A Holonic Metamodel for Agent-Oriented Analysis and Design

Massimo Cossentino, Nicolas Gaud, Stéphane Galland, Vincent Hilaire, and Abderrafiâa Koukam

Multiagent Systems Group,
System and Transport Laboratory
University of Technology of Belfort Montbéliard
90010 Belfort cedex, France
{massimo.cossentino,nicolas.gaud,stephane.galland,
vincent.hilaire,abder.koukam}@utbm.fr
http://set.utbm.fr

Abstract. Holonic multiagent systems (HMAS) offers a promising software engineering approach for developing applications in complex domains characterized by a hierarchical structure. However the process of building MASs and HMASs is mostly different from the process of building more traditional software systems and it introduces new design and development issues. Against this background, this paper introduces organization-oriented abstractions for agent-oriented software engineering. We propose a complete organizational meta-model as the basis of a future complete methodology that will spread from requirements analysis to code production. In addition to dealing with this last aspect, we introduce our platform, called Janus, that we specifically designed to implement and deploy holonic multiagent systems by adopting concepts like role and organization as the leading issues of the analysis-development process.

Key words: Agent Oriented Software Engineering, Holonic Modeling, Methodology, Holonic multiagent systems

1 Introduction

Sociological concepts have always been a source of inspiration for multiagent researches and recently the agent community has been returning the favor by exploring the potential of agent-based models for studying sociological phenomena (e.g. [1,2,3]). The result of this interaction has been the formalization of a number of sociological, psychological and philosophical concepts with important applications in engineering agent systems. *Holon* and organizational concepts like *Role* and *Organization* are examples of these important concepts. For a successful application and deployment of MAS, methodologies are essential [4]. Methodologies try to provide an explicit frame of the process to model and design software applications. Several methodologies have been proposed for MAS [5] and

Accepted by the 3nd International Conference on Industrial Applications of Holonic and Multi-Agent Systems. V. Marik, V. Vyatkin, A.W. Colombo (Eds.). September 3–5. Regensburg, Germany.

Republished in LNAI 4659, pp. 237–246.

some of them with a clear organizational vision (e.g. [6]). Most of these methodologies recognize that the process of building MASs is mostly different from the process of building more traditional software systems. In particular, they all recognize (to varying extents) the idea that a MAS can be conceived in terms of an organized society of individuals in which each agent plays specific roles and interacts with other agents [7,6]. However, most of them consider agents as atomic entities thus rendering them inappropriate for Holonic MAS (HMAS). In our approach the role is emphasized as a fundamental entity spreading from requirements to implementation. Notice that some of the most known implementation platforms (Jade [8], FIPA-OS [9] and some others) usually do not support the role concept. In our point-of-view the role element offers a number of advantages, e.g. a greater reusability, modularity of developed solutions, and finally encourages a quicker development with less code bugs.

The approach presented in this paper is based on a meta-model for HMAS. It provides a step-by-step guide from requirements to code and it can model a system at different levels of details by using a suite of refinement methods. We propose a meta-model, namely HoloPASSI, as a basis of extension of the PASSI methodology [10] to deal with the analysis and design of HMAS.

The goal of this paper is not to describe the complete methodological process but it rather provides some organization-oriented abstractions that will become the basis of this process. The elements of the meta-model are organized in three different domains. The problem domain deals with the user's problem in terms of requirements, organization, role and ontology. The Agency Domain addresses the holonic solution to the problem described in the previous domain. Finally, the Solution Domain describes the structure of the code solution in the chosen implementation platform. The platform Janus that was developed in our labobatory is selected. It is specifically designed to implement and deploy HMAS. This paper is organized as follows. Section 2 briefly summarizes previous works on Holonic Systems and outlines the key points behind the concept of holon. Section 3 will detail the Problem, Agency and Solution domains of our meta-model.

2 Theoretical Background

The concept of holon is central to our discussion and therefore a definition of what is a holon should be helpful before proceeding. In multiagent systems, the vision of holons is much closer to the one that MAS researchers have of *Recursive* or *Composed* agents. A holon constitutes a way to gather local and global, individual and collective points of view. A holon is thus a self-similar structure composed of holons as sub-structures and the hierarchical structure composed of holons is called a *holarchy*. A holon can be seen, depending on the level of observation, either as an autonomous "atomic" entity or as an organization of holons (this is often called the *Janus effect*).

Two overlapping aspects have to be distinguished in holons: the first is directly related to the holonic nature of the entity (a holon, called super-holon, is composed of other holons, called sub-holons or members) and deals with the

Accepted by the 3nd International Conference on Industrial Applications of Holonic and Multi-Agent Systems. V. Marik, V. Vyatkin, A.W. Colombo (Eds.). September 3–5. Regensburg, Germany.

Republished in LNAI 4659, pp. 237–246.

government and the administration of a super-holon. This aspect is common to every holon and thus called the *holonic* aspect. The second aspect is related to the problem to solve and the work to be done. It depends on the application or application domain. It is therefore called the *production* aspect.

Holonic Systems have been applied to a wide range of applications, Manufacturing systems [11,12], Health organizations [13], Transportation [14], Adaptive Mesh Problem [15], Cooperative work [16] to mention a few. Thus it is not surprising that a number of models and frameworks have been proposed for these systems, for example PROSA [17], MetaMorph [12]. However, most of them are strongly attached to their domain of application and use specific agent architectures. In order to allow a modular and reusable modeling phase that minimizes the impact on the underlying architecture, a meta-model based on an organizational approach is proposed. The adopted definition of role comes from [18]: "*Roles identify the activities and services necessary to achieve social objectives and enable to abstract from the specific individuals that will eventually perform them. From a society design perspective, roles provide the building blocks for agent systems that can perform the role, and from the agent design perspective, roles specify the expectations of the society with respect to the agents activity in the society*". However, in order to obtain generic models of organizations, it is required to define a role without making any assumptions on the agent which will play this role. To deal with this issue the concept of capacity was defined [19]. A capacity is a pure description of a know-how. A role may require that individuals playing it have some specifics capacities to properly behave as defined. An individual must know a way of realizing all required capacities to play a role.

3 Engineering Holons

As PASSI, the HoloPASSI methodology introduces three domains. The first is the problem domain dedicated to the description of a problem independently of a specific solution. The second is the agency domain which introduces agent concepts to describe an agent solution on the basis of the elements of the problem domain. The third and last domain is the solution domain which includes the elements used to implement at the code level the solution described in the second domain. The following sub-sections describe these three domains. The HoloPASSI meta-model is described by an UML class diagram in figure 1. Each domain is separated by a dashed line.

3.1 Problem Domain

The PASSI, and then HoloPASSI, methodologies are driven by requirements. Thus the starting phase is the *Functional Requirements* and *Non Functional Requirements* analysis. (Functional) Requirements can be identified by using classical techniques such as use cases. Each requirement is associated to an *Organization* (see figure 1). An *Organization* is defined by a set of *AbstractRoles*, their *Interactions* and a common context. The associated context (and therefore

Accepted by the 3nd International Conference on Industrial Applications of Holonic and Multi-Agent Systems. V. Marik, V. Vyatkin, A.W. Colombo (Eds.). September 3–5. Regensburg, Germany.

Republished in LNAI 4659, pp. 237–246.

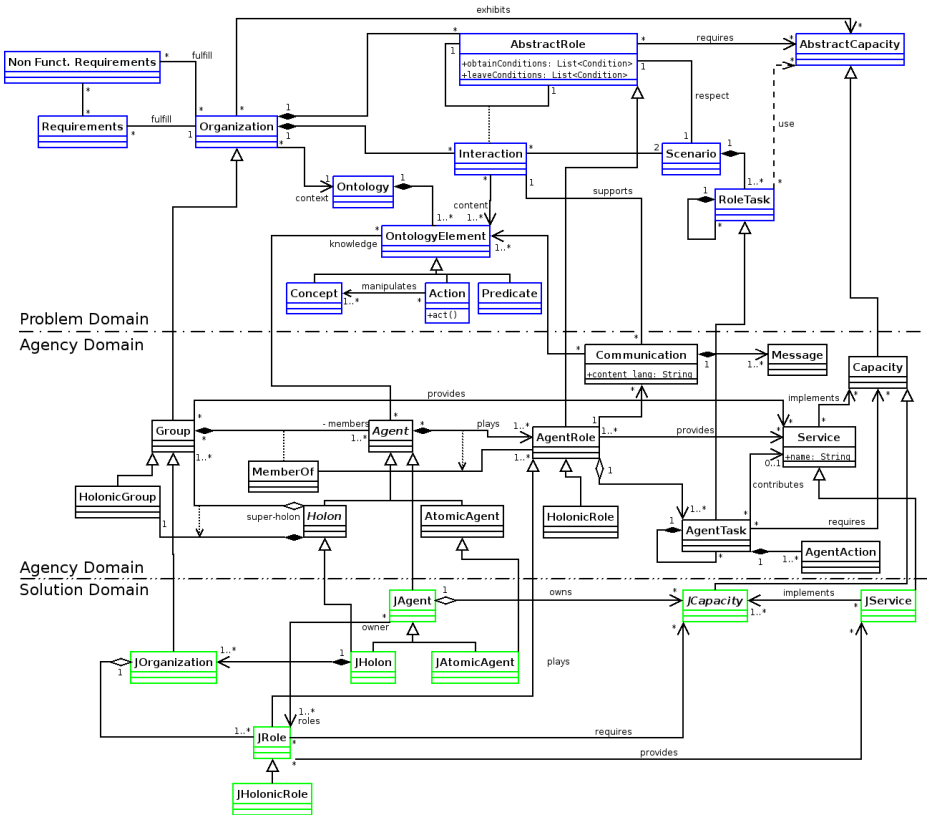

Fig. 1. The Organizational Meta-Model of HoloPASSI

the operating environment too) is defined according to an ontology. An ontology is described in terms of *concepts* (categories, entities of the domain), *predicates* (assertions on concepts properties), *actions* (performed in the domain) and their relationships. The aim of an organization is to fulfill one or more (functional and non functional) requirements. An *Interaction* is composed of the event produced by a first role, perceived by a second one, and the reaction(s) produced by the second role. The sequence of events from one to the other can be iterated several times and includes a not a priori specified number of events and participants. These roles must be defined in the same organization. Figure 2 details an example of an organization and its associated ontology. The *Project Management* organization in figure 2(a) defines two roles *Manager* and *Employee*, and two interactions *Supervise* and *Assigns*. The context of the organization is defined according to the domain ontology described in figure 2(b). As described by John H Holland : "The behavior of a whole complex adaptive system[*cas*] is more than a simple sum of the behaviors of its parts; *cas* abound non linearity" [20]. The notion of capacity was introduced to control and exploit these additional

Accepted by the 3nd International Conference on Industrial Applications of Holonic and Multi-Agent Systems. V. Marik, V. Vyatkin, A.W. Colombo (Eds.). September 3–5. Regensburg, Germany.

Republished in LNAI 4659, pp. 237–246.

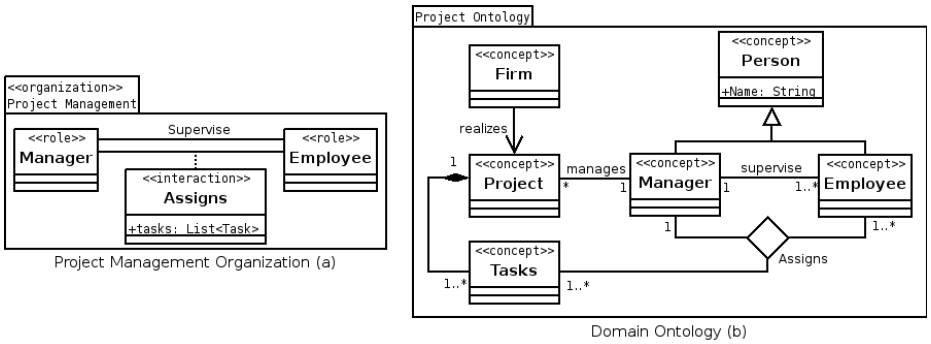

Fig. 2. Organization and Ontology description using two specific UML profiles for class diagram

behaviors, emerging from roles interactions, by considering an organization as able to provide a capacity. It describes what an organization is able to do. Organizations used to model roles interactions offer a simple way to represent how these capacities are obtained from the roles.

Let us now consider our previous example of the Project Management organization. The role *Manager* requires for example the capacity of choosing between various employee the most appropriate one to fulfill a task. Each entity wishing to play the *Manager* role must have an implementation of this capacity (through a service for instance implementing a classical algorithm). The choice between various employees effectively depends on personal characteristics of the entity (e.g. Acquaintances, Beliefs). Basing the description of role behavior on capacities, thus gives to the role more genericity and modularity.

An *AbstractRole* is the abstraction of a behavior in a certain context defined by the organization and confers a status within this context. The status is defined as a set of rights and obligations made available to the role, and also defines the way the entity playing the role is perceived by other entities playing another role in the same organization. Specifically, the status gives the playing entity the right to exercise its capacities. To clearly understand this status aspect, let us return to our preceding example. The status of *Manager* gives the right to use his authority to assign a task to one of his subordinates. No *Employee* will be surprised if a *Manager* uses his authority, because the way under which *Employee* perceive their responsible (status), gives him this right. Another important aspect is that the role (and not the individual, like an agent or a holon, who plays the role) belongs to the organization. This means that the same individual may participate to an organization by playing one or more roles that are perceived as different (and not necessarily related) by the organization. Besides, the same individual can play the same or a different role in other organizations.

The goal of each *AbstractRole* is to contribute to (a part of) the requirements of the organization within which it is defined. The behavior of a *AbstractRole* is specified within a *Scenario*. Such a scenario describes how a goal can be achieved.

Accepted by the 3nd International Conference on Industrial Applications of Holonic and Multi-Agent Systems. V. Marik, V. Vyatkin, A.W. Colombo (Eds.). September 3–5. Regensburg, Germany.

Republished in LNAI 4659, pp. 237–246.

It is the description of how to combine and order interactions, external events, and RoleTasks to fulfill a (part of a) requirement (the goal). A *RoleTask* is the specification of a parameterized behavior in form of a coordinated sequence of subordinate units (a *RoleTask* can be composed of other *RoleTasks*). The definition of these units can be based on capacities, required by the role.

3.2 Agency Domain

After modeling the problem in terms of organizations, roles, capacities and interactions, the objective is, now, to provide a model of the agent society in terms of social interactions and dependencies between entities (Holons and/or Agents) involved in the solution. From an overview at the Agency Domain part of the HMAS meta-model reported in figure 1, some elements are the specialization of other elements defined in the Problem Domain. They constitute the backbone of our approach and they move from one domain to the other in order to be refined and they contribute to the final implementation of the system. These elements are: (*i*) *Group* is a specialization of the *Organization*. It is used to model groups of *Agents* that cooperate in order to achieve a goal. This element is further specialized in the *HolonicGroup* element that is a group devoted to contain roles taking care of the holon internal decision-making process (composed-holon's government). (*ii*) *AgentRole* is the specialization of *AbstractRole*. An *AgentRole* interacts with the others using communications (that are a more refined way for interacting compared to the simple Interactions allowed to the AbstractRole). Several *AgentRoles* are usually grouped in one *Agent* that is in turn a member of the *Group*. An *AgentRole* can be responsible for providing one of more services. (*iii*) *Capacity* is the specialization of the *AbstractCapacity*. It finds an implementation in the *Service* provided by roles and it is used to model what is required by an *AgentTask* in order to contribute in providing a service. (*iv*) *AgentTask* is the specialization of the *RoleTask*. It is aggregated in *AgentRole* and contributes to provide (a portion of) an *AgentRole*'s service. At this level of abstraction, this kind of task is no more considered atomic but it can be decomposed in finer grained *AgentActions*.

A very important element of the MAS meta-model is newly introduced in the Agency Domain; this is the *Agent*. An *Agent* is an entity which can play a set of roles defined within various organizations; these roles interact each other in the specific context provided by the agent itself. The *Agent* context is given by the knowledge, the capacities and the environment. Roles share this context by the simple fact of being part of the same agent. For instance, this means that an agent can play the role of *Buyer* in an organization and later the same agent can sell the goods it had just acquired thus playing for the same organization a different role (*Seller*); conversely, the same agent can also belong to another organization (for instance devoted to monitoring businesses) and thus it can play another role (*AffairMonitor*) to trace the results and the performance exploited during the first acquisition process. It is worth to note that the agent is still not an implementing element but rather it needs further refinements; only when it will become a JAgent (in the Solution Domain) it can really be coded. Figure

Accepted by the 3nd International Conference on Industrial Applications of Holonic and Multi-Agent Systems. V. Marik, V. Vyatkin, A.W. Colombo (Eds.). September 3–5. Regensburg, Germany.

Republished in LNAI 4659, pp. 237–246.

3(a) depicts the context defined by an agent as an interaction space for the roles it plays. These roles, in turn, belong to different organizations, each one defining its own context. An agent in our approach defines a particular context of interaction between roles belonging to different organizations. This aspect is depicted in figure 3(a).

The concept of *Holon* is specialized from the *Agent*. Naturally our definition of holon integrates the *production* and *holonic* aspects previously described in section 2 and it merges them within an organizational approach. A holon is thus a set of roles that can be defined on various organizations interacting in the specific context provided by the agent. A holon can play several roles in different organizations and be composed of other holons. A composed holon (super-holon) contains at least a single instance of a *holonic organization* to precise how members organize and manage the super-holon and a set (at least one) of *production organizations* describing how members interact and coordinate their actions to fulfill the super-holon tasks and objectives. An atomic (non composed) holon is an AtomicAgent. Figure 3(b) illustrates this definition of holon.

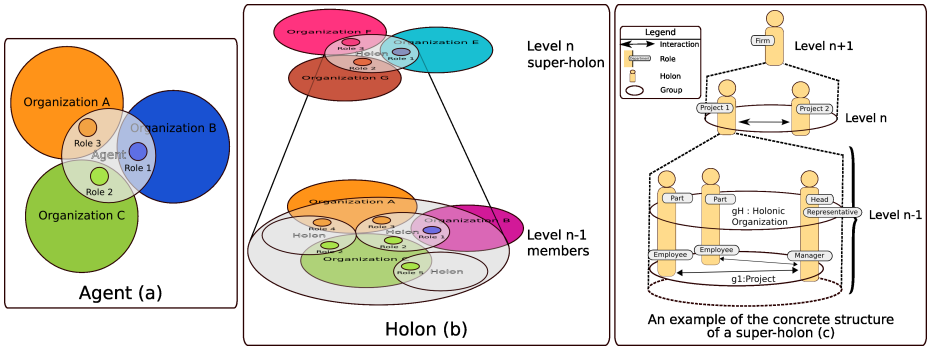

Fig. 3. Agent and Holon symbolic representation, and an example of the concrete structure of a super-holon

The holonic aspect considers how members organize and manage the super-holon. A specific organization, called *Holonic organization*, is defined to describe the government of a holon and its structure (in terms of authority, power repartition). Depending on the level of abstraction, a super-holon can be considered as an atomic entity (let's say level n) or as an organization of holons (let's say level n-1). In the same manner several different holons could be seen as interacting individuals, parts of some organization or as parts of a super-holon. These interactions usually happen in form of communications. Interactions between layers, instead, can happen in two ways: i) (internal) interactions of roles of the same agent if the same agent plays different roles within a holon. For instance, an agent can be the Head delegated to accept some contract (a role of the holonic organization, played at level n) but also the worker which will do part of the work

Accepted by the 3nd International Conference on Industrial Applications of Holonic and Multi-Agent Systems. V. Marik, V. Vyatkin, A.W. Colombo (Eds.). September 3–5. Regensburg, Germany.

Republished in LNAI 4659, pp. 237–246.

related to that contract in the production organization at level n-1; the Agent existence in this case enables the interactions among the different roles. ii) (external) interactions (mostly communications) between roles at different layers of several agents. For instance the Head (layer n) responsible for accepting a contract asks worker roles (layer n-1) to provide the service.

Figure 3(c) illustrates the concrete structure of a super-holon *Project 1* composed of a *production* organization called *g1:Project* and an instance of the holonic organization.

An agent should be able to estimate the agent's competences in order to identify the most appropriate collaborators. The Capacity concept allows us to represent the competences of an agent or of a set of agents. A Capacity describes what an *Agent* should be able to do in order to satisfy the requirements it is responsible for. This means that the set of Capacities obtained by refining the AbstractCapacity of the Problem Domain, becomes the specification of the system requirements in the Agency Domain. Indeed, Capacities describe what the holon is capable of doing at an abstract level, independently of how it does it (this is a concern dealt by the Service concept).

A service implements a capacity thus accomplishing a set of functionalities on behalf of its owner: a role. These functionalities can be effectively implemented by a set of capacities required by the owner role. A role can thus publish some of its capacities and other members of the group can take profit of it by means of a service exchange. Similarly a group, able to provide a collective capacity can share it with other groups by providing a service.

The relation between capacity and service is thus crucial in our meta-model. A capacity is an internal aspect of an organization or an agent, while the service is designed to be shared between various organization or entities. To publish a capacity and thus allow other entities to benefit from it, a service is created.

3.3 Implementing Solution Domain with Janus

This part of the model is related to the Holon Implementation Model; its objective is to provide an implementation model of the solution. This part is thus dependent on the chosen implementation and deployment platform. A platform called *Janus*[1] was built in our lab. It is specifically designed to deal with the holonic and organizational aspects. The goal of Janus is to provide a full set of facilities for launching, displaying, developing and monitoring holons, roles and organizations.

The two main contributions of Janus are its native management of holons and its implementation of the notion of *Role*. In contrast with other platforms such as MadKit [21], JADE, FIPA-OS, the concept of Role in Janus is considered as a first class entity. It thus allows the user to directly implement organizational models without making any assumptions on the architecture of the holons that will play the role(s) of this organization. An organization is defined by a set

[1] http://www.janus-project.org

Accepted by the 3nd International Conference on Industrial Applications of Holonic and Multi-Agent Systems. V. Marik, V. Vyatkin, A.W. Colombo (Eds.). September 3–5. Regensburg, Germany.

Republished in LNAI 4659, pp. 237–246.

of roles and a set of constraints to instantiate these roles (e.g. maximal number of authorized instances). Thus, organizations designed for an application can be easily reused for another. Janus so promotes reusability and modularity, moreover the use of organizational design patterns is strongly encouraged. Each organization is a singleton and it can be instantiated by several groups. Group is the runtime context of interaction. It contains a set of roles and a set of Holons playing at least one of these roles. In addition to its characteristics and its personal knowledge, each agent/holon has mechanisms to manage the scheduling of its own roles. It can change dynamically its roles during the execution of the application (leave a role and request a new one). The life-cycle of each agent is decomposed into three main phases : activation, life, termination. The life of an agent consists in consecutively execute its set of roles and capacities. To describe the personal competences of each agent/holon, Janus implements the concept of *JCapacity* that is an abstract description of a competence; each agent can be equipped from its birth or can dynamically acquire an implementation of a new JCapacity (this function is still under development). In addition to the integration of these personal characteristics, a holon provides an execution context for roles and capacities.

4 Conclusion

This article focuses on the key issues related to the identification of appropriate abstractions for organizational software engineering and to the basis of a suitable methodology from requirement to implementation of complex applications in terms of HMAS. In so doing, the two main contributions of this article are a complete organizational meta-model for the analysis and design of complex systems, and a specific platform, *Janus*, designed to easily implement and deploy models issued from the HoloPASSI meta-model. It fully implements organizational and holonic concepts. However it is also able to support more "traditional" multiagent systems.

This work is a part of larger effort to provide a whole methodology and its supporting set of tools for the analysis, design and implementation of complex applications. Future works will deepen the meta-model concepts and associate a methodology to guide the developer during his work of modeling and implementing a complex (and possibly holonic) multi-agent system.

References

1. Carley, K., Prietula, M., eds.: Computational Organization Theory. (1994)
2. Epstein, M., Axtell, R.: Growing Artificial Societies: Social Science from the Ground Up. MIT Press (1996)
3. Prietula, M., Carley, K., Gasser, L.: Simulating Organizations: Computational Models of Institutions and Groups. AAAI Press (1998)
4. Gasser, L.: Mas infrastructure definitions, needs, prospects. In: Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems. (01)

Accepted by the 3nd International Conference on Industrial Applications of Holonic and Multi-Agent Systems. V. Marik, V. Vyatkin, A.W. Colombo (Eds.). September 3–5. Regensburg, Germany.

Republished in LNAI 4659, pp. 237–246.

5. Iglesias, C., Garijo, M., Gonzalez, J.: A survey of agent oriented methodologies. In: Workshop on Intelligent Agents V: Agent Theories, Architectures and Languages (ATAL-98). Volume 1555., Springer-Verlag (1999) 313–327
6. Zambonelli, F., Jennings, N., Wooldridge, M.: Developing multiagent systems: the gaia methodology. ACM Trans. on Software Engineering and Methodology **12**(3) (2003)
7. Jennings, N.: On agent-based software engineering. Artificial Intelligence **177**(2) (2000) 277–296
8. Bellifemine, F., Poggi, A., Rimassa, G.: JADE: a FIPA2000 compliant agent development environment. In: Agents. (2001) 216–217
9. Poslad, S., Buckle, P., Hadingham, R.: FIPA-OS: the FIPA agent platform available as open source. In: Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 2000). (2000) 355–368
10. Cossentino, M.: IV : From Requirements to Code with the PASSI Methodology. In: Agent-Oriented Methodologies. (2005) 79–106
11. Wyns, J.: Reference architecture for Holonic Manufacturing Systems - the key to support evolution and reconfiguration. PhD thesis, Katholieke Universiteit Leuven (1999)
12. Maturana, F.: MetaMorph: an adaptive multi-agent architecture for advanced manufacturing systems. PhD thesis, The University of Calgary (1997)
13. Ulieru, M., Geras, A.: Emergent holarchies for e-health applications: a case in glaucoma diagnosis. In: IEEE IECON 02. Volume 4. (2002) 2957– 2961
14. Bürckert, H., Fischer, K., G.Vierke: Transportation scheduling with holonic mas - the teletruck approach. In: Conf. on Practical Applications of Intelligent Agents and Multiagents. (1998) 577–590
15. Rodriguez, S., Hilaire, V., Koukam, A.: Towards a methodological framework for holonic multi-agent systems. In: Workshop of Engineering Societies in the Agents World. (2003) 179–185
16. Adam, E.: Modele d'organization multi-agent pour l'aide au travail cooperatif dans les processus d'entreprisee: application aux systemes administratif complexes. PhD thesis, Univ. de valenciennes et du hainaut-cambresis (2000)
17. Brussel, H.V., Wyns, J., Valckenaers, P., Bongaerts, L., Peeters, P.: Reference architecture for holonic manufacturing systems: Prosa. Computers in Industry **37** (1998) 255–274
18. Dignum, V., Dignum, F.: Coordinating tasks in agent organizations. or: Can we ask you to read this paper? In: Coordination, Organization, Institutions and Norms Engineering Societies in the Agents' World. (2006)
19. Rodriguez, S., Gaud, N., Hilaire, V., Galland, S., Koukam, A.: An analysis and design concept for self-organization in holonic multi-agent systems. In: Engineering Self-Organising Systems. Volume 4335 of LNAI., Springer-Verlag (2007) 15–27
20. Holland, J.: Hidden order: how adaptation builds complexity. (1995)
21. Ferber, J., Gutknecht, O., Michel, F.: From agents to organizations: an organizational view of multi-agent systems. In: AOSEIV@AAMAS03. LNCS 2935 (2004) 214–230

Accepted by the 3nd International Conference on Industrial Applications of Holonic and Multi-Agent Systems. V. Marik, V. Vyatkin, A.W. Colombo (Eds.). September 3–5. Regensburg, Germany.

Republished in LNAI 4659, pp. 237–246.