Aspects Of Co-operation in Distributed Manufacturing Systems

Paulo Sousa, Nuno Silva
Instituto Superior de Engenharia do Instituto Politécnico do Porto
Rua Dr. António Bernardino de Almeida, 431
4200-072 Porto
PORTUGAL
E-mail: {psousa | nsilva}@dei.isep.ipp.pt

Tapio Heikkilä
VTT Automation
Oulu
FINLAND
E-mail: Tapio.Heikkila@vtt.fi

Martin Kollingbaum
University of Cambridge
UNITED KINGDOM
E-mail: mjk27@eng.cam.ac.uk

Paul Valckenaers
Katholieke Universiteit Leuven
BELGIUM
Email: Paul.Valckenaers@mech.kuleuven.ac.be

Abstract: It is current practice in manufacturing enterprises the policy of directly controlling 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 business in general, and manufacturing in particular, lead to new approaches regarding the organisation and software architecture, mainly through 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.

Paulo Sousa studied Computer Science at *Instituto Politécnico do Porto* (Polytechnic Institute of Porto, Portugal) - ISEP/IPP from 1990 through 1995 with a specialisation in Industrial Informatics. In 1998, he successfully concluded a post-graduation on "Distributed Systems, Computer Architectures and Computer Communications" at *Universidade do Minho* (University of Minho, Portugal), and started his Ph.D on Holonic Manufacturing Systems. Paulo Sousa worked for 3 years as an application developer for a Portuguese software house in the field of electronic archives, database retrievals and component building. In 1996 he became Assistant Professor at ISEP/IPP. His main research interests are Computer Graphics and Distributed Intelligent Systems. *URL:* http://www.dei.isep.ipp.pt/~psousa

Nuno Silva studied Computer Science with a specialisation in Industrial Informatics at Polytechnic Institute of Porto, Portugal (ISEP/IPP). In 1998 he got his M.Sc degree from the University of Porto regarding Holonic Manufacturing Systems. He has been Assistant Professor at ISEP/IPP since 1995.

Dr. Tapio Heikkilä received the degree of Diploma Engineer in Control Engineering, Licentiate of Technology in Systems and Control Engineering, and Doctor of Technology in Computer Engineering from the University of Oulu, Oulu, Finland in 1983, 1986 and 1991, respectively. From 1983 to 1986 he worked as assistant and researcher at the University of Oulu. From 1986 to 1993 he worked at the Technical Research Centre of Finland (VTT), in Electronics Labory as a scientist, senior scientist and the head of Mechatronic Section. Currently he is chief research scientist at the Automation Institute of VTT . He has also been a guest researcher at the Electrotechnical Laboratory in Tsukuba, Japan, and at the Fraunhofer - Institute for Manufacturing Engineering and Automation (IPA). Since 1996 he has also held a position of Docent of Systems Engineering at the University of Oulu. Dr. Heikkilä's research interests include intelligent manufacturing systems, intelligent robotics, mechatronic systems and multi-agent technologies. Dr. Heikkilä is a member of IEEE, of the Robotics Society of Finland, and the Finnish Automation Society.

Martin Kollingbaum was born in 1962 in Austria. In 1987 he joined the Institute of Flexible Automation, Technical University Vienna and was involved in the development of database technology for telecommunication facilities. After graduating in Computer Science from the Technical University Vienna, between 1990 and 1994 he worked in industry as a software engineer . In 1994 he again joined the Institute of Flexible Automation to be involved in a research project on agent technology in industrial environments. 1997 he worked as a research scholar at the University of South Carolina. Since October 1997 he has been research associate at the University of Cambridge, UK.

Paul Valckenaers received the Applied Mathematics Engineering degree in 1983, the Computer Science Engineering degree in 1985, and the Mechanical Engineering Ph.D degree in 1993, all from the Katholieke Universiteit Leuven, Belgium. Since 1986 he has been with the Mechanical Engineering Department, division PMA, of the Katholieke Universiteit Leuven. His main research interests are in programming, scheduling and control of flexible production systems and design theory for the development of flexible and complex production systems. His current research activities focus on Holonic Manufacturing Systems (HMS) and multi-agent systems.

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.

Three common trends are observed in today's manufacturing context [1] (from the analysis of different opinions):

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

Another observation is the current shift towards the Customised Society [2], where the one-size-fits-all model is replaced by the one-of-a-kind production model. These new trends suggest 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[3].

Computer supported manufacturing systems (e.g. CIM) had been characterised by monolithic structure, centralised control, low flexibility and adaptability, but also efficiency under intensive and repetitive processes. However, as a shift is being made towards the Customised Society, these features no longer represent an added value to the manufacturing enterprise.

It is common sense knowledge that decentralised structures offer better flexibility and adaptability than rigid hierarchical ones. Distribution and decentralisation have also proven its value in computer science fields such as databases, file systems, etc. and become increasingly usual with the advent of the Internet. 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.

Original Computer Integrated Manufacturing (CIM) concept lacks features such as distribution and decentralisation. Hence, Distributed Manufacturing Systems (DMS) are proposed as the next evolutionary step from traditional CIM architectures.

The paper is organised as follows: Section 2 presents Distributed Manufacturing Systems, and their reason of existence, characteristics and behaviours. Section 3 introduces the field of Agent-based computing as a technology for building DMSs. Section 4 presents three different approaches to model Distributed Manufacturing Systems. In Section 5 some related work is presented and compared from the point of view of co-operation. Final remarks can be found in Section 6.

2. Distributed Manufacturing Systems

On 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 become 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, impairs 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 [1]:

- *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 Section 2.2 respectively, DMS properties and behaviours are described and in Section 4 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 to each composing entity. Basic properties include:

- *Autonomy* – An entity is said to be autonomous if it has the ability to operate independently of the rest of the system and if it possesses some kind of control over its actions and internal state [4]. I.e. autonomy is the ability of an entity to create and control the execution of its own plans and/or strategies [5], 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 the manufacturing system's structure and behaviour during operation. This expresses different competencies, responsibilities and relations between entities;
- *Reaction* – An entity is said to be reactive if it adjusts its plans according to its perceptions;

Other properties deserve a thorough description and as such they 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 [1] each one specified as a routine procedure, defined ahead of time so that the needs to manage it are in place[3].

In manufacturing, Flexibility is related to physical flexible machinery. Flexible is in the sense that 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 with companies which 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 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.

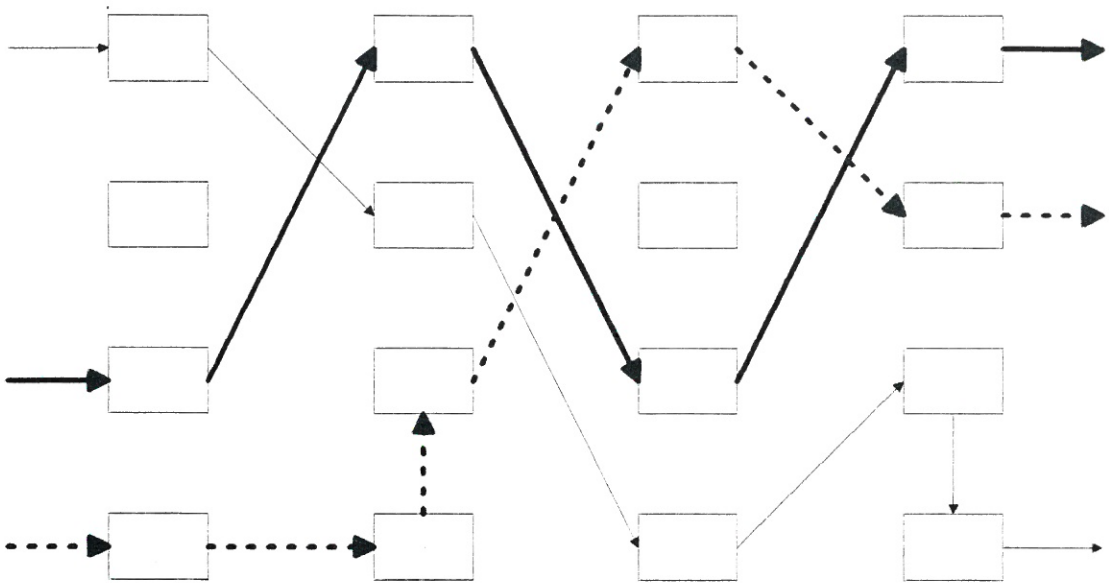

Figure 1. Flexible Manufacturing System

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 [3] 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 occur.

Figure 2 shows the expected cost for developing, installing and running an "adaptable" (bold line) and a "non-adaptable"

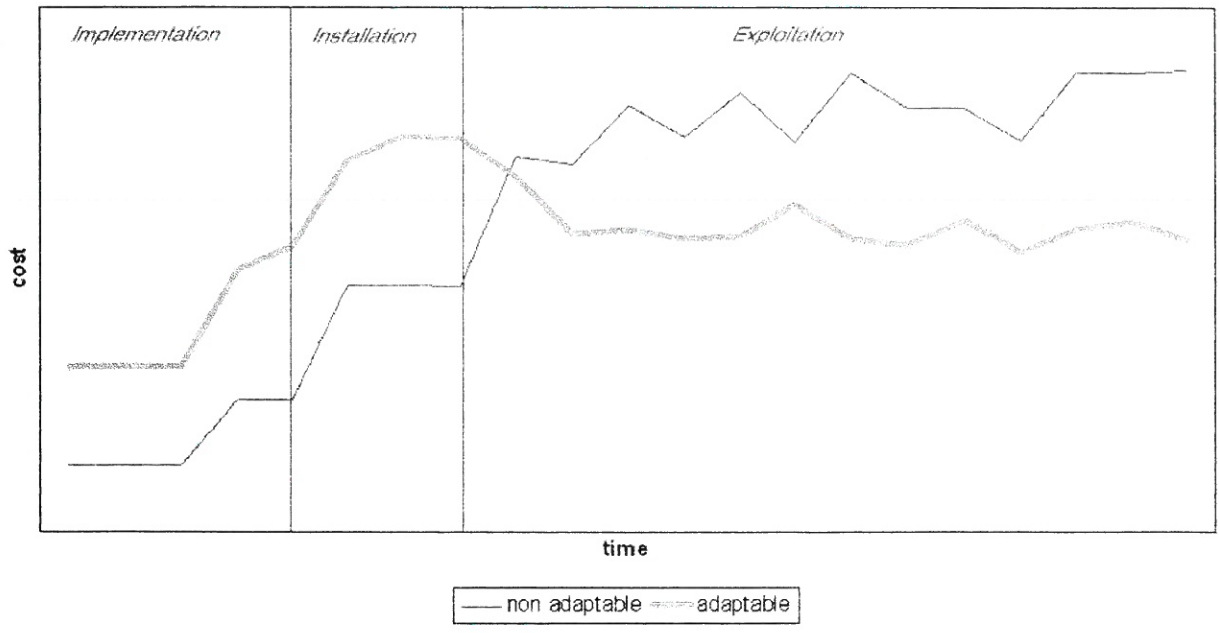

Figure 2 . Expected Cost/Time Plot for Adaptable and Non-adaptable Systems

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–tolerance, reconfiguration, etc. is greater than the one to be invested if the problem were 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's. 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 down-time – idle production units – which, as in automotive industry, is around 5,000 USD per minute [6].

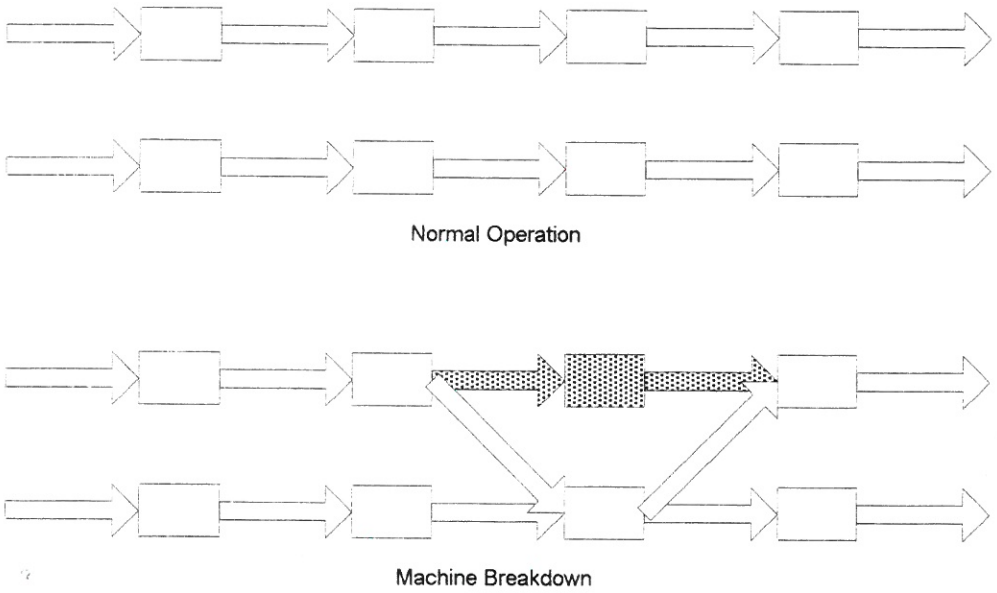

Figure 3. Adaptability needs Flexibility

An important functionality of an adaptable system, related to maintenance and extensibility, is some form of "plug & play". It should be possible 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 a machine has a breakdown 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 [3] 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 to adapt itself and to pursue strategic partnerships in order to prosper in an extremely dynamic and exigent economy.

Agility needs co-operation. According to [7], "To 'Bridge gaps between visible and feasible'

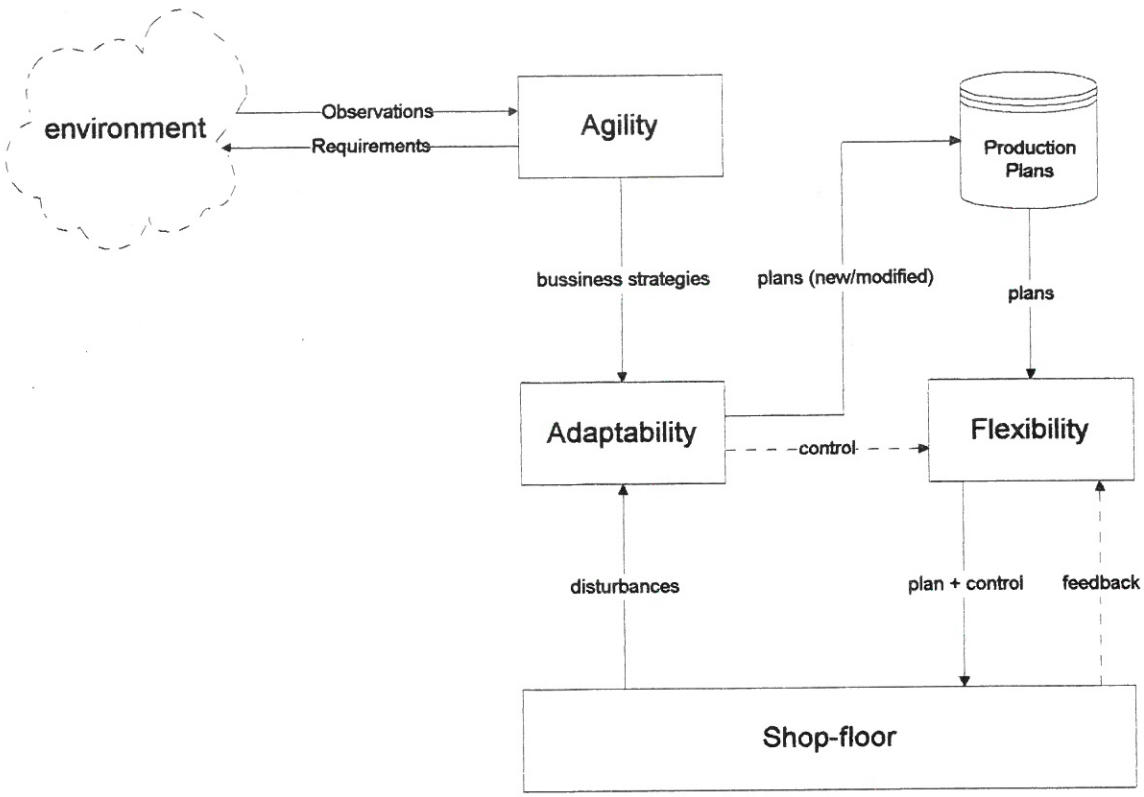

Figure 4. Agility, Adaptability and Flexibility

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 their 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 basis of decision for the download of the correct production plan.

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 making current products, or plans for the production of 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 [8]. A coherent system will minimise or avoid conflicting and redundant efforts among entities [9].

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 [5]. 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 every participant expects to benefit from the co-operation process.
- *Co-ordination* – is the process of managing interdependencies between activities [10]. I.e. "the harmonious functioning of parts for effective results" [11].
- *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" [12].
- *Control or command* – is a process where one entity rules the actions of another (limiting its autonomy). Bradshaw [13] defined control as: *Who* (an agent) does *what* (task/action), *where* (spatial localisation), *when* (temporal localisation) and *with whom* (other agents).

During the system's execution, it is 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 workload.

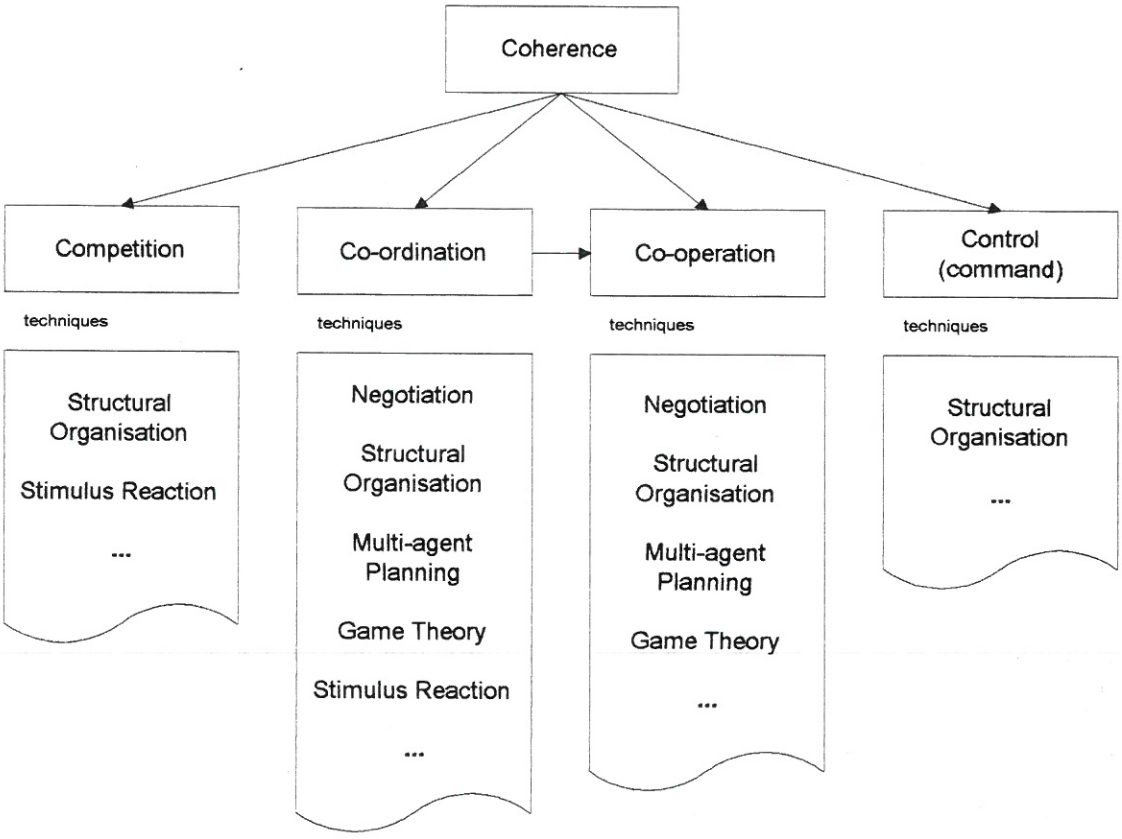

Figure 5. Relation Between Coherence, Co-ordination, Co-operation, Competition and Control

Figure 5 shows the relation among the above-mentioned behaviours. This Figure 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 implicitly defines the agent's responsibilities, capabilities, connectivity and control flow [9]. 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 entity makes from the environment and other entities.
- *Negotiation* – is the effort made by two or more entities to achieve an agreement benefiting them; 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) [14]. E.g. prisoner's dilemma.

For a detailed description of co-ordination issue (and its relation to co-operation, as well as co-ordination techniques) please refer to [9].

2.2.1 Co-operation

One popular technique used in many DMSs is the Contract Net Protocol [15]. 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 begin to co-operate. In addition, refined models of the Contract Net may allow for the proposal and counter-proposal exchange between agents negotiating the best deal for everybody [16]. Another refinement is the Contract Net with Constraint Propagation [17] model where agents compete for operations (sub-tasks), but also co-ordinate their actions with other agents for the fulfilment of dependencies between operations.

As shown above this system exhibits several behaviours (even for a simple model as the original Contract Net is). However, the one that lasts longer is co-operation, hence 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 [17].

- A product needs several operations, and probably these operations are performed on several resources. The job (i.e. the manufacturing order) must, in order to be executed, establish a contract with the needed resources, 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 [18] 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 pay-off rather than its relative gain. Under this hypothesis, provided that its pay-off is positive, an entity does not care if another gets a higher pay-off (implying a negative relative gain) [19].

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 which 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.

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 autonomously acting software entities with capabilities to co-ordinate activities for creating a desired overall system behaviour. Agent technology is especially attractive to application domains like manufacturing environments, because "agents are best suited for applications that are modular, decentralised, changeable, ill-structured, and complex." [20].

In the following, an overview of important aspects of this technology will be taken. For detailed information, please refer to [21] and [22] among many other valuable sources. 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 planning 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) [23]. 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 believe, 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" [24], is an example of 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 so that 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 depend on how single agents choose their course of action. This decision process is based on information, which is available to an agent. It is very common to describe agents exchanging messages and updating their internal models or beliefs about the world and other agents. However, as pointed out 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 broadcasting 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 the following three phases:

i. *Negotiation*: agents negotiate on 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 of a "master" or "manager," which distributes tasks or collects results. In detail, task -sharing itself can be implemented in the 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 [15]; 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 said, 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 sub-tasks for his own task, and starts the negotiation process in order 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 sub-tasks;

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 [25,26,27] 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 the so-called speech acts.

A speech act is determined by 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 [28] 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) [29] 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. [30] 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 [31] can be used to model ontologies. A special modelling language is described by IDEF5 [32] providing a method determined to create ontologies.

4. 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 [33, 34], Bionic Manufacturing Systems [35] and Holonic Manufacturing Systems [36], which were comprised in a broader term designated Open Hierarchical Systems. These paradigms suggest the idea that manufacturing systems will continue to need a hierarchical structure beside the increased autonomy assigned to individual entities. They also advise the hierarchy needed 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.

4.1 Fractal Factory

Fractal Factory paradigm is based on the mathematical fractal concept and the associated theory of chaos. The fractal is characterised by constant evolution will respect to its partners and environment [37]. Warneke [33] 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 adopt the project- orientation* management [34]. 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 combine 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 projects requests different competencies, resources are dynamically associated with projects; thus, each resource may belong to multiple projects yet maintaining its core competencies.

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

The fractal factory must be understood as a non-linear system [34], structurally reactive and adaptive to the dynamic context. Additionally, the concept easily models different enterprise dimension (e.g. strategic, social, cultural, informational and technological) and thus it can easily model enterprise reality in different perspectives.

4. 2 Bionic Manufacturing System

Okino [35] 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 mentions 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 biological systems, autonomy and spontaneous behaviour concern the responsibility for their 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 biological systems and manufacturing systems.

The cell, organ or living being is modelled in BMS by the modelon concept, which is composed of other modelons, to form a hierarchical structure. Each modelon has a very static set of properties and behaviours, which can be combined with others, to form 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 existing 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.

4. 3 Holonic Manufacturing Systems

The Holonic paradigm arises from Herbert Simon's and Arthur Koestler's studies about the biological society evolution and organisation. Simon observed that complex systems are hierarchical systems formed by intermediate stable forms. Later, when analysing Simon's theory and comparing it with its own observations, Koestler perceived that each system and its intermediate forms did not exist as self-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 [38]. 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" [38]. 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 about the environment;
- Each holon is composed of other holons and thus each holon is also a holarchy;
- Each holon can dynamically belong to multiple holarchies;
- The holarchy has fixed rules and directives (the *canon* [37]).

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.*

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 including operational features, individual goals and ability to define its own tasks and execution plans. However, it combines its set of competencies with its lateral partners, with whom it co-operates in order to achieve both individual and system goals.

4. 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 [37], Table 1 resumes the most important structural concepts, while Table 2 refers to the operational behaviours.

The Creation and Moulding 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 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 biological entity, characterised by its birth, living (including reproduction) and death. At the beginning of the

Table 1. Structural Aspects of FF, BMS and HMS

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

Table 2. Dynamic Properties of Fractal Factory, BMS and HMS

		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

system there exists 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, describe respectively the purposes behind the group creation and the applied procedures. FF grouping is project -oriented, that 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 on predefined characteristics, thus it is design specific.

Concerning the Goals parameters, the FF orientates 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.

Summarising, 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 be the most traditional, however easily understandable and structurally adaptable paradigms.

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 *canon*, which imply a limited autonomy in the entity a limited autonomy. The modelon, beside the fact that it is created with predefined goals, is highly dependent 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, that means the system restructures itself to meet project needs. Each fractal engages in negotiation in order to participate in project execution and thus to improve its own goals and performance. In the 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.

Concerning Individual control, Fractal entity, based on vitality characteristic, always pursues 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 a holarchy) as an self-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 flow bottom-up in the hierarchy, which, in turn, will control the sub -hierarchies based on these results.

With respect to Co-operation, two sub-parameters are considered, the event that triggers the co-operation and the process whereby it is accepted between entities. In FF, since the main idea of the paradigm is the project -orientation, as a new project arrives in the system, all the entities will know the fact and engage in negotiations to co-operate. The BMS paradigm relies on 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 , which makes it the preferred approach to design and

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

Apart from 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 also mould internal coherent behaviours and states.

5. Applications and Examples of Co-operation in Distributed Manufacturing Systems

In this Section examples of 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 Section a short comparison between the features and characteristics of these systems, from the point of view of establishing co-operation, is made.

5.1 Supply Chain Management

5.1.1 Agent -based Manufacturing Enterprise Integration

In their MetaMorph I system, [39] 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 on high -level policies to break the decision deadlocks (cf. the 'staff' agent in [40]). Mediator agents can use brokering and recruiting communication mechanisms to find related agents for establishing collaborative sub-systems ('co-ordination clusters' or 'virtual clusters'; cf. 'co-operation domains' in HMS [41]).

In MetaMorph II [39] have applied MetaMorph I framework at 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 to co-ordinate execution of machines, AGV's, etc. Dynamic scheduling and rescheduling are established by the mediation mechanism and Contract Net bidding mechanism. The Machine Mediator selects a machine after having received 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 sub-system), and marketing (integrates customer services).

5.1.2 Integration of A Group of Agented Manufacturing Capabilities

AARIA (Autonomous Agents at Rock Island Arsenal) by [42] 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 [43], including a multi-site co-ordination and control. In fact, AARIA has lately been 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 satisfy him. Further on, each agent in the internal supply chain 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.

5.1.3 Intelligent Assistants to Manufacturing Planners and Schedulers

RedPepper [44] originated in the 80's from the work of Monte Zweben in NASA Ames Research Centre 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 cost for material, machine and labour usage. The considered constraints include promise dates, request dates, inventory shortages, aggregate capacity, safety stocks, excess stocks, and 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 dispatches 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 improved minor actions.

5.2 Distributed Scheduling Systems

5.2.1 Distributed Scheduler

Wishes behind the distributed scheduling system ReDS [45] were not so interested in 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 system more easily, it was essential to subdivide the problem into independently smaller and less complex problems; however, by doing so one would lose 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 statisticians. 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 overcome 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 a 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.

5.2.2 Micro-opportunistic Scheduling

MICRO-BOSS micro-opportunistic scheduling system [46] 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 make 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 bottlenecked in which time periods, and for which operations. Once the decision to be made 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 then look 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 [47], Sycara extends the concept of micro-opportunistic scheduling to distributed scheduling. Agents construct aggregate demand profiles, maintain them and exchange updates of them on a regular basis. As agents make 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 the behaviour of other agents, but this 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.

5.2.3 Scheduling by An Asynchronous Team of Agents

A-Team (Asynchronous Team) was developed at Carnegie Mellon University [48]. 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 in 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 produce better scheduling solutions. Transfer agents move information among shared memories, the deconstruction agent creates partial schedules, and the reconstruction agent rebuilds a full schedule. Destroyer agents remove weak schedules.

5.3 Agent -based Manufacturing Control

5.3.1 Integrated Flow Control Framework

The integrated flow control framework of Lin and Solberg [49] follows the data flow model of Lewis' [50, 51]. 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 a 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 on a dynamic resource unification scheme.

5.3.2 Holonic Assembly System

Holonic Assembly System [52, 53] provides scheduling & resource allocation to assembly tasks through negotiation. The system is composed of different basic entities: Task Manager, Task, Resource and Product. Each of these is potentially composed of 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 on 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 for running it flexibly, namely blackboard, broker and pooling service.

5.3.3 Holonic Shot Blasting System

A Holonic Shot Blasting System [41,54] 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 models. The planning activities of the robots are 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 and space) to prevent collision, and sensing planning for accurate location of the work pieces. The robots are supporting each other by providing sensor information to locate work pieces and by sharing the workspace to gradually construct 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.).

5.3.4 Handling Production Change and Disturbances

Mascada [43] is heading to production systems that are robust against changes and disturbances in production as well as in 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 [40]). The agents spread information (cf. pheromones) and communicate through the environment. The agents act based on this local information. Actions and decision -making are 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. throughput or lead time.

5.4 Comparison From the Point of View of Communication, Negotiation, Co-ordination and Co-operation

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

There are two extreme approaches concerning the similarity of agents in a multi-agent system, and these are also represented in the cases of Table 3. Some concluding remarks are given in the following.

The system may be composed of homogeneous agents communicating directly with each other. The agents represent typically the manufacturing resources (cells, stations, robots, AGVs, etc.).

in the communication link between the agents. In this case both the structures of the agents and the co-ordination protocols can be very simple. The drawback lies in that there appears to be more failure points in the system.

Table 3. Comparison of Related Work

APPLICATION	Principal subject & topology of communication	Principle of co-ordination	Strategy for co-operation
MetaMorph II	orders, goals; agent ↔ mediator	Contract negotiation	Finding feasible resources
AARIA	orders, goals; agent ↔ broker	Contract negotiation	Market-based with cost profiles
RedPepper	order constraints; agent ↔ user	Constraint propagation	Finding feasible resources
ReDS	goals, status; agent ↔ agent	Contract negotiation	Statistics for decision support
MICRO-BOSS, CORTES	load profiles; agent ↔ agent	Constraint propagation	Local load balance w/ global consequences
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 Shot Blasting System	goals, tasks; agent ↔ coop. domain	Contract negotiation or blackboard	Gradual composition of mutual plans
Mascada	Pheromones; Agent ↔ environment	Distributed blackboard	Emergent social behaviour

In principle the system can be very robust against device failures and changes in the agent system, because there are (depending on the communication protocols) no single-point-of-failures in the communication links. However, the agents tend to be then rather complicated (the agents need to do 'everything'), unless the application is focusing on a limited area, like scheduling. Communication is typically carried out following the Speech-Act techniques, for example with variants of the Contract Net protocol. In addition, the communication load may become a burden because of wide acquaintances between the agents.

The other approach is represented by systems of heterogeneous agents, with a set of specialized agent types. This streamlines and simplifies the internal agent structures. A nice example from this category is the PROSA architecture by van Brussel et al, where agents are specialized to product, resource, order and staff agents. More extreme examples have further specialization of agents for co-ordination purposes, like mediators, brokers, or co-operation domains, which form an essential part

Co-ordination and co-operation are closely related to the high -level communication protocol. With contract net a market-based approach is typically applied, based on satisfying cost constraints, or to some extent, on optimising global or local costs. In the case of behavioural agents, like in Mascada, the communication follows a blackboard principle, either directly, in an agent-to-agent manner, or even indirectly, via the environment. Again, the agents can be simple, and the system can react quickly to changes and disturbances, because the robustness is an in-built feature. However, the optimisation can be a hard problem, and a better performance goal is then to maintain the performance at an acceptable level rather than try to optimise it.

6. 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 their behaviour was explained in what concerns the whole system and each of its components (i.e. entity). DMS's basic properties (i.e. autonomy, distribution, decentralisation, dynamism, and reaction) as well as more complex ones (i.e. flexibility, adaptability and agility) were presented, explained, and correlated 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 somewhat "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 as those 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.

Acknowledgment

The first two authors would like to express their gratitude to Professor Carlos Ramos for his valuable review and comments. This work has been supported by ESPRIT project no. 21955 – Intelligent Manufacturing Systems -Working Group (IMS-WG).

REFERENCES

1. SILVA, N., **Sistemas Holónicos de Produção – Especificação e Implementação**; Dissertação de Mestrado; Faculdade de Engenharia da Universidade do Porto, September 1998.

2. COX, W. and ALM, R., **The Right Stuff: America's Move to Mass Customization**, Federal Reserve Bank of Dallas, Annual Report, 1998.

3. http://www.agilityforum.com

4. CASTELFRANCHI, C., **Guarantees for Autonomy in Cognitive Agent Architecture**, in M. Wooldridge and N. Jennings (Eds.) Intelligent Agents: Theories, Architectures, and Languages, Lecture Notes in Artificial Intelligence, Vol. 890, SPRINGER-VERLAG, Heidelberg, 1995, pp. 56-70.

5. VALCKENAERS, P., VAN BRUSSEL, H., BONNEVILLE, F., BONGAERTS, L. and WYNS, J., **IMS Test Case 5: Holonic Manufacturing Systems**, Proceedings of the IMS Workshop at IFAC'94, Vienna, June 13-15, 1994.

6. PARUNAK, H., **Agents While You Wait**, A Tutorial presented at the 4th International Conference on the Practical Application of Intelligent Agents and Multi Agent Technology, London, April 1999.

7. http://www.agility-forum.de/eaf_aim.html

8. SYCARA, K., **Multi-Agent Compromise via Negotiation**, in L. Gasser and M. Huhns (Eds.) Distributed Artificial Intelligence 2, MORGAN KAUFMANN, 1989.

9. NWANA, H., LEE, L. and JENNINGS, N., **Coordination in Software Agent Systems**, BT TECHNOLOGY JOURNAL, 14(4), October 1996, pp.79-88.

10. MALONE, T. and CROWSTON, K., **The Interdisciplinary Study of Co-ordination**, ACM COMPUTING SURVEYS, Vol. 26(1), March 1994, pp. 87-119.

11. MERRIAN-WEBSTER (URLa), **Coordinating Merrian-Webster Online Dictionary**. http://www.m-w.com/cgi-bin/dictionary?coordinating

12. MERRIAN-WEBSTER (URLb). **Competing Merrian-Webster Online Dictionary**. http://www.m-w.com/cgi bin/dictionary?competing

13. BRADSHAW, J., **Software Agents: The Next Generation**, a tutorial presented at the 2nd International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'97), London, 21-23 April 1997.

14. OSBORNE, M. and RUBINSTEIN, A., **A Course in Game Theory**, MIT PRESS, 1994.

15. DAVIS, R. and SMITH, R., **Negotiation As A Metaphor for Distributed Problem Solving**, ARTIFICIAL INTELLIGENCE, Vol. 20(1), 1983, pp. 63-109.

16. SYCARA, K., **Negotiation Planning: An AI Approach**, EUROPEAN JOURNAL OF OPERATIONAL RESEARCH, Vol. 46, 1990, pp. 216-234.

17. SOUSA, P. and RAMOS, C., **A Distributed Architecture and Negotiation Protocol for Scheduling in Manufacturing Systems**, COMPUTERS IN INDUSTRY – Special Issue on Life Cycle Approaches to Production Systems: Management, Control and Supervision, Vol. 38(2), ELSEVIER, March 1999, pp.103-113.

18. AXELROD, R., **The Evolution of Cooperation**, New York, 1984.

19. PONTRANDOLFO, P. and OKOGBAA, O., **Global Manufacturing: A Review and A Framework for Planning in A Global Corporation**, INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH, Vol. 37(1), 1999, pp.1-19.

20. PARUNAK, H., **What Can Agents Do in Industry, and Why? An Overview of Industrially-Oriented R&D at CEC**, Industrial Technology Institute, CIA'98, 1998.

21. WOOLDRIDGE, M. and JENNINGS, N., **Agent Theories, Architecture and Languages: A Survey**, Lecture Notes in Artificial Intelligence, Vol. 890, SPRINGER-VERLAG, 1995.

22. NWANA, H., **Software Agents: An Overview**, KNOWLEDGE ENGINEERING REVIEW, Vol. 11(3), October/November 1996.

23. RAO, A. and GEORGEFF, M., **BDI Agents: From Theory to Practice**, Technical Report 56, Australian Artificial Intelligence Institute, Melbourne, April 1995.

24. BROOKS, R. A., **A Robust Layered Control System for A Mobile Robot**, IEEE JOURNAL OF ROBOTICS AND AUTOMATION, 2(1), 1986, pp. 14-23

25. AUSTIN, J., **How To Do Things With Words**, OXFORD UNIVERSITY PRESS, NY, 1962.

26. COHEN, P. and PERRAULT, C., **Elements of A Plan-Based Theory of Speech Acts**, in A. H. Bond and L. Gasser (Eds.) Readings in Distributed Artificial Intelligence, MORGAN KAUFMANN PUBLISHERS, San Mateo, CA, 1988. pp. 169-186.

27. SEARLE, J., **Speech Acts: An Essay in the Philosophy of Language**, CAMBRIDGE UNIVERSITY PRESS, Cambridge, 1969.

28. FINN, T., LABROU, Y. and MAYFIELD, J., **KQML As An Agent Communication Language**, in J. Bradshaw (Ed.) Software Agents, AAAI PRESS/MIT PRESS, 1997.

29. GENESERETH, M. and FIKES, R., **Knowledge Interchange Format Version 3.0 Reference Manual**, Technical Report Logic-92-1, Computer Science Department, Stanford University , CA, 1992.

30. GRUBER, T., **A Translation Approach to Portable Ontologies**, KNOWLEDGE ACQUISITION, 5(2), 1993, pp. 199-220.

31. POOLEY, R. and STEVENS, P., **Using UML, Software Engineering With Objects and Components**, ADDISON-WESLEY, 1999.

32. http://www.idef.com

33. WARNEKE, H. J., **The Fractal Company**, SPRINGER-VERLAG, 1993.

34. SIHN, W., **The Fractal Factory: A Practical Approach to Agility in Manufacturing**, Proceedings of the 2nd World Congress on Intelligent Manufacturing Processes and Systems, Budapest, June 10-13, 1997, pp. 617-621.

35. OKINO, N., **Bionic Manufacturing System**, in J. Peklenik (Ed.) CIRP, Flexible Manufacturing Systems: Past-Present-Future, 1993, pp. 73-95.

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

37. THARUMARAJAH, WELLES, J. and NEMES, E. , **Comparison of the Bionic, Fractal and Holonic Manufacturing System Concepts**, INTERNATIONAL JOURNAL OF COMPUTER INTEGRATED MANUFACTURING, Vol. 9(3), 1996, pp. 217-226.

38. KOESTLER, A., **The Ghost in the Machine**, HUTCHINSON & CO, London, 1967.

39. SHEN, W. and NORRIE, D., **A Hybrid Agent-Oriented Infrastructure for Modelling Manufacturing Enterprises.** http://ksi.cpsc.ucalgary.ca/KAW/KAW98/shen/index.html

40. VAN BRUSSEL, H., WYNS, J., VALCKENAERS, P., BONGAERTS, L. and PEETERS, P., **Reference Architecture for Holonic Manufacturing Systems: PROSA**, COMPUTERS IN INDUSTRY, Vol. 37, 1998, pp. 255-274.

41. RANNANJÄRVI, L. and HEIKKILÄ, T., **Software Development for Holonic Manufacturing Systems**, COMPUTERS IN INDUSTRY, 37, 1998, pp. 233 - 253.

42. PARUNAK, H., BAKER, A. and CLARK, S., **The AARIA Agent Architecture: An Example of Requirements Driven Agent Based System Design,** 1998.

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

44. PEOPLESOFT INC. (URL) **Optimizing the Assets of the Supply Chain: Today's Manufacturing Challenge** http://www.peoplesoft.com/products_and_services/enterprise_solutions/redpepper.html

45. HADAVI, K., **ReDS: A Real Time Production Scheduling System**, in M. Zweben and M. Fox (Eds.) Intelligent Scheduling, MORGAN KAUFMANN PUBLISHERS, 1994, pp. 581 - 604.

46. SADEH, N., **Micro-Opportunistic Scheduling: The Micro-Boss Factory Scheduler**, in M. Zweben and M. Fox (Eds.) Intelligent Scheduling, MORGAN KAUFMANN PUBLISHERS, 1994.

47. SYCARA, K., ROTH, S., SADEH, N. and FOX, M., **Coordinating Resource Allocation in Distributed Factory Scheduling**, IEEE EXPERT, Vol. 6(1), February 1991, pp.29-40.

48. CHEN, S. Y., TALUKDAR, S. N. and SADEH, N. M., **Job Shop Scheduling By An Asynchronous Team of Optimization Agents**, Carnegie -Mellon University, 1997.

49. LIN and SOLBERG, J., **Integrated Shop Floor Control Using Autonomous Agents**, IEE TRANSACTIONS, Vol. 24(3), July 1992.

50. LEWIS, W., **Data Flow Architectures for Distributed Control of Computer Operated Manufacturing Systems**, Ph. D Thesis, School of Industrial Engineering, Purdue University, West Lafayette, IN, May 1981.

51. LEWIS, W., BARASH, M. and SOLBERG, J., **Computer Integrated Manufacturing System Control: A Dataflow Approach**, JOURNAL OF MANUFACTURING SYSTEMS, Vol. 2, 1987.

52. SILVA, N., SOUSA, P. and RAMOS, C., **Proposal for A Dynamic Scheduling Architecture and Method Using A Holonic Approach,** Proceedings of Intelligent Manufacturing Systems '98, Gramado-RS, Brazil, 10-12 November 1998.

53. SILVA, N. and RAMOS, C., **Holonic Dynamic Scheduling Architecture and Services**, Proceedings of International Conference on Enterprise Information Systems '99 (ICEIS '99), Setubal, 1999.

54. HEIKKILÄ, T., AGOSTINO, N., RANNANJÄRVI, L. and SALONEN, P., **Feature-Based Product Modelling for Holonic Shot Blasting Systems**, Proceedings of the IEEE/IMACS CCSC Multiconference, Athens, 4-8.July 1999.