Computers in Industry 60 (2009) 428–440

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.elsevier.com/locate/compind

Engineering Holonic Manufacturing Systems

Adriana Giret *, Vicente Botti

Universidad Politécnica de Valencia, Departamento de Sistemas Informáticos y Computación, Camino de Vera s/n-46022, Valencia, Spain

ARTICLE INFO

Article history:
Available online 20 March 2009

Keywords:
Multi-agent Systems
Holonic Manufacturing Systems
Software engineering method

ABSTRACT

Holonic Manufacturing Systems (HMS) was first proposed as a new manufacturing paradigm at the beginning of 1990 and has subsequently received a lot of attention in academic and industrial research. The application of holonic concepts to manufacturing was initially motivated by the inadequacy of existing manufacturing systems in the following two aspects: (i) dealing with the evolution of products within an existing production facility, and (ii) maintaining a satisfactory performance outside of normal operating conditions. In spite of the large number of developments reported in Holonic Control Architectures and Holonic Control Algorithms, there is very little work reported on Methodologies for HMS. Manufacturing requirements impose significant properties on HMS making the modelling process of HMS a complex and difficult task. In this work we describe how to engineer Holonic Manufacturing Systems using a Multi-agent System method. Our approach is based on holonic systems specific requirements, and it incorporates software engineering principles in order to assist the system designers at each development stage. It implements a collaborative-engineering process for HMS, and provides clear and unambiguous analysis and design guidelines.

1. Introduction

The manufacturing sector is currently facing a fundamental change from a seller's market to a buyer's market [1]. Competition has intensified from a national scale to a global arena [2], product life cycles have shrunk, yet there is an escalating requirement to satisfy the specific and individual needs of customers. The manufacturer's success is not only measured by their ability to cost-effectively produce a single product, but also by flexibility, agility and versatility. The changes in markets, customer requirements, and technology have become the competition criteria. These rapid environmental changes have forced companies to improve their manufacturing performance in conditions of increasing uncertainty. In order to survive, manufacturing systems need to adapt themselves at an ever-increasing pace to incorporate new technologies, new products, new organizational structures, etc.

The above trends have motivated researchers in academia and industry to create and exploit new production paradigms on the basis of autonomy and cooperation because both concepts are necessary for creating flexible behaviour and thus to adapt to the changing production conditions. Such technologies provide a natural way to overcome such problems, and design and implement distributed intelligent manufacturing environments [1]. Distributed intelligent manufacturing systems, and Holonic Manufacturing Systems are considered important approaches for developing industrial distributed systems.

In this work we discuss the engineering of Holonic Manufacturing Systems (HMS) with ANEMONA. ANEMONA is a software engineering method that has merged modelling concepts from distributed intelligent system development and the specific modelling requirements of Holonic Manufacturing Systems. Section 2 overviews the insights of Intelligent Manufacturing Systems, focusing on HMS and agent-based manufacturing. Section 3 discusses the complexity of modelling HMS. The relationships between the agent and holonic approaches are analyzed in Section 4. The features of ANEMONA are presented in Section 5. Section 6 summarizes a simplified real case study. Finally, Section 7 details the conclusions and future works.

2. Intelligent manufacturing systems

It is well known that manufacturing systems are large-scale and complex systems for a number of operational and structural reasons. This complexity makes such systems difficult to control and predict. Moreover, in order to meet the challenges of the "new manufacturing" these systems will need to satisfy fundamental requirements such as [3,1]: Enterprise Integration, Distributed Organization, Heterogeneous Environments, Interoperability, Open and Dynamic Structure, Cooperation, Integration of humans with software and hardware, Agility, Scalability, and Fault Tolerance.

* Corresponding author. Tel.: +34 963877000; fax: +34 963877359.
E-mail address: agiret@dsic.upv.es (A. Giret).
URL: http://www.dsic.upv.es/_agiret

0166-3615/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.compind.2009.02.007

Many manufacturing paradigms promise to meet these challenges. Two of these paradigms, Distributed Intelligent Manufacturing Systems, and Holonic Manufacturing Systems have recently been attracting a lot of attention in academia and industry.

Techniques from Artificial Intelligence have already been used in Intelligent Manufacturing for more than twenty years [1]. However, recent developments in Multi-agent Systems in the domain of Distributed Artificial Intelligence have brought to light new and interesting possibilities. Distributed Intelligent Manufacturing Systems, or agent-based manufacturing systems, are based on Multi-agent System (MAS) technology [4]. MAS studies the coordination of intelligent behaviours among a group of (possibly pre-existing) agents [4]. Today MAS is a very active area of research and is beginning to have commercial and industrial applications.

Holonic Manufacturing is based on the concept of "holonic systems", developed by Koestler [5]. Koestler proposed the word holon to describe the hybrid nature of sub-wholes/parts in real-life systems; holons are simultaneously self-contained wholes to their subordinated parts, and dependent parts when seen from the inverse direction.

Koestler also pointed out that holons are autonomous self-reliant units, which have a degree of independence and handle contingencies without asking higher authorities for instructions. A holarchy is a hierarchy of self-regulating holons, which function (a) as autonomous wholes in supra-ordination to their parts, (b) as dependent parts in sub-ordination to controls on higher levels, and (c) in coordination with their local environment.

Work in the HMS program has translated these concepts to the manufacturing world, viewing the manufacturing system as consisting of autonomous modules (holons) with distributed control. The HMS concept combines the best features of hierarchical and heterarchical organization [6]. It preserves the stability of hierarchy while providing the dynamic flexibility of heterarchy. In a HMS a holon is an autonomous and co-operative manufacturing system building block for transforming, transporting, storing and/or validating information and physical objects. The holon consists of an information processing part and often a physical processing part [7]. A holon can be part of another holon.

In the last ten years, an increasing amount of research has been devoted to HMS over a broad range of both theoretical issues and industrial applications. McFarlane and Bussmann have divided these research efforts into three groups [8]: Holonic Control Architectures, Holonic Control Algorithms, and Methodologies for HMS. In spite of the large number of developments reported in the first two areas (for a detailed study see [8]), there is very little work about Methodologies for HMS. There is a definite need to have methodologies for holonic systems [8] that are based on software engineering principles in order to assist the system designer at each stage of development. This methodology should provide clear, unambiguous analysis and design guidelines.

3. Modelling Holonic Manufacturing Systems

Manufacturing requirements impose significant properties on HMS [3] making the modelling process of HMS a complex and difficult task. These properties define functional attributes and specific requirements for the HMS structure and the HMS development process that must be considered in an engineering method. The main modelling requirements are as follows.

A HMS should integrate the entire range of manufacturing activities (from order booking through design, production, and marketing) to model the agile manufacturing enterprise [3].

The association and organization relations defined in a holarchy structure require holonic entities to be organized in hierarchy and heterarchy structures [3]. These types of organizational structures are also called temporal relaxed hierarchies. The holarchy structure implements the benefits of both hierarchy and heterarchy structures in manufacturing control systems. That is, determinisms of hierarchies and flexibility of heterarchies.

Manufacturing control units require a routine-based behaviour that is both effective and timely [9]. To overcome the limitations of the current approaches to manufacturing control, the control and planning processes must be distributed in a holarchy structure. Each unit must have the freedom to choose the right actions at the right moment. Such, a goal-driven and co-operative, approach places new requirements on the design of manufacturing systems [8]. The control should be distributed to the components of the manufacturing system instead of being divided into central control functions. A local controller must be equipped with reactive and goal-directed decision-making capabilities. Moreover, it must co-operate in a flexible manner. Thus, a Methodology for HMS should lead directly to autonomous cooperating entities from the control task on a factory resource or factory function [9,3].

Manufacturing systems are large-scale complex systems. In order to manage this intricate and huge structure a Methodology for HMS should define a development process that is guided by abstraction levels. The engineering method should also provide modelling artifacts, tools and guidelines for managing this process. Moreover, the large-scale nature of these systems demands a collaborative-engineering approach. The holonic collaborative-engineering process is achieved by different levels of abstractions (holarchies) in which each level may be developed by different development groups in a concurrent fashion.

The concepts of fractal and holonic system design in manufacturing were proposed to combine top-down hierarchical organizational structure with decentralized control, which takes the bottom-up perspective [10,7]. Although it is possible to organize holonic structures in a completely decentralized manner, for efficiency reasons it is more effective to use a holarchy organizational approach. These organizational requirements have to be assured by the engineering method. To this end a holarchy-oriented development process should be guided by a mixed top-down and bottom-up approach.

4. Multi-agent systems and holonic systems

In the HMS research field there are successful applications for manufacturing enterprise integration and supply chain management, manufacturing planning, scheduling and execution control, materials handling and inventory management, among others [1,11]. Analyzing these applications we can deduce that, due to the lack of specific methods for implementing holonic systems, multi-agent technology is the most used tool for mainstream HMS research. Agent technology is used to implement HMS applications because the two approaches are not much different. In fact holons and agents are very similar concepts (for a detailed comparison of these two notions see [11]). For instance in [12] the authors present a "Holonic Multi-agent System" definition which is useful as a formalization of organizational structures.

Multi-agent systems represent a new problem-solving paradigm, where the difficult specification at design time of how a problem should be solved, is answered by the interaction of the individual agents at run-time, the idea being that the solution of a given problem emerges from this interaction. Many distributed problems exhibit an inherent structure and we need to mirror this structure in the structure of the relationship between (agentified) problem solvers. In [11], we pointed out that the recursive structure is the only holon property that is not presented as such in the agent definition. To cope with this limitation, in [13] we have proposed the Abstract Agent notion as a modelling artifact for

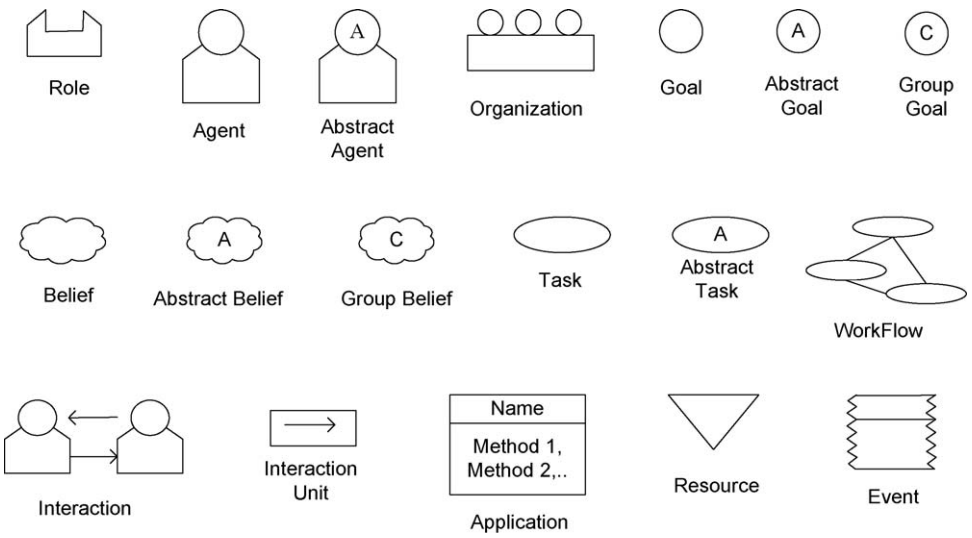

Fig. 1. Some graphical notations of ANEMONA.

autonomous entities with recursive structures. The Abstract Agent extends the traditional agent definition adding a structural perspective to the agent concept: "... *an Abstract Agent can be an agent*; *or it can be a MAS made up of Abstract Agents* ..."

The Abstract Agent is an attempt to unify the concepts of holons and agents and to close the gap between holons and agents in the analysis and design steps. This will make it easer to translate the modelling products that are obtained from Methodologies for HMS into coding elements for the implementation of the holonic system. In our approach for HMS modelling, a holon (Abstract Agent) that appears as a single entity to the outside world may in fact be composed of many Abstract Agents and conversely, many Abstract Agents may decide that it is advantageous to join the coherent structure of a holon (of a higher level of abstraction) and thus act as single entity.

MAS methodologies may be good candidates for developing HMS due to the similarities of the two approaches. However, there are some extensions that must be included in a MAS methodology to be able to model the HMS requirements in a proper way: holon recursive structure, system abstraction levels, HMS-specific guidelines, and a mixed top-down and bottom-up development approach. To this end, in this work we present the HMS-specific features of ANEMONA. ANEMONA is a MAS methodology for Intelligent Manufacturing Systems. Our approach attempts to satisfy the requirements analyzed in Section 3.

5. Engineering Holonic Manufacturing Systems with ANEMONA

ANEMONA is a MAS methodology for HMS analysis and design, based on the Abstract Agent notion and the HMS modelling requirements. ANEMONA integrates features from HMS, MAS and Enterprise Modelling techniques [16,17].

In ANEMONA, the HMS is specified, by dividing into more specific characteristics that form different views of the system. We use Abstract Agent and holon as similar notions [11]. The views can be considered general MAS models which can also be applied to other domains. The way in which the views (models) are defined [18,19] is inspired by the INGENIAS [14] and the RT-MESSAGE [15] methodologies, bearing in mind the requirements analyzed in Section 3.

There are five views, or models. The *agent model* is concerned with the functionality of each Abstract Agent: responsibilities and capabilities. The *organization model* describes how system components (Abstract Agents, roles, resources, and applications) are grouped together. The *interaction model* addresses the exchange of information or requests between Abstract Agents. The *environment model* defines the non-autonomous entities with which the Abstract Agents interacts. The *task/goal model* describes relationships among goals and tasks, goal structures, and task structures. Fig. 1 shows some graphical notations of ANEMONA.

In Fig. 2 we can see the development stages of ANEMONA.[1] The first stage, *System Requirement Analysis* and the second stage *Holon Identification and Specification* define the analysis phase. The aim of the analysis phase is to provide high-level HMS specifications from the problem *Requirements*, which are specified with the help of the problem domain experts and which can be updated at any stage of the development. The analysis adopts a top-down recursive approach. One advantage of a recursive analysis is that its results, i.e. the *Analysis Models*, provide a set of elementary elements and assembling rules. The next stage in the development process is the *Holon Design* stage, which is a bottom-up process to produce the *System Architecture* from the *Analysis Models* of the previous stage. The aim of the *Holon Implementation* stage is to produce an *Executable Code for the SetUp and Configuration* stage. Finally maintenance functions are executed in the *Operation and Maintenance* stage.

In the following sections the different phases of ANEMONA are presented.

5.1. Problem specification and requirements

The *Requirements* specification describes the manufacturing system's requirements and its associated control problems. This document may have any structure; it may use natural language and/or a more formal specification notation such as Finite State Machine, Petri nets, or IDEF0 [21]. ANEMONA suggests the use of the following items to specify the system's requirements: a description of the organizational chart of the company in which the system will be integrated, the business processes of the company, the intended scope of the system in the company, the processes to be controlled, a specification of the operation conditions, and a specification of the system goals.

[1] The specification of the development process of ANEMONA is presented using SPEM diagrams [20].

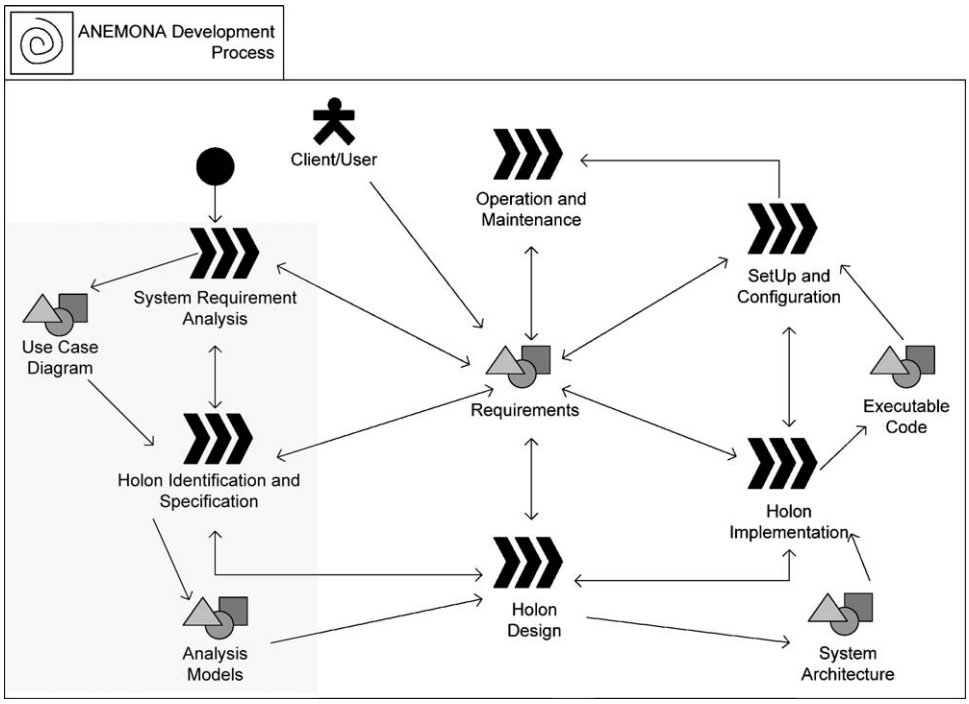

Fig. 2. Stages of the methodology.

5.2. Analysis

The work-flow presented in (Fig. 3) defines the ANEMONA *Analysis* phase. The HMS is specified in terms of the five models defined in ANEMONA [18] and UML *Use Case Diagrams*. The main goal of the analysis phase is to identify the constituent holons and to provide an initial holon specification. Since a manufacturing system is a complex and a large-scale problem, this process is conducted in an incremental and collaborative fashion, using different abstraction levels. Each iteration of the analysis phase identifies and specifies holarchies of different levels of recursion (holons made up of holons). The first iteration identifies an initial holarchy, which is made up of holons that cooperate to fulfil the global system requirements. At the end of every iteration, the design team must analyze each holon in order to figure out the advantages of decomposing it into a new holarchy. In this way, each new iteration will have as many concurrent and collaborative processes as constituent holons of the previous iteration which it was decided would be decomposed. This process is repeated until every holon is defined and there is no need for further decompositions.

The analysis phase follows a rapid iterative approach in order to find out a first holonic structure in every abstraction layer. This approach reduces excessive time invested in the analysis of complex and large-scale problems. Moreover, it rapidly configures the starting point and the requirement specification of the different concurrent and collaborative analysis processes for every design team.

The first step in the analysis phase (see Fig. 3) is to *Determine Use Cases* by building a *Use Case Model* from the system *Requirements*. The Use Cases are identified following the *HMS UC Guidelines* [22] of ANEMONA to identify cooperation domains and system goals as Use Cases. Uses Cases can be considered simpler sub-problems that, taken together, define the entire system. In subsequent iterations the cooperation domains may be decomposed into new ones. There are two groups of *HMS UC Guidelines*: the system goal identification guidelines and the use cases guidelines. The first group is made up of 6 questions. Answering these questions the engineer can identify the system goals. On the other hand, the second group of guidelines helps the engineer to identify and specify the use cases of the HMS in order to fulfil the system goals. The complete list of the *HMS UC Guidelines* can be found in [22].

In the second analysis step, *Specify Use Case Realization*, the Use Cases of the previous step are analyzed. Every Use Case is associated with a supplier of its functionality: a *Role*. The interaction and relationships among the Use Cases are modeled by building *Interaction Models* and *Organization Models*.

In the third step, *Identify Holons*, the designer works with the work products of the previous step, the system *Requirements*, and the *PROSA Guidelines* [22] to identify any new Abstract Agent and to categorize the identified Abstract Agents. The *PROSA Guidelines* are defined based on PROSA types of holons [23]. The PROSA reference architecture specifies that there are four types of holons in any manufacturing system: work order holon, product holon, resource holon and staff holon. Based on these types of holons the *PROSA Guidelines* propose 28 guides (the complete list is available at [22]) to identify and specify Abstract Agents from the *Use Case Diagrams*, the *Organization Diagrams*, the *Interaction Diagrams*, and the *Task/Goal Diagrams* obtained in the previous analysis steps.

Based on the *PROSA Guidelines* the designer must refine both the *Organization* model and the *Interaction* model by adding new or modified relations and interactions among holons in the cooperation domains. The *Environment Model* is built in the fourth step, *Specify Environment Relations*, to represent non-autonomous domain entities with which the holons have to work.

When the five models are specified following the analysis steps described above, the software engineer team has to decide if it is necessary to apply new analysis iterations. To this end, they have to analyze every Abstract Agent in the current iteration in order to figure out whether it is convenient to decompose some of them.

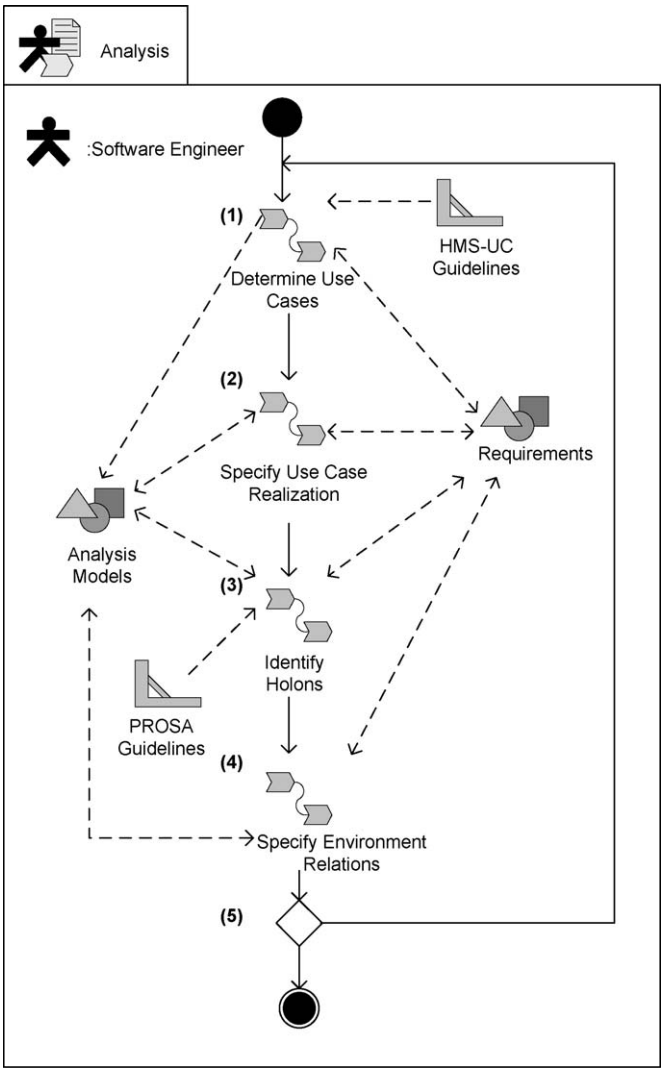

Fig. 3. Analysis phase.

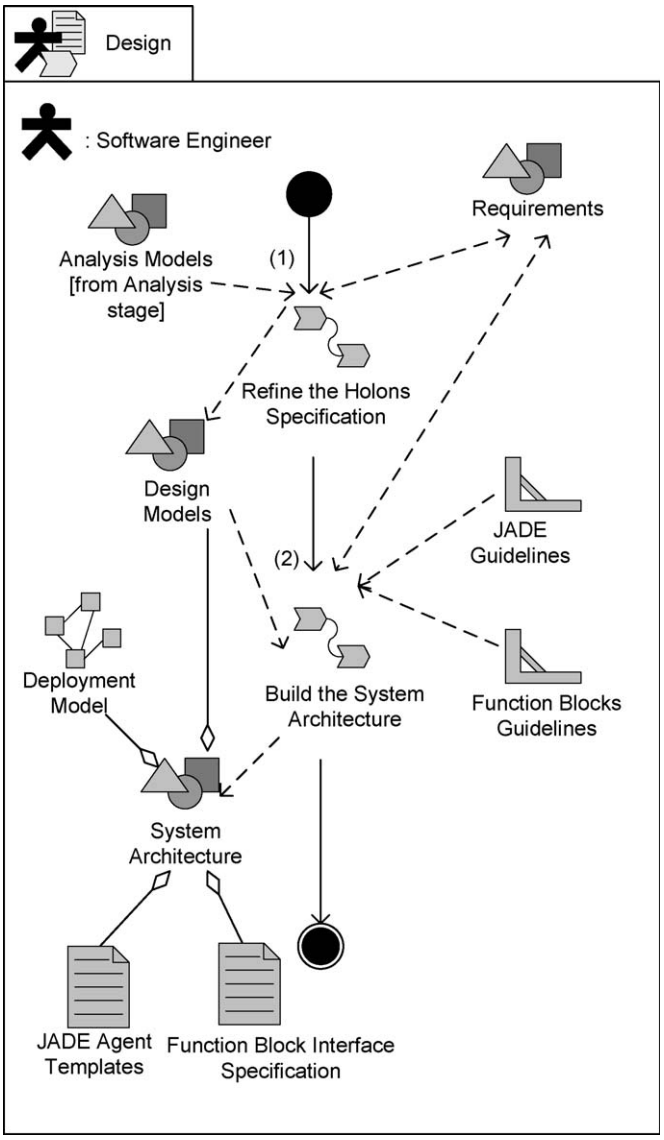

Fig. 4. Design phase.

For every Abstract Agent that it was decided would be decomposed a concurrent analysis process has to be applied in the next iteration. Each of these processes may be conducted in a collaborative fashion by different engineering teams.[2] The requirement specification for every concurrent process is defined by the different models in the previous iteration, which specify the given Abstract Agent. In this way, the incremental analysis steps define the integration rules and models developed by every engineering team responsible of the different analysis processes. When there is no need for further decompositions the teams can proceed to the following stage in the development process (*Holon Design*).

5.3. Holon Design

In the design phase the initial system architecture, i.e. *Analysis Models*, must be completed with details of the target implementation platform (Fig. 4). This phase is divided into two steps.

The first design step, *Refine Holon Specification*, is dedicated to completing analysis models without taking platform-modelling issues into account. The design teams must focus on the "atomic" holons of the previous phase in order to complete their definitions. The *Agent Model* must be revised to include the internal execution states of the holon and their transitions. The *Task/Goal Model* must be analyzed to ensure that each agent goal has a corresponding task that pursues it. Pre and post conditions have to be identified for every modeled task. The *Environment Model* must be detailed to include the resource attributes and the agents' perceptions in terms of application events.

Once every atomic holon is completely specified, the design teams must move up to the nearest abstraction level in the holarchy structure, i.e. to the cooperation domain in which the given holon interacts. Dependencies among cooperation domains/holarchies are refined in the *Organization Model*. The *Interaction Model* is enhanced with preconditions, task executions and effects on the environment and on interacting holons. This bottom-up process must be repeated until there is no higher cooperation domain in the *Analysis Models*.

The last design step is *Build System Architecture*. ANEMONA defines design guidelines to implement the HMS as proposed by

[2] One team may be decomposed into new teams in order to develop the collaborative analysis processes of the next iteration. The way in which a team is decomposed into sub-teams may be done following Abstract Agent complexity features, engineers' experience in related manufacturing processes, etc.

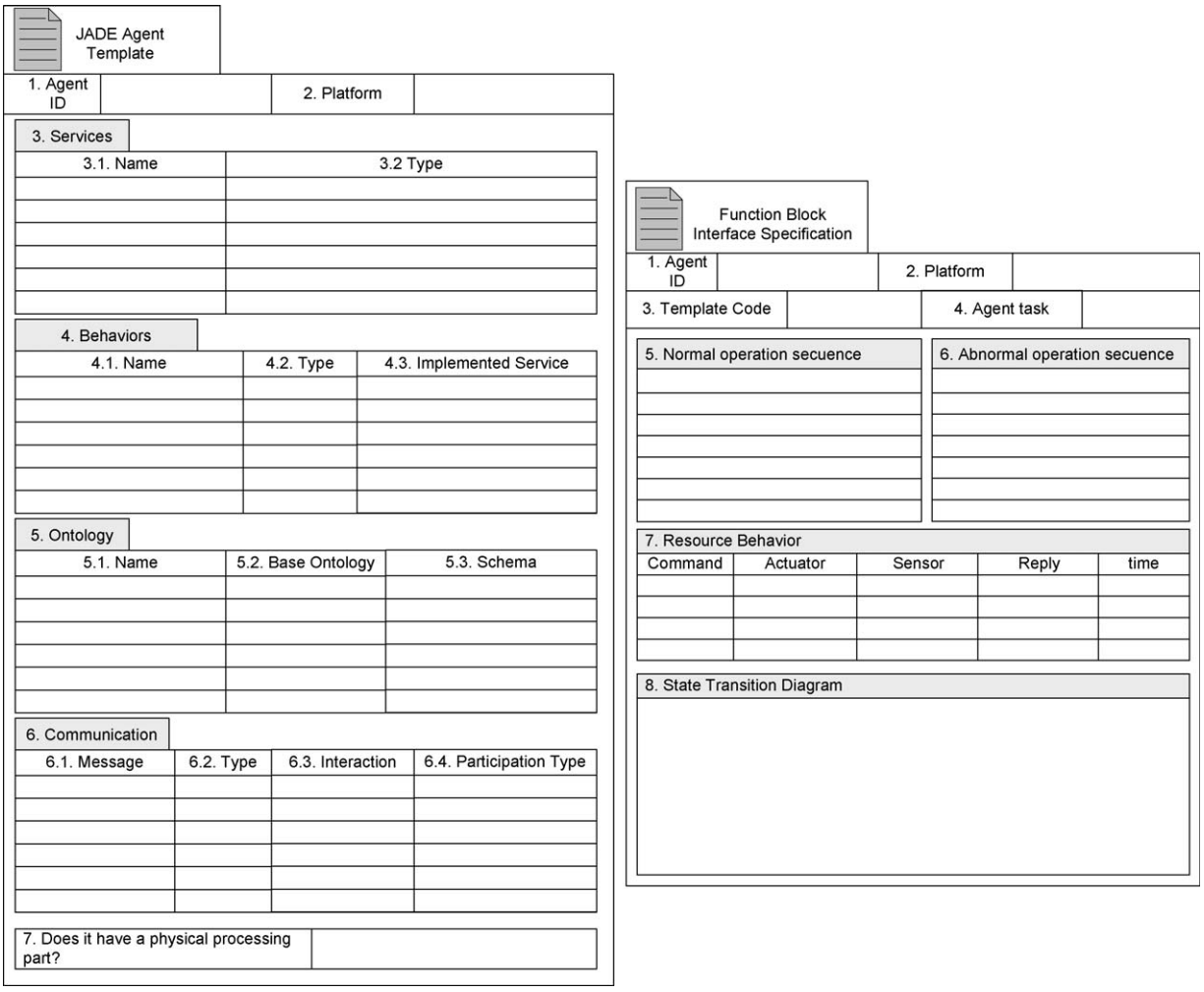

Fig. 5. JADE agent template and function block template.

Christensen in [7]. For high-level control (intra-holon information processing and inter-holon cooperation), ANEMONA provides design guidelines for JADE [24] and for SIMBA [15]. The *JADE Agent Template* (Fig. 5) contains JADE specific characteristics such as agent identifiers, agent behaviours, agent communication and services. The following are some JADE guidelines:

- If the HMS represents a manufacturing system in which there is some kind of distribution (that is, departments, factory floors geographically distributed, or different branches), the control of each of these holons may be managed by a different agent platform ...
- If the HMS has no physical distribution, the engineering team has to analyze the Abstract Agents identified in the *Design Models*. If there is a large number of agents, the team may consider the possibility of distributing the HMS among different agent platforms. In order to figure out the number of these platforms, they have to analyze the relationships and dependencies among the cooperation domains, security and data encapsulation requirements, functionality, internal interactions, etc. The number of the resulting groups has to be used as a guideline to the number of agent platforms ...
- For the information processing part of every agent the JADE agent template of Fig. 5 has to be completed ...
- In order to complete Section 3 of the JADE agent template (Fig. 5) the *Agent Model*, *Task*/*Goal Model* and the *Interaction Model* have to be used. This section must be completed with all the tasks a given agent offers as services. Section 3.2 must indicate whether the service is offered for internal agents only, or it is also available for agents of other platforms ...

For low-level control (physical operations), our methodology provides design guidelines for function blocks (IEC 61499 series of standards) [25]. The *Function Block Interface Specification* (Fig. 5) contains a table so that there is an ordered list of corresponding physical device commands and responses for every agent physical action (task). Some guidelines are as follows:

- For each task (physical processing part) of each agent (resource holon) of the *Design Model* a *Function Block Interface Specification* (Fig. 5) has to be completed ...
- In order to complete the operation sequence of the resource machine (Sections 5 and 6 of Fig. 5) the engineer has to analyze the device manufacturer specification, the *Agent Models* and *Task*/*Goal Models* produced in previous phases ...

The work product of this step is the System Architecture composed of the Design Models (built in the previous step), the JADE agent templates, the SIMBA Specification and the Function Block Interface Specification. A JADE agent template is produced for each agent in the Design Models. A SIMBA Specification is built for every hard real time agent with no physical processing part. The

Function Block Interface Specification is produced for the physical processing part of each agent representing physical processes, equipment or machines. The last Design Model is the UML Deployment Diagram. This diagram models the physical architecture of the different system nodes, the agent platforms and the allocation of containers.

5.4. Implementation. SetUp and Configuration. Operation and Maintenance

From the *System Architecture* the *Holon Implementation* phase produces the *Executable Code* for the HMS. In this phase the programmers have to implement the information processing part of each JADE agent and the physical processing part of each agent representing physical processes, equipment or machines. For the first task the designers may use the JADE programmers guide [24], and for the second task they may use the programmers guide defined by the standard IEC 61499 [25]. To implement the intra-holon communication the programmers can implement a blackboard system [26] or a special management service interface function block [27]. Configuration activities are carried out in the *SetUp and Configuration* phase to deploy the HMS at the target destination. Finally in the *Operation and Maintenance* phase maintenance activities are performed. In the case of new requirements, a new development process must be initiated.

6. A simplified real case study

In order to illustrate ANEMONA's concepts and development process in this section we present a simplified real case study of a Ceramic Tile Factory.

The Ceramic Tile sector is very competitive. This competition is ultimately reflected in an increase in the variety of products and services, together with a decrease in production costs. Moreover, improvements in customer service are also needed to stand up to ever-expanding foreign companies. In this way, in the Ceramic Tile Factory under study, different products with diverse sizes, designs and compositions are produced. Moreover, the factory has to deal with two kinds of clients: building firms and wholesalers. Several business processes can be distinguished in a Ceramic Tile Factory. Firstly, a design department defines which ceramic products will be produced in the current season. Then, the commercial department, based on historical sales data, orders, etc., makes a prediction of sales and orders forecast. Later, medium term production orders are defined, sequencing the different product lots to be produced. This sequence configures a tentative Master Plan, which is normally used as the major input data to generate the production programming, which includes activities such as determining start time and resource allocation for a specific lot production. The production department uses the Master Plan, information about raw materials availability, plant status, etc., to generate the production-programming schedule. Finally, all tasks related with the production and final storage of the different product lots are carried out. Usually, the Ceramic Tile production process is represented as a three stage hybrid flow shop with sequence dependency in which three stages can be identified: press and glass lines (first stage), kiln (second stage), and classification and packed lines (third stage). Each stage is a productive phase with different times, resources and objectives. Finally, the commercial department sells factory products and manages orders from clients.

The HMS for the Tile Factory must: (i) integrate the different departments of the company, (ii) arrange factory resources for both on-demand and stock production orders, and (iii) automate resources and processes controls at different levels in the company.

Let us assume that the engineering team has completed the *Requirements* specification of the Ceramic Tile Factory and they are going to begin the analysis of the HMS. They have to follow the work-flow presented in Fig. 3 to specify the HMS in terms of the five models defined in ANEMONA and UML *Use Case Diagrams*. Fig. 6 shows an initial Use Case Diagram of the Tile Factory, in which we can see the different cooperation domains [28] that have to be implemented in order to fulfil the system's goals. These Use Cases where identified following the *HMS UC Guidelines* of ANEMONA.

Based on the *PROSA Guidelines* the engineering team produces both the *Organization model* (Fig. 7) and the *Interaction model* by adding new or modified relations and interactions among holons in the cooperation domains. Fig. 8 shows the *Interaction Diagram* to *Request Row Material*. In this interaction the *Factory Manager* and the *Warehouse Manager* cooperate to fulfil the abstract goal "*To get Raw Material*". The *Agent Model* (Fig. 9) is built to specify holon capabilities and responsibilities in terms of tasks and goals which are described in detail in the *Task/Goal Model*.

Let us suppose that iteration 1 of the analysis phase is completed. Table 1 summarizes the holons of the HMS in iteration 1. Based on this summary it needs a second iteration of the analysis phase, in which 8 Abstract Agents have to be decomposed. Therefore, 8 concurrent analysis processes have to be applied in iteration 2.

Let us assume that the software engineer is analyzing the *Factory Holon* in iteration 2. The different analysis steps are applied to its requirement specification in the same way as in the first iteration. For example the *Organization Diagram* in Fig. 10 is produced after applying the first analysis step.

Let us suppose that all of the analysis steps were executed in iteration 2 and, after applying a similar study to the one applied in Table 1, the Abstract Agents of the *Factory Holon* which need decomposition are the following: *Graze and Press Holon*, *Oven Warehouse Holon*, *Oven Holon*, *Classification Warehouse Holon*, *Classification Holon*, and *Maintenance Holon*. Therefore, a third iteration of the analysis phase is needed. The third analysis iteration is completed and the Ceramic Tile Factory is composed of 46 types of (atomic) agents organized into 13 organizations and

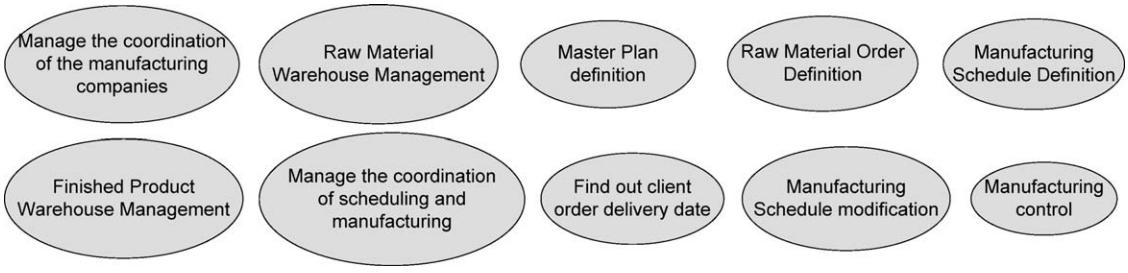

Fig. 6. UML Use Case Diagram of the Ceramic Tile Factory.

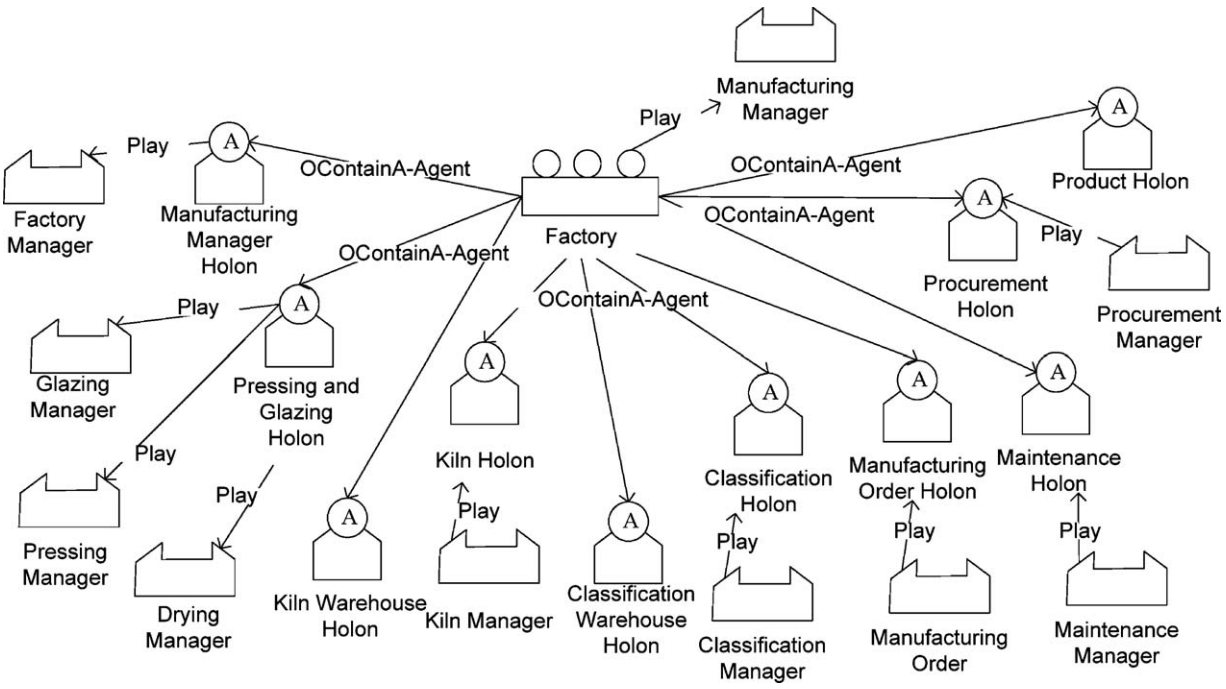

Fig. 7. An Organization Diagram with the Abstract Agents of the Ceramic Tile Factory HMS.

there is no need for further decompositions. Therefore, we can proceed to the following stage in the development process (*Holon Design*).

The *Analysis Models* of the Ceramic Tile Factory reveal that there are 46 atomic agents. These agents will be the first focus of the software engineer team in order to complete their design. Fig. 11 shows the *Agent Diagram* of the *Oven Holon*. In this figure we can see the mental state *Normal Functioning* of the *Oven Holon*, in which the agent is in charge of executing temperature control tasks in the different oven zones. In addition, these tasks have time constraints

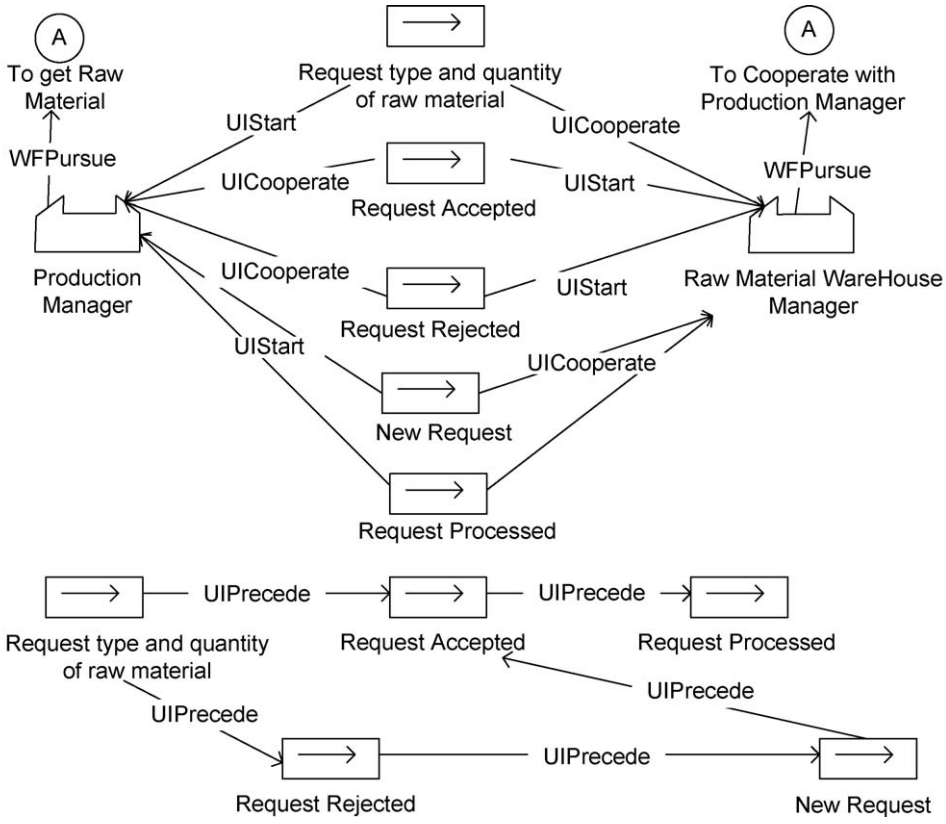

Fig. 8. Request Row Material Interaction Diagram of the Ceramic Tile Factory.

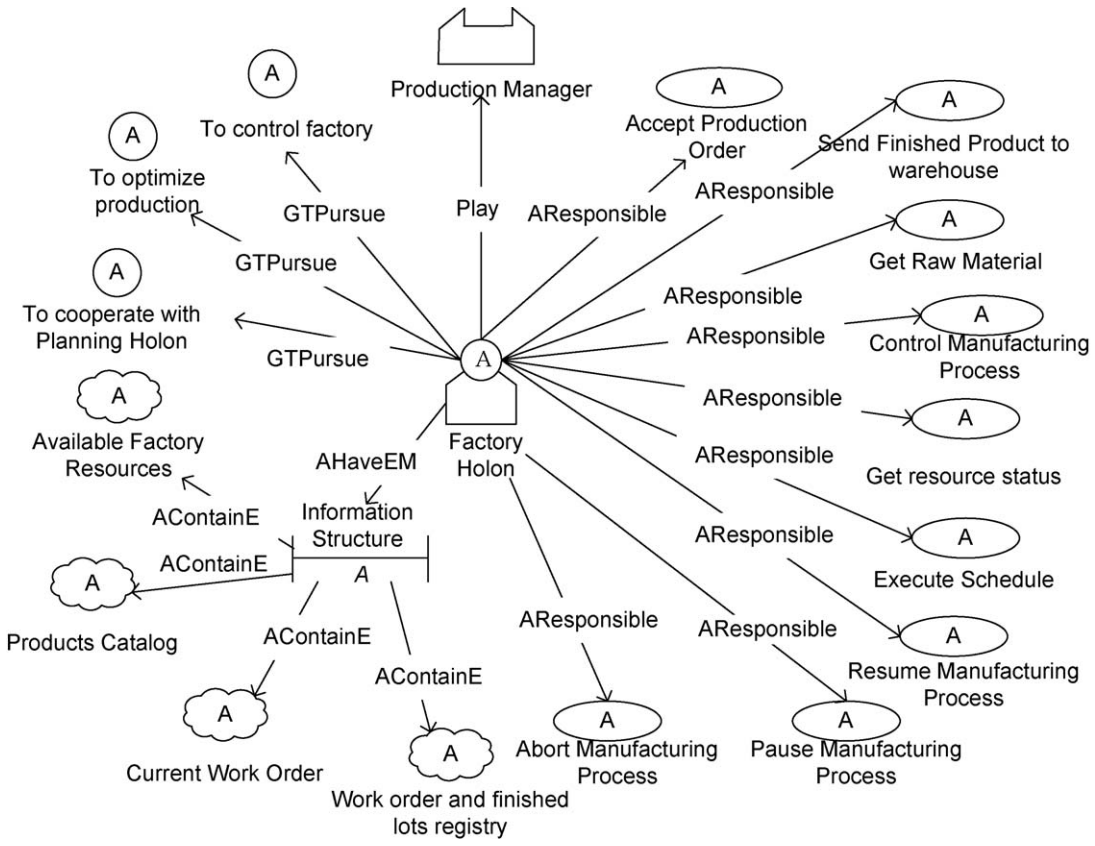

Fig. 9. Agent Diagram of the Factory Holon.

(hard and periodic) and have an execution period and worst case execution time.

In the last design step, Build System Architecture, the engineering team produces the JADE agent template, the Function Block Interface Specification and the Deployment Diagrams. Fig. 12 shows an example Function Block Interface Specification for the task Divert Tile to Belt of the Conveyor Belt Holon. The last Design Model is the UML Deployment Diagram. This diagram models the physical architecture of the different system nodes and the agent platforms and container allocation. Fig. 13 shows the Deployment Diagram of the Factory Holarchy.

Finally the programming team has to implement the information processing part of each JADE agent and the physical processing part of each agent representing physical processes, equipment or machines.

In order to evaluate this experimental prototype, different simulation tests have been executed in which the operating conditions have been analyzed. These tests give a measure of the system's reliability, robustness, flexibility and efficiency, not only under normal conditions, but also under non-standard operating circumstances. More specifically, tests on the functionality and efficiency of the system; concurrent programming capacity; system reaction to unexpected events such as machine failures; and agent failures are also carried out. A complete analysis of these tests can be studied in [29].

Fig. 14 shows some significant results of the evaluation experiments for the *Tasks Scheduling* cooperation scenario on the factory floor. This graphic represents the time elapsed from when an error happens until its detection by the *Plant Wrapper* agent (''PW detects''); and until the *Scheduling Execution Monitor*

Table 1
Holons of the iteration 1 of the analysis phase.

Holon	Atomic	Comments
WareHouse Holon	No	There are many warehouses of raw materials with many processes and resources.
Purchases Holon	No	In the purchasing process distributed information is needed.
KCG Holon	Yes	Its implemented functions are simpler and may be controlled by a single agent.
Client Order Holon	No	The client order can be decomposed into parts that can be processed in different factories of KCG.
Production Holon	Yes	The coordinating functions that it implements are simpler and can be controlled by a single agent.
Factory Holon	No	The factory contains a set of resources and processes to control. In addition, its elements can be organized in different ways.
Ceramic Product Holon	No	There is a catalog of ceramic products, each of which has its own characteristics.
Scheduling Holon	No	In the scheduling process there are many processes and works with many resources.
Work Order Simulation Holon	Yes	The work order simulation is not decomposed into parts or sub-orders.
Master Plan Holon	Yes	The master plan is not decomposed into sub-plans.
Planning Holon	Yes	The planning algorithm is not distributed.
Finished Products Stock Holon	No	There are many warehouses with many processes and resources.
Purchases Holon	No	The purchasing process is carried out in different shops with many distributed processes to be controlled

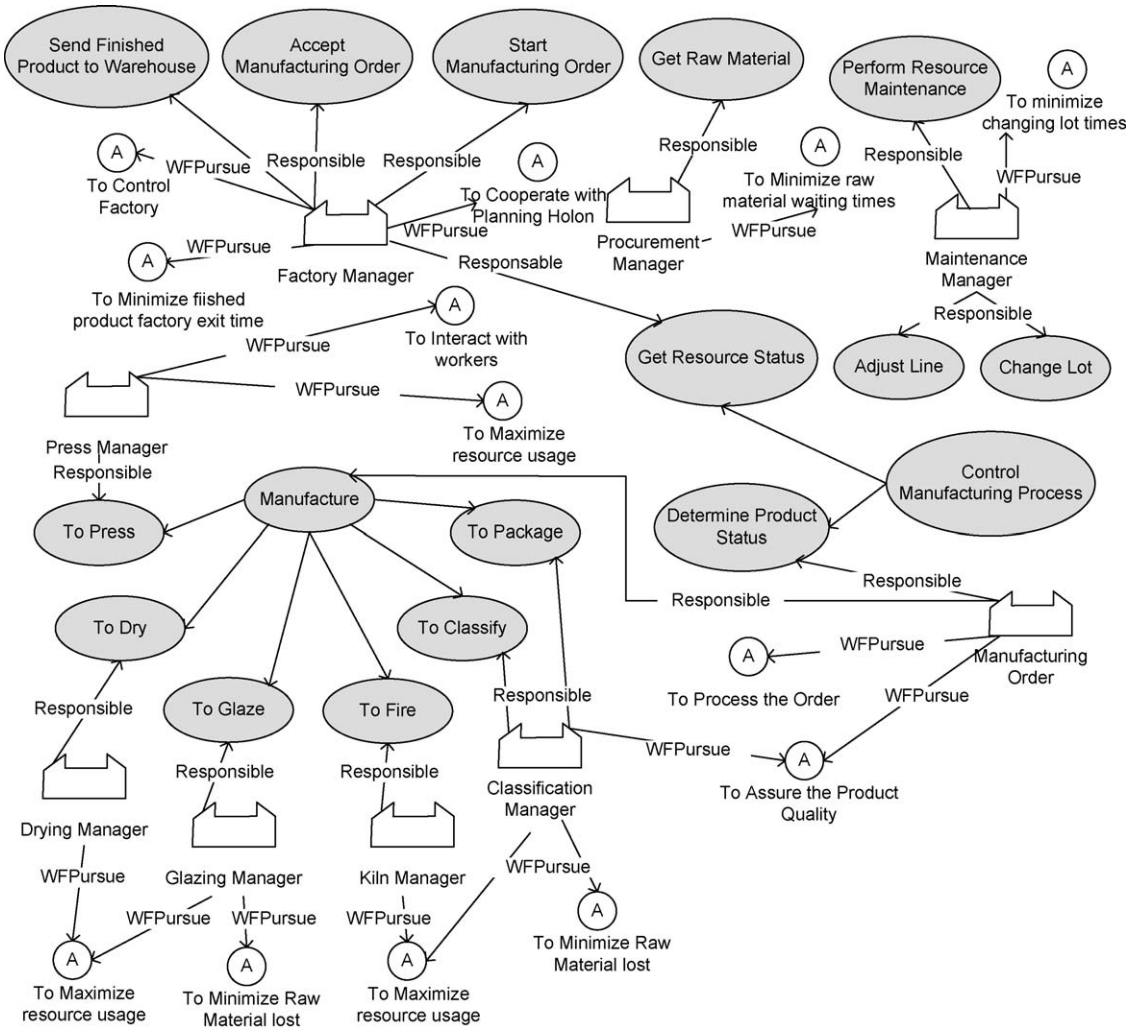

Fig. 10. Organization Diagram of the Factory Holon.

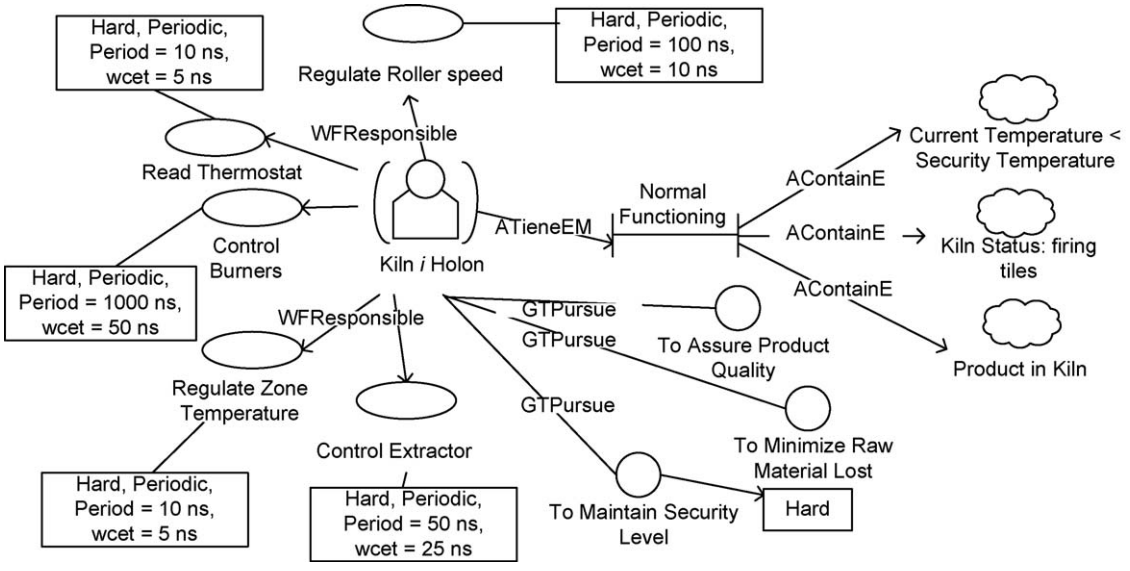

Fig. 11. Design Agent Diagram of the Oven Holon.

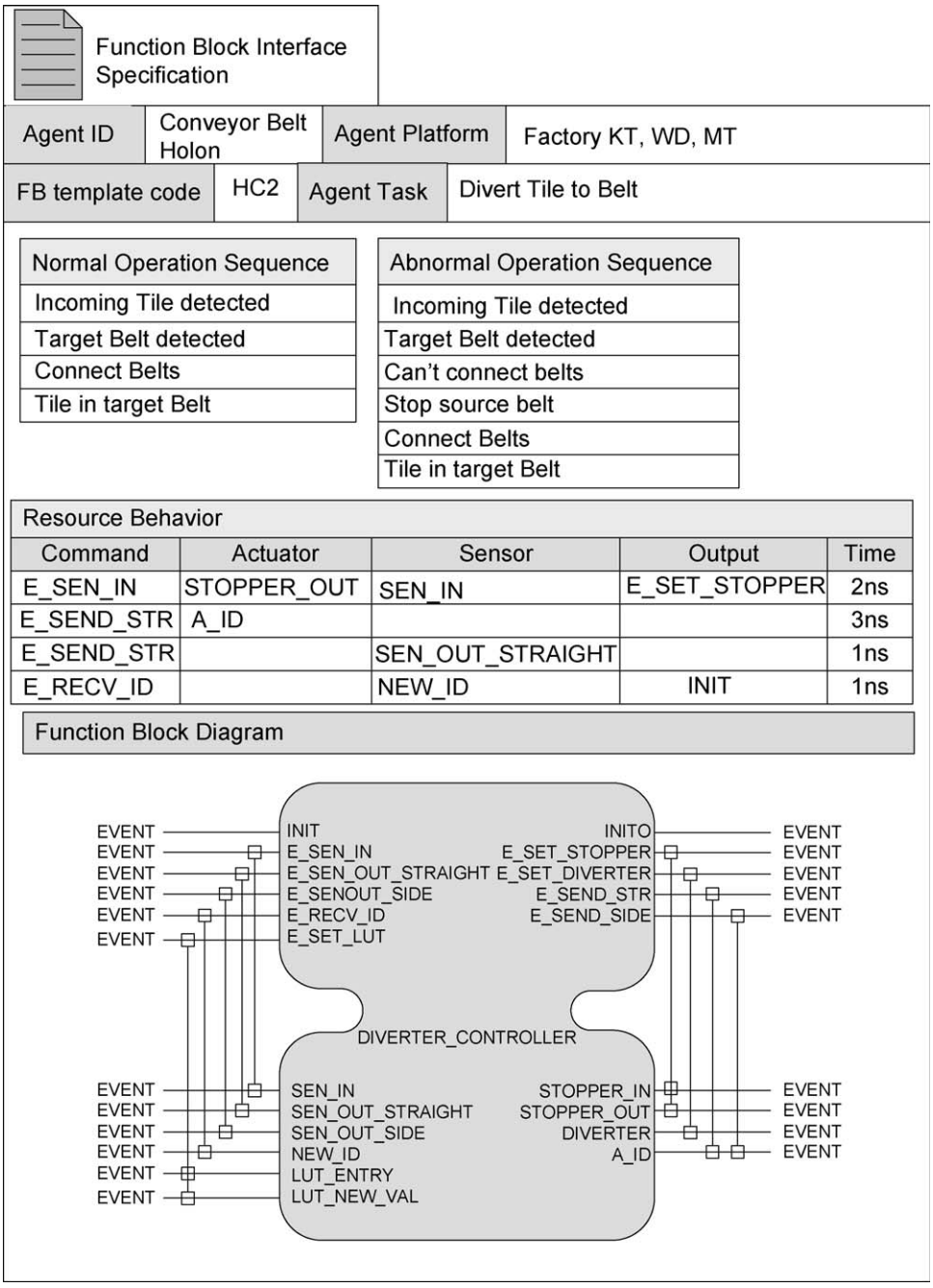

Fig. 12. A Function Block Interface Specification for the task *Divert Tile to Belt* of the *Conveyor Belt Holon*.

agent checks whether the confirmed schedule must be re-programmed ("SEM evaluates"). When the error does not affect the confirmed schedule, the process usually takes around 0.02 s from the failure event until the *Scheduling Execution Monitor's* decision that the error does not affect it. But, as shown in Fig. 14, when the *Plant Wrapper* or the *Scheduling Execution Monitor* agents are busy, the time required increases to 0.25 s.

7. Discussion and future works

In this work, we have described the advantages of using ANEMONA for engineering Holonic Manufacturing Systems. ANEMONA is based on HMS modelling requirements and integrates features from HMS, MAS and Enterprise Modelling techniques. In ANEMONA the HMS is specified by dividing it into more specific characteristics that form different *views* of the system. The ANEMONA development process is a mixed top-down and bottom-up approach. It implements a collaborative-engineering process for HMS, which is achieved by different levels of abstractions (in which each level may be developed by different development groups in a concurrent fashion). The aim of the analysis phase is to provide high-level HMS specifications from the problem *Requirements*, which are specified by the *Client*/*User* and which can be updated at any development stage. The analysis adopts a top-down recursive approach. One advantage of a recursive analysis is that its results, i.e. the *Analysis Models*, provide a set of elementary elements and assembling rules. The next step in the development process is the *Holon Design* stage, which is a bottom-up process to produce the *System Architecture* from the *Analysis Models* of the previous stage. The aim of the *Holon*

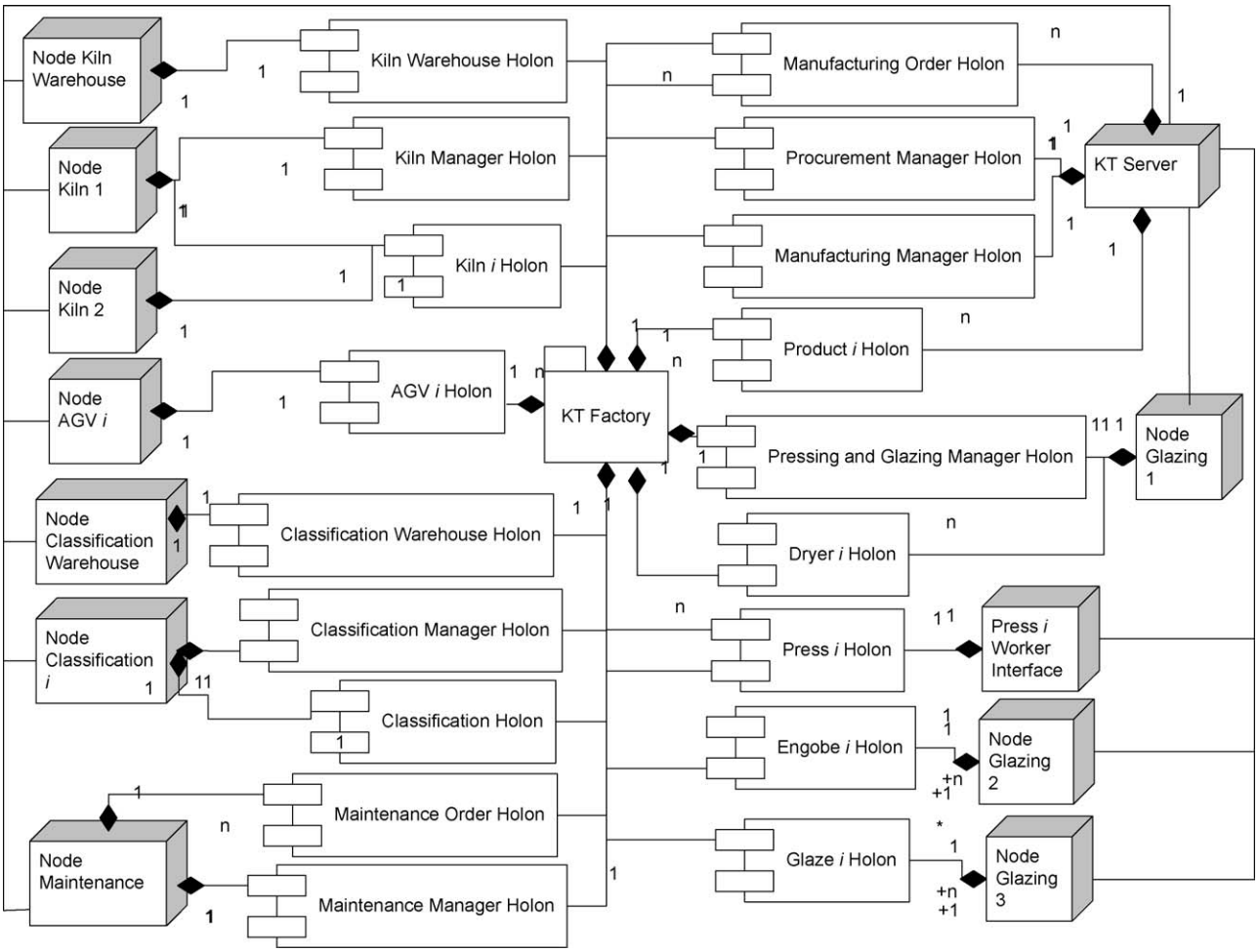

Fig. 13. Deployment Diagram of the Factory Holarchy.

Implementation stage is to produce an *Executable Code* for the *SetUp and Configuration* stage. Finally, maintenance functions are executed in the *Operation and Maintenance* stage. Our approach provides HMS-specific guidelines to help the designer in every development step.

ANEMONA expands the state of the art methodologies for HMS in that it provides a set of appropriate and sufficiently prescriptive methods, models, techniques and guidelines for every HMS modelling requirement. The development process of ANEMONA provides HMS designers with clear, HMS-specific modelling guidelines, and complete development phases for the HMS life cycle.

We are currently applying our methodology to many real-life industrial case studies, using an agent-supported simulation tool guided by ANEMONA modelling entities [30]. We are also working on connecting ANEMONA to state of the art agent platforms for holonic structures such as the JANUS platform [31].

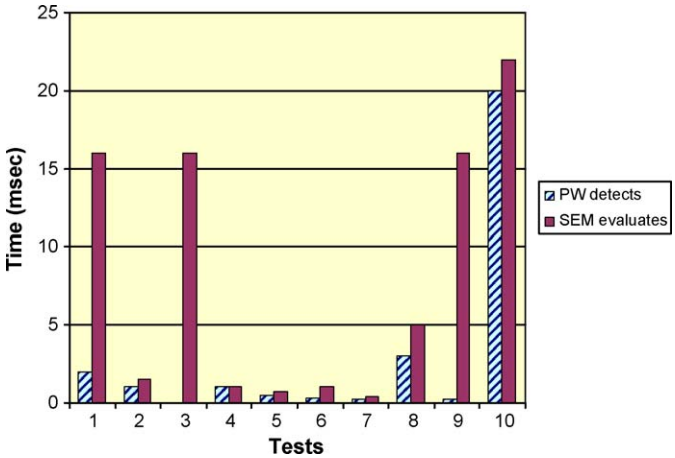

Fig. 14. Time needed for error recovery when error does not affect the calculated schedule.

Acknowledgements

This work is partially supported by research grants CONSOLIDER-INGENIO 2010 under grant CSD2007-00022 and grant TIN2006-14630-C03-01, which is cofounded by the Spanish government and FEDER funds.

References

[1] W. Shen, Q. Hao, H. Yoon, D.H. Norrie, Applications of agent systems in intelligent manufacturing: an updated review, International Journal of Advanced Engineering Informatics 20 (4) (2006) 415–431.
[2] L. Jin-Hai, A. Anderson, R. Harrison, The evolution of agile manufacturing, Business Process Management Journal 9 (2003) 170–189.
[3] P.R. HMS, HMS Requirements, HMS Server. http://hms.ifw.unihannover.de/, 2008 (accessed October 2008).
[4] M. Wooldridge, N.R. Jennings, Intelligent agents—theories, architectures, and languages, Lecture Notes in Artificial Intelligence, vol. 890, Springer-Verlag, 1995, , ISBN: 3-540-58855-8.
[5] A. Koestler, The ghost in the machine, in: Arkana Books, 1971.

[6] D. Dilts, N. Boyd, H. Whorms, The evolution of control architectures for automated manufacturing systems, Journal of Manufacturing Systems 10 (1) (1991) 79–93.
[7] J. Christensen, HMS/FB architecture and its implementation, in: Agent-Based Manufacturing. Advances in the Holonic Approach, Springer-Verlag, 2003, pp. 53–88.
[8] D.S.B. McFarlane, Holonic manufacturing control: rationales developments and open issues, in: Agent-Based Manufacturing. Advances in the Holonic Approach, Springer-Verlag, 2003, pp. 301–326.
[9] S. Bussmann, N. Jennings, M. Wooldridge, Multiagent Systems for Manufacturing Control. A Design Methodology, Springer-Verlag, 2004.
[10] H. Warnecke, Die Fraktale Fabrik Revolution Unternehmenskultur, Springer-Verlag, 1992.
[11] A. Giret, V. Botti, Holons and agents, Journal of Intelligent Manufacturing 15 (2004) 645–659.
[12] K. Fischer, M. Schillo, J. Siekmann, Holonic multiagent systems: a foundation for organisation of multiagent systems, in: Holonic and Multi-Agent Systems for Manufacturing, 2003, pp. 71–80, LNAI 2744, ISSN 0302-9743.
[13] A. Giret, V. Botti, Towards an abstract recursive agent, Integrated Computer-Aided Engineering 11 (2) (2004) 165–177.
[14] J. Pavon, J. Gomez, Agent oriented software engineering with INGENIAS., in: 3rd International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS 2003): V. Marik, J. Mller, M. Pechoucek: Multi-Agent Systems and Applications II, LNAI 2691, 2003, 394–403.
[15] V. Julian, V. Botti, Developing real-time multiagent systems, Integrated Computer-Aided Engineering 11 (2004) 135–149.
[16] ESPRIT: CIMOSA: Open System Architecture for CIM, Volume 1 of Research Reports ESPRIT, Project 688/5288 AMICE, Springer-Verlag, 1993.
[17] UEML: Unified Enterprise Modelling Language. http://www.ueml.org/, 2008 (accessed October 2008).
[18] A. Giret, V. Botti, Towards a recursive agent oriented methodology for large-scale MAS, Agent-Oriented Software Engineering IV, LNCS, vol. 2935, 2004, pp. 25–35.
[19] A. Giret, V. Botti, On the definition of meta-models for analysis of large-scale MAS, in: Multiagent System Technologies, LNAI, vol. 3187, 2004, pp. 273–286.
[20] OMG, O.M.G.: Software Process Engineering Metamodel Specification Version 1.0. http://www.omg.org/docs/formal/02-11-14.pdf, 2008 (accessed October 2008).
[21] IDEF: IDEF Family of Methods. http://www.idef.com/, 2008 (accessed October 2008).
[22] A. Giret, V. Botti, From system requirements to holonic manufacturing system analysis, International Journal of Production Research 44 (18–19) (2006) 3917–3928.
[23] H. Van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, P. Peeters, Reference architecture for holonic manufacturing systems: PROSA, Computers in Industry 37 (1998) 255–274.
[24] JADE: Java Agent DEvelopment Framework. http://jade.tilab.com/, 2008 (accessed October 2008).
[25] IEC: International Electrotechnical Commission: Function Blocks, Part 1—Software Tool Requirements, PAS 61499-2, 2001.
[26] D. McFarlane, M. Kollingbaum, J. Matson, P. Valckenaers, Development of algorithms for agent-oriented control of manufacturing flow shops, in: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 2001.
[27] M. Fletcher, R. Brennan, Designing a holonic control system with IEC 61499 function blocks, in: Proceedings of the International Conference on Intelligent Modeling and Control, 2001.
[28] M. Fletcher, E. Garcia-Herreros, J. Chritensen, S. Deen, R. Mittmann, An open architecture for holonic cooperation and autonomy, in: Proceeding of HoloMAS'2000, 2000.
[29] M. Garcia, S. Valero, F. Argente, A. Giret, V. Julian, A FAST method to achieve flexible production programming systems, IEEE Transactions on Systems, Man, and Cybernetics–Part C: Applications and Reviews 38 (2) (2008) 242–252.
[30] N. Ruiz, A. Giret, V. Botti, An agent-supported simulation architecture for manufacturing systems, in: Proceedings of Agent-Directed Simulation, 2007, pp. 63–70.
[31] N. Gaud, S. Galland, V. Hilaire, A. Koukam, An organizational platform for holonic and multi-agent systems, in: Proceedings of Sixth international Workshop on Programming Multi-Agent Systems, 2008.

Adriana Giret is from Villarrica, Paraguay. She received the BS and ECS degrees in Computer Science from the Catholic University of Asuncion, Paraguay, in 1998 and 1999, respectively. In 2005, she received her PhD degree in Computer Science at the Polytechnic University of Valencia, Spain, where she is a Lecturer. She has published more than 70 papers in journals, books, conferences and workshops. She has been involved in many research projects in the field of multi-agent systems. Her research interests include multi-agent systems, Holonic Manufacturing Systems, and agent-oriented methodologies.

Vicente Botti received the MS in Electrical Engineer (1983) and the PhD in Computer Science (1990) from the Polytechnic University of Valencia, Spain. He is currently a Full Professor at the Polytechnic University of Valencia, where he has also been the Head of the Dept. of Informatics Systems and Computation. His current research interests include: multi-agent systems (more specifically real time multi-agent systems), methodologies for developing multi-agent systems, artificial societies, real time systems, mobile robotics (in which he has developed his own models, architectures and applications) and soft computing techniques. He has participated in more than thirty research projects founded by the European Commission and Spanish Government (principal researcher of ten of these projects), and in more than ten technology-transfer agreements (principal researcher of five these agreements). He has published more than two hundred papers in specialized conferences and scientific journals.