MAS Methodology for HMS

Adriana Giret, Vicente Botti, and Soledad Valero

Departamento de Sistemas Informáticos y Computación,
Universidad Politécnica de Valencia, Spain
46022 Valencia, Spain
Phone: +34 96 387 7000
{agiret, vbotti, svalero}@dsic.upv.es

Abstract. Developments in Holonic Manufacturing Systems (HMS) have been reported in three main areas: architectures, algorithms, and methodologies for HMS. Despite the advancements obtained in the first two areas the methodologies for HMS have not received great attention. To date, many of the developments in HMS have been conducted in an almost "empirical way", without design methodology. There is a definite need to have methodologies for HMS that can assist the system designer at every development steps. This methodology should also provide clear and unambiguous analysis and design guidelines. To this end, in this work we present a Multi Agent Methodology for HMS analysis and design.[1]

1 Introduction

In the last ten years, an increasing amount of research has been devoted to holonic manufacturing (HMS) over a broad range of both theoretical issues and industrial applications. We can divide these research efforts into three groups [1]: (i) Holonic Control Architectures, (ii) Holonic Control Algorithms and (iii) Methodologies for HMS. In spite of the large number of developments reported in the first two areas (for a detailed study see [1]), there is very little work reported on Methodologies for HMS. In [2], *a formal specification approach for HMS control* is presented, but it is still in a developmental stage. There are no defined development phases, and no detailed descriptions to explain how to model issues such as cooperation in the holarchy, holon autonomy and system flexibility. In [3], it is proposed an agent organization to model each holon/holarchy that is independent of any holon architecture. However, it is focused only on the holarchy definition and does not define the development phases.

There is a definite need to have methodologies for holonic systems [1], 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. We believe that methodologies from the Multi Agent Technology (MAS) are good candidates for modeling HMS due to the following: the similarities between the holonic and the agent approaches, the wide

[1] This work is partially supported by research grants TIC2003-07369-C02-01 from the Spanish Education Department and CICYT DPI2002-04434-C04-02.

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 39–49, 2005.

use of agents as the implementation tool for holonic systems, and the availability of complete MAS Methodologies. 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.

In this work we present a MAS Methodology for HMS analysis and design. Section 1, introduces the Abstract Agent notion to model the holon recursive structure. Section 2, lists the requirements for a methodology for HMS and Section 3, presents it. Finally, in Section 4, we summarize the conclusions and future works.

2 Abstract Agent and Holon

The HMS consortium has defined the following holon characteristics and holonic concepts [4]:

- Holon - an autonomous and cooperative building block of a manufacturing system 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. A holon can be part of another holon.
- Autonomy - the capability of a holon to create and control the execution of its own plans and/or strategies (and to maintain its own functions).
- Cooperation - the process whereby a set of holons develops mutually acceptable plans and executes them.
- Self-organization - the ability of holons to collect and arrange themselves in order to achieve a production goal.
- Holarchy - a system of holons that can cooperate to achieve a goal or objective. The holarchy defines the basic rules for cooperation of the holons and thereby limits their autonomy.

An agent is an autonomous and flexible computational system that is able to act in an environment [5].

Holons and agents are very similar concepts (for a detailed comparison of these two notions see [6]). In [6], 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 [7] we proposed the Abstract Agent notion as a modeling artifact for 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 simplify and 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. Thanks to the integration of the holon recursive property into an Abstract Agent, the Abstract Agent is useful not only for HMS

but for the modeling of complex systems as well. An Abstract Agent that acts in organizational structures can encapsulate the complexity of subsystems (simplifying representation and design) and can modularize its functionality (providing the basis for integration of pre-existing Multi Agent Systems and incremental development). The Abstract Agent facilitates the modelling of organization of organizations (as well as, Multi Agent Systems of Multi Agent Systems).

3 Requirements of a Methodology for HMS

Manufacturing requirements impose important properties on HMS [4]. These properties define functional attributes and specific requirements for the HMS structure and the HMS development process which must be considered in the methodology. We have defined a HMS methodology requirements list based on the study of the developments reported in HMS and on our experience with software methodologies:

1. Manufacturing control systems require autonomous entities to be organized in hierarchy and heterarchy structures [4].
2. Manufacturing control units require a routine-based behavior that is both effective and timely [8].
3. A methodology for HMS should lead straight-forward from the control task on a factory resource or factory function to autonomous entities [8,4].
4. A methodology for HMS should define a development process that is guided by abstraction levels, and should also provide modeling artifacts, tools and guidelines to manage this process.
5. A methodology for HMS should define a mixed top-down and bottom-up development process.
6. A methodology for HMS should integrate the entire range of manufacturing activities (from order booking through design, production, and marketing) to model the agile manufacturing enterprise [4].

Bearing these requirements in mind we have studied software engineering methodologies which are best suited for problems of this kind. This study has demonstrated that MAS methodologies are good candidates to work with. To this end, we have defined a MAS methodology for HMS which is based on INGENIAS [9] (a complete MAS methodology that has good performance in the development of complex systems). Our approach attempts to satisfy the requirements listed in this section.

4 Methodology

In this section, we present a MAS methodology for HMS analysis and design that is based on the Abstract Agent notion. Every software engineering methodology must define and provide notation, tools, and a development process. In the followings sub-section, we present the notation and the development process of our methodology. The development tools for this methodology will be available in the near future. We use Abstract Agent and holon as similar notions [6].

4.1 Notation

In our approach, the HMS is specified by dividing it in more specific characteristics that form different *views* of the system. These views are defined in terms of MAS technology; therefore, we talk about agents, roles, goals, beliefs, organizations, etc. The views can be considered as general MAS models that can also be applied to other domains. The way in which the views (models) are defined [10,11] is inspired by the INGENIAS methodology. The extensions we have made to the INGENIAS meta-models deal with the following: the addition of the Abstract Agent notion and the properties to model real-time behaviours [12], the redefinition of some relations to conform to the new modeling entities and the dependencies between them. These extensions are motivated by requirements 1 and 2 of Section 3. Here we summarize the 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.

Figure 1 shows some graphical notations of our methodology. The next section shows some example diagrams in which the usage of these notations in the different models is illustrated.

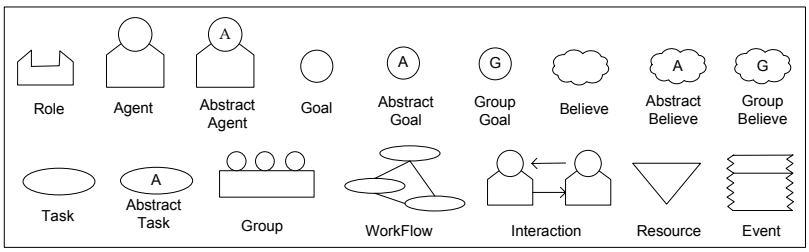

Fig. 1. Some graphical notations of the methodology

4.2 The Development Process

The development process of our methodology provides the HMS designer with clear and HMS-specific modeling guidelines. It also provides complete development phases for the HMS life cycle. The development process is motivated by requirements 3, 4, 5 and 6 of Section 3. In this section, we present the specification of the development process using SPEM diagrams [13]. We also, illustrate

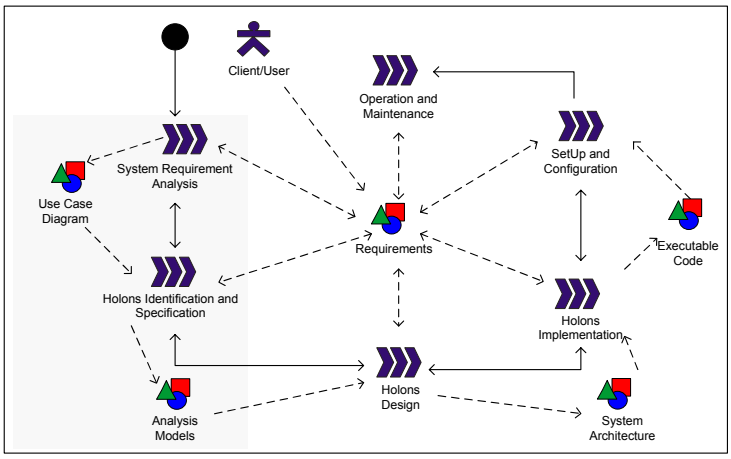

Fig. 2. Stages of the methodology

the development process with modeling diagrams from a Ceramic Tile Factory case study[2]. The tile factory is divided into departments each of which is in charge of a specific function (marketing, tile design, production planning, factory, raw material warehouse, finished tile warehouse, etc.). The tile production process is as follows: the clay is obtained, mixed, refined, dried, pressed or extruded, decorated/glazed and baked in ovens known as kilns. 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.

The development stages are presented in Figure 2. The first stage, *System Requirements Analysis* and the second stage *Holons Identification and Specification* define the analysis phase of our approach. 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 by 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 *Holons 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 *Holons Implementation* stage is to produce an *Executable Code* for the *SetUp and Configuration* stage. Finally maintenances functions are executed in the *Operation and Maintenance* stage.

In the analysis phase (Figure 3a), the designer must specify the HMS in terms of the models presented in Section 4.1 and the UML *Use Case Diagrams*. This is a top-down, recursive, incremental process. The main goal of the analysis phase is to identify the constituent holons and to provide an initial holon specification.

[2] The case study requirements where defined in a joint research project between the GTI-IA group and the CIGIP group of the Polytechnic University of Valencia.

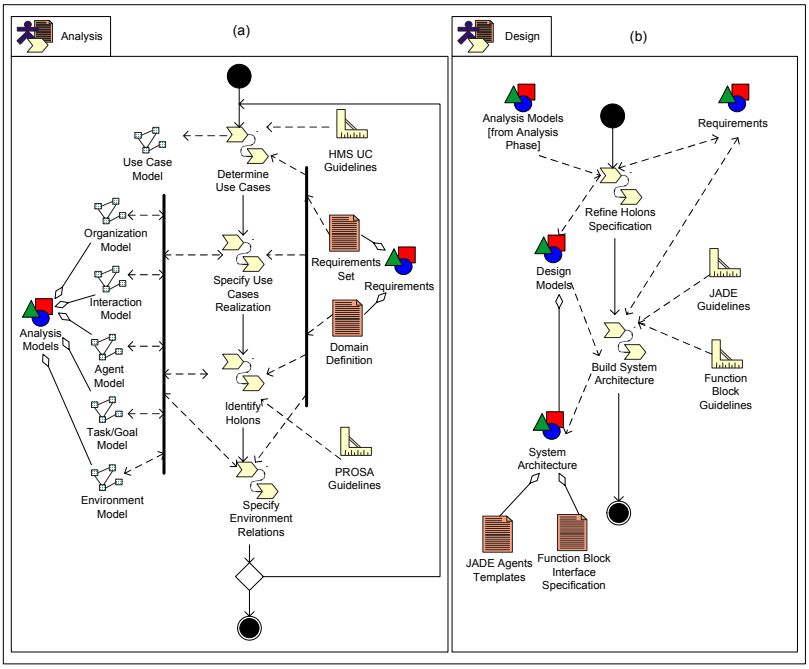

Fig. 3. Analysis and design phases

The designer must produce the *Analysis Models* from the *Requirements Set* and the *Domain Definition*. 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 designer 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 processes as constituent holons of the previous iteration that was decided to decompose. This process is repeated until every holon is completely defined and there is no need for further decompositions. The working definitions of an iteration is as follows.

- The first step is to *Determine Use Cases* by building a *Use Case Model* from the system *Requirements*. We have defined the *HMS UC Guidelines* to help the designer to identify domains cooperations [14] and the system goals as use cases. Use cases can be considered as simpler sub-problems that taken together define the entire system. Figure 4a shows an initial Use Case Diagram of the Tile Factory which is a work product of the first iteration of the analysis phase.
- The use cases of the previous step are analyzed in the step *Specify Use Cases Realization*. Every use case is represented as an Abstract Agent and the interaction and relationships among them are modeled by building *Interaction*

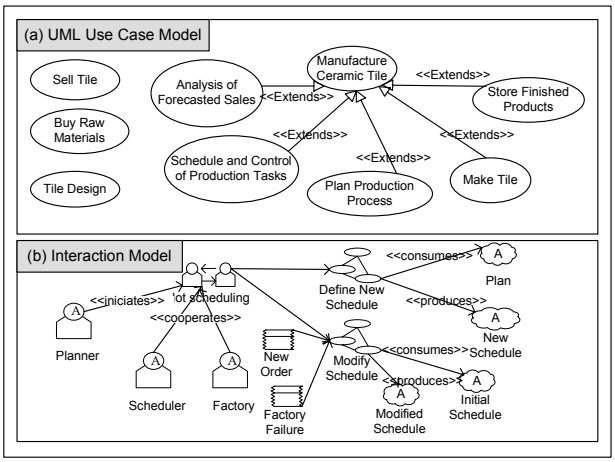

Fig. 4. a) UML Use Case Diagram of the Ceramic Tile Factory. b) An Interaction diagram for scheduling of factory tasks.

Models and *Organization Models*. Figure 5a illustrates the corresponding Organization Model obtained from the Use Case model of Figure 4a.

- In the third step, *Identify Holons*, the designer works with the work products of the previous step, the system *Requirements*, and the *PROSA Guidelines* to identify any new Abstract Agent and to categorize the identified Abstract Agents. The *PROSA Guidelines* are defined based on PROSA types of holons [15]. Some sample rules for identifying holons are: (i) Each production means (a factory, a department, a shop, machine, conveyor, pipeline, component, tool, tool holder, personnel, etc.) and the information processing that controls it, is modeled as an Abstract Agent; (ii) Each product definition or recipe is modeled as an Abstract Agent; (iii) Each task in the manufacturing system (customer order, make-to-stock order, prototype-making order, order to maintain and repair resources, ect.), is represented as an Abstract Agent. Based on these rules 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. Figure 4b shows the Interaction Model to define a new schedule for an order, or to modify a previously defined schedule due to factory failures or new orders. The *Agent Model* (Figure 5b) is built to specify holon capabilities and responsibilities in terms of tasks and goals which are described in detail in the *Task/Goal Model*.
- 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.

In the design phase the initial system architecture, i.e. *Analysis Models*, must be completed with details of the target implementation platform (Figure 3b). This phase is divided into two steps.

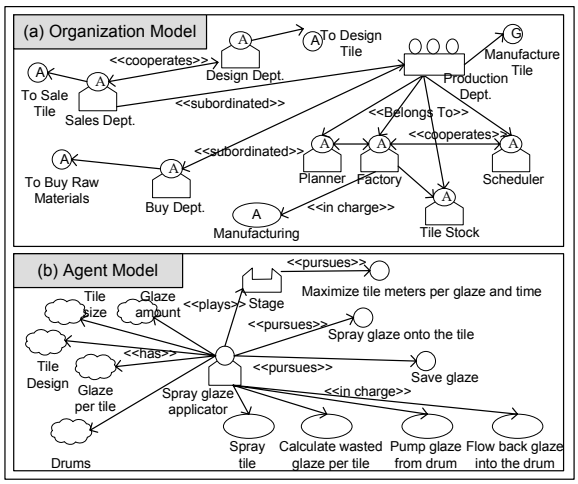

Fig. 5. a) An Organization Diagram of the Ceramic Tile Factory. b) The Agent Model of the *Spray glaze applicator* holon.

- The first design step, *Refine Holons Specification*, is dedicated to complete analysis models without taking into account platform modeling issues. The designer 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 modelled 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 designer 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*. Our approach defines design guidelines to implement the HMS as proposed by Christensen in [16]. For high-level control (intra-holon information processing and inter-holon cooperation), our methodology provides design guidelines for JADE [17]. For the low level control (physical operations), our methodology provides design guidelines for function blocks (IEC 61499 series of standards) [18]. The work product of this step is the *System Architecture* composed of the *Design Models* (built in the previous step), the *JADE Agent Templates* and the *Function Block Interface Specification*. A *JADE Agent Template* is produced for each

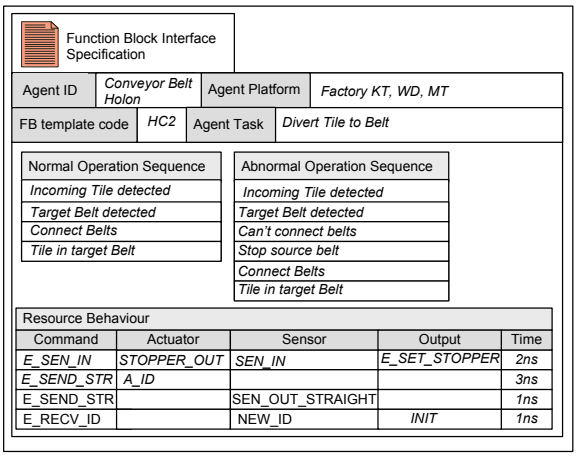

Fig. 6. A Function Block Interface Specification for the task *Divert Tile to Belt* of the *Conveyor Belt Holon*

```
public class GlazeLineManager extends Agent {
// The type of the applicator to find
private String typeApplicator;
// The list of known Glaze Applicators
private AID[] glazeApplicator;
// Put agent initializations here
protected void setup() {
....

//Add a SearchSprayGlazeBehaiour that schedules a request to Glaze Applicators every minute
addBehaviour(new SearchSprayGlazeBehaiour(this, 60000) {
protected void onTick() {
// Update the list of glaze applicators
DFAgentDescription template = new DFAgentDescription();
ServiceDescription sd = new ServiceDescription();
sd.setType("spray-glaze-applicator");
template.addServices(sd);
try {
DFAgentDescription[] result = DFService.search(myAgent, template);
glazeApplicator = new AID[result.length];
for (int i = 0; i < result.length; ++i) {
glazeApplicator[i] = result.getName();
}
}
catch (FIPAException fe) {
fe.printStackTrace();
}
// Perform the request
myAgent.addBehaviour(new RequestPerformer());
}
} );
...
```

Fig. 7. The JADE fragment code for the behaviour of the *Glaze Line Manager* to search for a *Spray Glaze Applicators* in a glaze line

agent in the *Design Models*. The *Function Block Interface Specification* is produced for the physical processing part of each agent representing physical processes, equipment or machines. The *JADE Agent Template* contains JADE specific characteristics such as agent identifiers, agent behaviours, agent communication and services. The *Function Block Interface Specification* contains a table so that there is an ordered list of corresponding physical

device commands and responses for every agent physical action (task). Figure 6 shows an example *Function Block Interface Specification* for the task *Divert Tile to Belt* of the *Conveyor Belt Holon*.

From the *System Architecture* the *Holons Implementation* phase produces the *Executable Code* for the HMS. In this phase the programmer 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. For the first task the designer may use the JADE programmers guide [17], and for the second task he may use the programmers guide defined by the standard IEC 61499 [18]. Figure 7 shows an example JADE fragment code for the *Glaze Line Manager* holon. To implement the intra-holon communication the programmer can implement a blackboard system [19] or a special management service interface function block [20]. 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.

5 Conclusion

In this work, we have presented the notation and development process of a MAS methodology for HMS. In our approach the HMS is specified by dividing it in more specific characteristics that form different *views* of the system. These views are define based on INGENIAS (a proven MAS methodology for complex systems) [9]. We have extended the INGENIAS models to be able to model the HMS requirements properly. These extensions include the notion of Abstract Agent [7] and the properties to model real-time behaviours [12]. They also include the redefinition of some relations to conform to the new modeling entities and the dependencies among them. The development process we have proposed is a mixed top-down and bottom-up approach. 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 by 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 *Holons 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 *Holons Implementation* stage is to produce an *Executable Code* for the *SetUp and Configuration* stage. Finally maintenances functions are executed in the *Operation and Maintenance* stage. Our approach provides HMS-specific guidelines to help the designer in every development step.

We are currently working on the evaluation of our methodology with industrial case studies; for example: the Ceramic Tile Factory presented in this work, and an Assembly and Supplier Company of Automobile Parts. We are also working on CASE tools for our methodology.

References

1. McFarlane, D., S., B.: Holonic Manufacturing Control: Rationales, Developments and Open Issues. In: Agent-Based Manufacturing. Advances in the Holonic Approach. Springer-Verlag (2003) 301–326
2. Leitao, P., Restivo, F.: An Approach to the Formal Specification of Holonic Control Systems. Holonic and Multi-Agent Systems for Manufacturing. LNAI 2744. ISSN 0302-9743 (2003) 59–70
3. Fischer, K., Schillo, M., Siekmann, J.: Holonic Multiagent Systems: A Foundation fo Organisation of Multiagent Systems. Holonic and Multi-Agent Systems for Manufacturing. LNAI 2744. ISSN 0302-9743 (2003) 71–80
4. HMS, P.R.: HMS Requirements. HMS Server, http://hms.ifw.uni-hannover.de/ (1994)
5. Wooldridge, M., Jennings, N.R.: Intelligent Agents - Theories, Architectures, and Languages. Lecture Notes in Artificia Intelligence, Springer-Verlag. ISBN 3-540-58855-8 **890** (1995)
6. Giret, A., Botti, V.: Holons and Agents. Journal of Intelligent Manufacturing **15** (2004) 645–659
7. Giret, A., Botti, V.: Towards an Abstract Recursive Agent. Integrated Computer-Aided Engineering **11** (2004) 165–177
8. Bussmann, S., Jennings, N., Wooldridge, M.: Multiagent Systems for Manufacturing Control. A design Methodology. Springer Verlag (2004)
9. Pavon, J., Gomez, J.: Agent Oriented Software Engineering with INGENIAS. 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
10. Giret, A., Botti, V.: Towards a Recursive Agent Oriented Methodology for Large-Scale MAS. Agent-Oriented Software Engineering IV. **LNCS 2935** (2004) 25–35
11. Giret, A., Botti, V.: On the definition of meta-models for analysis of large-scale MAS. In: Multiagent System Technologies. LNAI 3187 (2004) 273–286
12. Julian, V., Botti, V.: Developing real-time multiagent systems. Integrated Computer-Aided Engineering **11** (2004) 135–149
13. OMG, O.M.G.: Software Process Engineering Metamodel Specification Version 1.0. http://www.omg.org/docs/formal/02-11-14.pdf (2002)
14. Fletcher, M., Garcia-Herreros, E., Chritensen, J., Deen, S., Mittmann, R.: An Open Architecture for Holonic Cooperation and Autonomy. Proceeding of HoloMAS'2000 (2000)
15. Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., Peeters, P.: Reference Architecture for Holonic Manufacturing Systems: PROSA. Computers In Industry **37** (1998) 255–274
16. Christensen, J.: HMS/FB Architecture and Its Implementation. In: Agent-Based Manufacturing. Advances in the Holonic Approach. Springer Verlag (2003) 53–88
17. JADE: Java Agent DEvelopment Framework. http://jade.tilab.com/ (2005)
18. IEC: International Electrotechnical Commission: Function Blocks, Part 1 - Software Tool Requirenments.PAS 61499-2. (2001)
19. McFarlane, D., Kollingbaum, M., Matson, J., Valckenaers, P.: Development of algorithms for agent-oriented control of manufacturing flow shops. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics. (2001)
20. Fletcher, M., Brennan, R.: Designing a holonic control system with iec 61499 function blocks. In: Proceedings of the International Conference on Intelligent Modeling and Control. (2001)