation is achieved by emergent behaviour. The system is constructed of different types of agents (Product, Order, Resource, Staff; PROSA-architecture (Van Brussel et al., 1998). The agents spread information (cf. pheromones) and communicate through the environment. The agents act based on this local information. Actions and decision making is based on flexible rules of the agents, which make them run co-operatively and in a co-ordinated fashion. The rules and the information spreading mechanism focus on certain criteria for the production system, e.g., through put or lead time.

### 4.4 Comparison from the point of view of communication, negotiation, co-ordination and co-operation

Table 3 compares the applications within their characteristics relevant to establish collaboration. For this it is used 'Principal subject & topology of communication', 'Principle of co-ordination ', and 'Strategy for co-operation'.

## 5 Summary

Current practices and newly observed trends lead to the development of new ways of thinking, managing and organising in corporations, where *autonomy*, *decentralisation* and *distribution* are some of the buzzwords. In manufacturing, a new class of software architectures, and organisational models appeared to give form to the Distributed Manufacturing System concept.

As its name says, a DMS has several entities, which must interact with each other to become a system. DMSs were characterised and its behaviours explained in what refers to the whole system and to each component (i.e., entity). DMS's basic properties (i.e., autonomy, distribution, decentralisation, dynamism, and reaction) as well as more complexes ones (i.e., flexibility, adaptability and agility) were presented, explained, and co-related with each other.

From a behavioural point of view, a DMS can engage in several processes (i.e. co-operation, competition, co-ordination, and command) in order to reach a coherent state, a state where conflicts and redundant efforts are avoided.

However, DMSs are abstract, and the above mentioned properties and behaviours were also some what "abstractly" mentioned. Three specific classes of DMSs were presented and compared. These classes (i.e. Fractal Factory, Bionic Manufacturing and Holonic Manufacturing Systems) present organisational models for the corporation as well as guidelines for the development of the system regarding its implementation and expected behaviours.

Agent Technology has proven successful over the last decade in a wide range of applications. The desired properties for an agent are the same listed for an entity in a Distributed Manufacturing System, thus Agent Technology is suitable for building DMSs.

A survey of existing work in Distributed Manufacturing Systems was presented. Several areas of manufacturing are represented in the selected literature especially Supply Chain Management, Scheduling and Control. A comparison from the co-ordination and co-operation perspective was presented.

## Acknowledgement

The first two authors would like to express their gratitude to Professor Carlos Ramos for its valuable review and comments. This work is supported by ESPRIT project n. 21955 – Intelligent Manufacturing Systems Working Group (IMS-WG).

## References

(Agility Forum, URLa) http://www.agilityforum.com

(Agility Forum, URLb) http://www.agility-forum.de/eaf_aim.html

(Austin, 1962) Austin,J.L (1962) *How to do things with words.* Oxford University Press, NY, 1962

(Axelrod, 1984) Axelrod, R. (1984) *The Evolution of Cooperation*. New York: Basic

(Brooks, 1986) Brooks, R.A. (1986) *A robust layered control system for a Mobile Robot*, IEEE Journal of Robotics and Automation, 2(1):14-23, 1986

(Castefranchi, 1995) Castelfranchi, C. (1995). *Guarantees for autonomy in cognitive agent architecture*. In: Wooldridge, M. and Jennings, N. (Eds.) Intelligent Agents: Theories, Architectures, and Languages – Lecture Notes in Artificial Intelligence, vol. 890, pp. 56-70. Springer-Verlag: Heidelberg, Germany.

(Chen et al., 1997) Chen, S. Y, Talukdar, S. N., Sadeh, N. M. (1997): *Job Shop Scheduling by an Asynchronous Team of Optimization Agents*, Carnegie Mellon University

(Cohen, 1988) Cohen,P.R.,Perrault, C.R. (1988) *Elements of a plan-based theory of speech acts.* Alan H. Bond, Les Gasser (Ed.), Readings in Distributed Artificial Intelligence, pp. 169-186, Morgan Kaufman Publishers, San Mateo, CA, 1988.

(ComputerWire, 1997) ComputerWire Plc (1997), *Red Pepper Users Give PeopleSoft a Chilly Reception*, http://www.computerwire.com/ma/free1197.html, 4 p.

(Davis and Smith, 1983) Davis, R. and Smith, R. (1983). *Negotiation as a metaphor for distributed problem solving.* Artificial Intelligence, vol. 20, n. 1, pp. 63-109.

(Finn et al., 1997) Finn, T.; Labrou, Y. and Mayfield, J. (1997) *KQML as an agent communication language*. In: Software Agents. Bradshaw, J. (Ed.) AAAI Press/MIT Press. ISBN 0-262-52234-9.

(Genesereth, 1992) Genesereth, M.R.Fikes, R.E. (1992) *Knowledge Interchange Format Version 3.0 Reference Manual.* Technical Report Logic-92-1, Computer Science Department, Stanford University .

(Gruber, 1993) Gruber, T.R. (1993) *A translation approach to portable Ontologies*, Knowledge Acquisition, 5(2):199-220, 1993

(Hadavi, 1994) Hadavi, K. C., (1994) *ReDS: A Real Time Production Scheduling System.* In Zweben & Fox (ed), Intelligent Scheduling, Morgan Kaufman 1994, pp. 581 - 604

(Heikkilä et. al., 1999) Heikkilä T., Agostino N., Rannanjärvi L., Salonen P., (1999) *Feature-based product modelling for holonic shot blasting systems*. Proceedings of the IEEE/IMACS CCSC Multiconference, Athens, Greece, 4.7.-8.7.1999.

(IDEF URL) Http://www.idef.com

(IMS, 94) Intelligent Manufacturing Systems; A Program for International Cooperation in Advanced Manufacturing; Final Report of the International Steering Committee adopted at ISC6; Hawaii, January 24-26, 1994; URL: http://www.ims.org;

(Koestler, 1967) A. Koestler (1967); *The Ghost in the Machine*. Hutchinson & Co, London, 1967.

(Lin and Solberg, 1992) Lin and Solberg (1992) *Intergrated Shop Floor Control Using Autonomous Agents*. IEE Transactions, Vol. 24, Number 3, July 1992. Pp. 71

(Lewis, 1981) Lewis W.C. Jr., (1981) *Data Flow Architectures for Distributed Control of Computer Operated Manufacturing Systems*. Ph.D. Thesis, School of Industrial Engineering, Purdue University, West Lafayette, IN, USA, May 1981

(Lewis et. al., 1987) Lewis W.C. Jr., Barash M.M., Solberg J.J., (1987) *Computer Integrated Manufacturing System Control A Dataflow Approach*. Journal of Manufacturing Systems, 2 (1987)

(Malone and Croston, 1994) Malone, T. and Crowston, K. (1994) *The Interdisciplinary Study of Co-ordination*. ACM Computing Surveys, Vol. 26, No. 1, March 1994, pp. 87-119.

(MASCADA, URL) Mascada (Esprit LTR 22728): *Manufacturing Systems Capable of Handling Production Changes and Disturbances*. http://www.mech.kuleuven.ac.be/pma/project/mascada.htm

(MW, URLa) Merrian-Webster (URL). *Merrian-Webster Online Dictionary*. http://www.m-w.com/cgi-bin/dictionary?competing

(MW, URLb) Merrian-Webster (URL). *Merrian-Webster Online Dictionary*. http://www.m-w.com/cgi-bin/dictionary?coordinating

(Nwana, 1996) Nwana, H.S., (1996) *Software Agents: An Overview*, Knowledge Engineering Review, Vol. 11, No 3, Oct./Nov. 1996

(Nwana *et al.*, 1996) Nwana, H.; Lee, L. and Jennings, N. (1996) *Coordination in software agent systems*. BT Technology Journal, 14(4), pp. 79-88. October 1996.

(Okino, 1993) Okino, N. (1993) *Bionic Manufacturing System*. In J. Peklenik (ed.) CIRP, Flexible Manufacturing Systems Past-Present-Future, pp. 73-95.

(Osborne and Rubinstein, 1994) Osborne, M. and Rubinstein, A. (1994) *A course in game theory*. MIT Press.

(Parunak, 1998) H. Van Dyke Parunak; (1998) *What can Agents do in Industry, and Why? An Overview of Industrially-Oriented R&D at CEC*; Industrial Technology Institute; CIA'98

(Parunak *et al.*, 1998) Van Dyke Parunak H., Baker A. D., Clark S. J., *The AARIA Agent Architecture: An Example of Requiremens Driven Agent Based System Design*

(Parunak, 1999) Parunak, H. (1999) *Agents while you wait*. Tutorial presented at 4th International Conference on the Practical Application of Intelligent Agents and MultiAgent Technology. London, UK, April 1999.

(PeopleSof, 1997) PeopleSoft Inc., *Optimizing the Assets of the Supply Chain - Today's Manufacturing Challenge*, http://www.peoplesoft.com/products_and_services/enterprise_solutions/redpepper.html, 8 p.

(Pontrandolfo and Okogbaa, 1999) Pontrandolfo, P. and Okogbaa, O. (1999) *Global Manufacturing: a review and a framework for planning in a global corporation*. International Journal of Production Research, vol. 37, n.1, pp.1-19

(Pooley, 1999) Pooley, R., Stevens, P. (1999) *Using UML, Software Engineering with Objects and Components*, Addison Wesley 1999

(Rannanjärvi and Heikkilä, 1998) Rannanjärvi L., Heikkilä T., (1998) *Software Development for Holonic Manufacturing Systems*. Computers in Industry 37 (1998), pp. 233 - 253

(Rao and Georgeff, 1995) Rao, A. and Georgeff, M. (1995) *Bdi agents: From theory to practice.* Tech. Rep. 56, Australian Artificial Intelligence Institute, Melbourne, Australia, April 1995.

(Sadeh, 1994) Sadeh, N. (1994) *Micro-Opportunistic Scheduling: The Micro-Boss Factory Scheduler*. Intelligent Scheduling Published by Morgan Kaufmann, Mark Fox and Monte Zweben (Eds.). 1994.

(Searle, 1969) Searle, J.R. (1969) *Speech Acts: An Essay in the Philosophy of Language.* Cambridge University Press: Cambridge, UK, 1969.

(Shen and Norrie, 1998) Shen W., Norrie D. H., (1998) *A Hybrid Agent-Oriented Infrastructure for Modelling Manufacturing Enterprises*, http://ksi.cpsc.ucalgary.ca/KAW/KAW98/shen/index.html

(Sihn, 1997) Wilfried Sihn (1997); *The Fractal Factory: A Practical Approach to Agility in Manufacturing*; Proceedings of The Second World Congress On Intelligent Manufacturing Processes & Systems, pp. 617-621, Budapest, Hungary, June 10-13, 1997

(Silva, 1998) Nuno Silva; *Sistemas Holónicos de Produção – Especificação e Implementação*; Dissertação de Mestrado; Faculdade de

Engenharia da Universidade do Porto; September 1998

(Silva *et al.*, 1998) Silva N., Sousa P., Ramos C., (1998) *Proposal for a Dynamic Scheduling Architecture and Method Using a Holonic Approach*, Intelligent Manufacturing Systems '98 , 10-12 November 1998, Gramado-RS, Brazil

(Silva and Ramos, 1999) Silva, N. and Ramos, C. (1999) *Holonic dynamic scheduling architecture and services*. International Conference on Enterprise Informstion Systems '99 (ICEIS '99), Setubal, Portugal, 1999

(Sycara, 1989) Sycara, K., (1989) *Multi-agent Compromise via Negotiation*, In L. Gasser & M. Huhns (eds), Distributed Artificial Intelligence 2, Morgan Kaufmann, 1989.

(Sycara et al., 1991) Sycara, K., Roth, S., Sadeh, N. and Fox, M. (1991) *Coordinating Resource Allocation in Distributed Factory Scheduling*, IEEE Expert, Vol. 6, No. 1, pp. 29-40, February, 1991.

(Tharumarajah *et al.*, 1996) A. Tharumarajah, A. J. Welles e L. Nemes; (1996) *Comparison of the bionic, fractal and holonic manufacturing system concepts*; International Journal of Computer Integrated Manufacturing, 1996, vol. 9, nº 3, 217-226

(Valckeneaers *et al.*, 1994) Valcknears, P.; van Brussel, H.; Bonneville, F.; Bongaerts, L. and Wyns, J. (1994) *IMS Test Case 5: Holonic Manufacturing Systems*. IMS workshop at IFAC'94. Vienna, Austria, June 13-15.

(Van Brussel et al., 1998) Van Brussel, H.; Jo Wyns, Paul Valckenaers, Luc Bongaerts, Patrick Peeters; (1998) *Reference Architecture for Holonic Manufacturing Systems: PROSA*; Computers In Industry, vol. 37, pp. 255-274

(Warneke, 1993) Warneke, H. J. (1993) *The Fractal Company*. Springer-Verlag.

(Wooldridge and Jennings, 1995) Wooldridge,M., Jennings,N.R. (1995) *Agent Theories, Architecture and Languages: A Survey*, Lecture Notes in Artificial Intelligence, Vol.890, Springer Verlag 1995

(Zelm, 1995) Martin Zelm, François B. Vernadat, Kurt Kosanke; (1995) *The CIMOSA business modelling process*; Computers in Industry 27, pp. 123-142; 1995