A Metamodel and Implementation Platform for Holonic Multi-Agent Systems

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 offers a promising software engineering approach for developping applications in complex domains. However the process of building MASs and HMASs is radically different from the process of building more traditional software systems and so 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 methodological process from requirements to code. Besides to deal with this last aspect, we introduce our platform, called Janus, that was specifically designed to implement and deploy holonic multi-agent systems.

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 research and recently the agent community has been returning the favor by exploring the potential of agent-based models for studying sociological phenomena (e.g. [5, 10, 19]). 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* is one of these important concepts. For a successful application and deployment of MAS, methodologies are essential [12]. Methodologies try to provide an explicit frame of the process to model and design software applications. Several methodologies have been proposed for MAS [15] and some of them with a clear organizational vision (e.g. [26]). However, most of them see agents as atomic entities thus rendering them inappropriate for Holonic MAS (HMAS in the sequel). Most of these methodologies recognize that the process of building MASs is radically different from

Published in Proc. of the Fifth European workshop on Multi-Agent Systems(EUMAS'07). 13–14 December, 2007. Hammameth (Tunisia).

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 [16, 26]. 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 [2], FIPA-OS [18] 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 which provides a step-by-step guide from requirements to code allowing the modeling of a system at different levels of details using a suite of refinement methods. Our inspiration has been taken from the PASSI methodology introduced by M. Cossentino [7] and it has named our meta-model HoloPASSI.

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 dealing with the user's problem in terms of requirements, organization, role and ontology; the Agency Domain addressing the holonic solution to the problem described in the previous domain; the Solution Domain describing the structure of the code solution in the chosen implementation platform. Here the platform Janus that was developed in our lab is selected. It is specifically designed to implement and deploy HMAS (cf. section 3.3 for more details). These three domains will be detailed in the section 3. Section 2 briefly summarizes previous works on Holonic Systems and outlines the key points behind the concept of holon.

2 Theoretical Background

Holonic Systems have been applied to a wide range of applications, Manufacturing systems [25, 17], Health organizations [24], Transportation [4], Adaptive Mesh Problem [21], Cooperative work [1] 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 [3], MetaMorph [17]. 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 modelling phase that minimizes the impact on the underlying architecture a meta-model based on an organizational approach is proposed. The organizational concepts are presented first and then details are provided about the holonic framework.

2.1 Organizational Background

The adopted definition of role comes from [9]: "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 [20]. 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.

2.2 Holonic Framework

The concept of holon is central to our discussion and therefore a definition of what is a holon should be helpful before proceeding. In Multi-Agent 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. An holon is thus a self-similar structure composed of holons as sub-structures and the hierarchical structure composed of holons is called an *holarchy*. An 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*). The framework uses then organizations to model the status of the members (sub-holons) in the composition of higher level holons (super-holons) and to model the interactions of the members to achieve their goals/tasks.
We have adopted a moderated group structure for holonic MAS[13]. This decision is based on the wide range of configurations that are possible by modifying the commitments of the members toward their super-holon. In a moderated group, we can differentiate two status for the members. First, the *moderator* or *representative*, who acts as the interface with non-member holons, and, second, *represented* members, who are masked to the outside world by their representatives. Even if we use the name "*Moderated Group*" for compatibility with earlier works in this domain, it can be misleading. As we see it, the structure does not necessarily introduced any authority or subordination. The name makes reference to the different status found in the group. We can then adapt this organization by giving the representatives specific authorities according to the problem or constraints.
In order to represent a moderated group as an organization we have identified a set of roles that can represent these concepts. We have chosen to use four roles to describe a moderated group as an organization: Head, Part, Multi-Part and StandAlone. The three first roles describe a status of a member inside a super-holon. The Stand-Alone role represents, on the other hand, how non-members are seen by an existing holon.
As shown in the figure 1 the representatives of the super-holon play the *Head* role. A *Head* member becomes then part of the visible face of the super-holon. This means that the *head* becomes a kind of interface between the members of

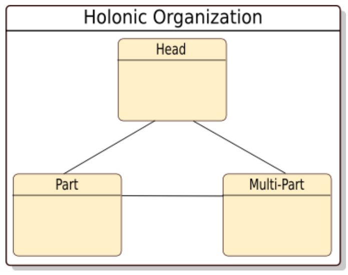

Fig. 1. RIO Diagram of the holons members

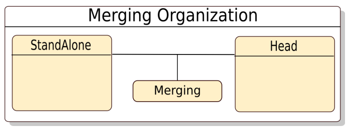

Fig. 2. RIO Diagram of the Merging Organization

the holon and the outside world. The *head* role can be played by more than one member at the same time.
The members can confer the *head* a number of rights or authorities. According to the level of authority given to *heads*, super-holon can adopt different configurations. Thus, the $Head$ role represents a privileged status in the super-holon. *Heads* will generally be conferred with a certain level of authority. However, these members have also an administrative load. This load can be variable depending on the selected configuration.
It is important to remark that when a set of holons merge into a super-holon a new entity appears in the system. In this case, they are not merely a group of holon in interaction as in "traditional" MAS theory. The super-holon is then an entity of its own right. Thus, it has a set of skills, is capable of taking roles, etc. At the same time, as $Heads$ constitute the interface of the super-holon, they are in charge of redistributing the information arriving from the outside. And, thus to "trigger" the (internal) process that will produce the desired result. The $Part$ role identifies members of a single holon. These members are represented by $Head$s within the outside world. While the holon belongs to a single super-holon, it will play this role. However, when the holon is not satisfied with its current super-holon it has two possibilities. The first is to quit its super-holon entirely and try to find a new holon to merge and collaborate with. The second is to try to merge with a second super-holon while remaining as a member of the first super-holon. In this case the holon will change his role to $Multi-Part$. The $Multi - Part$ role is an extension of the $Part$ role. It puts emphasis on a particular situation when a sub-holon is shared by more than one super-holon.
In order to support the integration of new member, we need to provide external holons with a "standard" interface so they can request their admition. From the super-holons point of view, external holons are seen as $StandAlone$ role players. When a super-holon is created, only $Heads$ belong to the interface of the super-holon. Thus, other members ($Part$ and $MultiPart$) should not be visible by external holons. This is modeled by the organization presented in figure **??**. In this organization, $StandAlone$ holons may interact only with the *heads* of the super-holon.

2.3 Holarchy example

In order to illustrate our framework we take an example and describe it with holonic concepts. This example consists in a simplified University. Imagine that we model the university as composed of Departments and research Laboratories. They are in turn composed of Professors and Researchers respectively. If we isolate the Computer Science and Laboratory Holon and their components from the university example and we add these holonic roles, we obtain figure 3. *Part* role players for the laboratory represent researcher that belong only to the laboratory, e.g. full time researchers. On the other hand, some researcher may, in addition to their activities in the laboratory, give lectures in the computer science department. These holons, like holon RP in figure 3, belong to both super-holons simultaneously and thus they play the $MultiPart$ role. In this example, the department and laboratory directors would be the $Heads$ of the C.S. Department and the laboratory respectively.

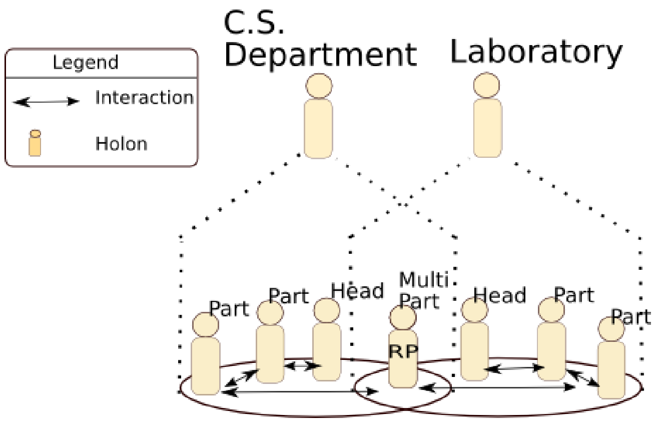

Fig. 3. Department and Laboratory Holons

As we mentioned earlier other organization will be used to specify domain dependant interactions (e.g. a *Lecture Organization* to describe how professors interact with their students) [22].

Based on these *holonic* roles –Head, Part and MultiPart– we have defined mechanisms to handle holons dynamics. They are based upon the affinity and satisfaction between holons. The notion of *Affinity* was inspired by a technique used for the Artificial Immune System [8]. The term *Satisfaction* has often been used to represent the gratification of an agent concerning its current state or the progress of its goals/tasks [6, 23].
The affinity between holons must be defined according to the domain of the application. The affinity measures, according to the application's objectives, the compatibility of two holons to work together toward a shared objective.
The *compatibility* of two holons means that they can provide help to each other to

progress towards their goals. Based on the application's objective, we define a set of rules that allow us to evaluate this *compatibility.* Generally speaking, we can say that two holons are compatible if they have shared goals and complementary services.

Using these two notions, holons are able to decide when they should join or leave a super-holon (satisfaction) and with which super-holon to merge with (affinity). Holons can then move from one super-holon to another as the system evolves.

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 4. Each domain is separated by a dashed line.

3.1 Problem Domain

The PASSI, and then HoloPASSI, methodologies are driven by requirements. So the starting point are the *Functional Requirements* and *Non Functional Requirements* concepts. (Functional) Requirements can be identified by using classical techniques such as use cases. Each requirement is associated to an *Organization* (see figure 4). An *Organization* is defined by a set of *Problem Roles*, their *Interactions* and a common context. The associated context (and therefore 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 by the event produced by a first role, perceived by a second one, and the reaction(s) produced by the second role (this sequence can be iterated in the same interaction). 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 5 details an example of an organization and its associated ontology. The *Project Management* organization (see figure 5(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 5(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" [14]. The notion of capacity

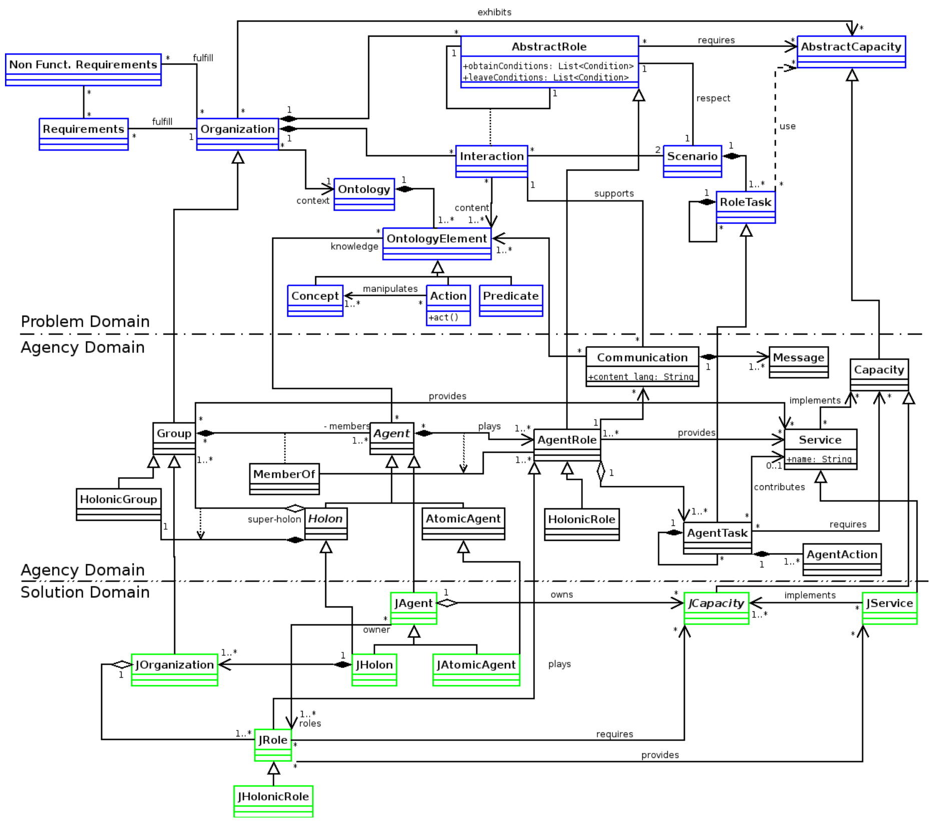

Fig. 4. The Organizational Meta-Model of HoloPASSI

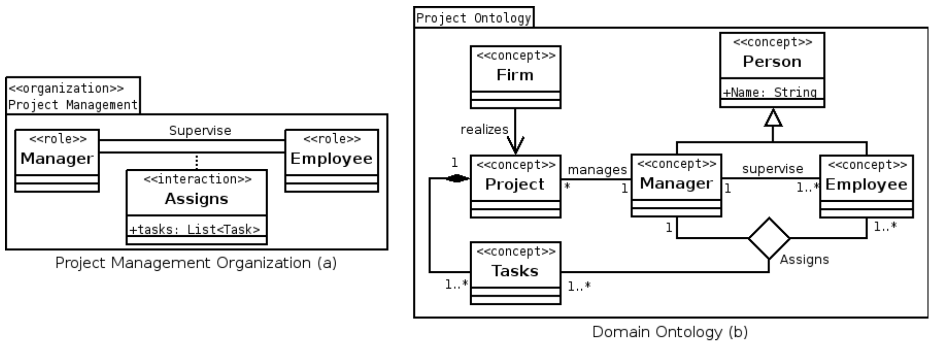

Fig. 5. Organization and Ontology description using two specific UML profiles for class diagram, and an example of the concrete structure of a super-holon

Published in Proc. of the Fifth European workshop on Multi-Agent Systems(EUMAS'07). 13–14 December, 2007. Hammameth (Tunisia).

was introduced to control and exploit these additional 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. A *role* may require that individuals playing it have some specific capacities to properly behave as defined. An individual must know a way of realising all required capacities to play a *role*. In other words, the main objective of the *capacity* is the definition of generic role behaviours by identifying which know-how a role requires from the individual that will play it; The *capacity* element constitutes a layer between the *role* behaviour and the future entities which will want to play this *role*. Basing the description of role behaviour on capacities gives to the role more genericity and modularity.
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.
A *Problem Role* 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 provided 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 preceeding 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. It is in the role of *Employee* to respect him and obey him. Another important aspect is that the role (and not the individual, like an agent or an 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 *Problem Role* is to contribute to (a part of) the requirements of the organization within which it is defined. The behavior of a *Problem Role* is specified within a *Scenario*. Such a scenario describes how a goal can be achieved. 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 4 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*) *AbstractGroup*: it is a specialization of the *Organization*. It is used to model groups of (*Abstract*)*Agents* that cooperate towards the achievement of a goal. This element is further specialized in the *AHolonicGroup* element that is a group devoted to contain roles taking care of the holon internal decision-making process (composed-holon's government). (*ii*) *AgentRole*: it is the specialization of *ProblemRole*. An *AgentRole* interacts with the others using communications (that are a more refined way for interacting of the simple Interactions allowed to the ProblemRole). Several *AgentRoles* are usually grouped in one *AbstractAgent* that is in turn a member of the *AbstractGroup*. An *AgentRole* can be responsible for providing one of more services. (*iii*) *AbstractCapacity*: it is the specialization of the *ProblemCapacity*. 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*: it 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 *AbstractAgent*. An (*Abstract*)*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 (*Abstract*)*Agent* context is given by the knowledge, the capacities and the environment the roles share for the simple fact of being part of the same agent. This, for instance, 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 in so doing 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 referred as an AbstractAgent because that such an entity is still not an implementing element but rather it needs of further refinement; only when it will become a JAgent (in the Solution Domain) it can really be coded.
Figure 6(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 defines 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 6(a).
The concept of *AbstractHolon* is specialized from the *AbstractAgent*. 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 (for instance in another organization i.e. another holon) and be composed by 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 interacts and coordinate their actions to fulfill the super-holon tasks and objectives. An atomic (non composed) holon is an AtomicAgent. Figure 6(b) illustrates this definition of holon.

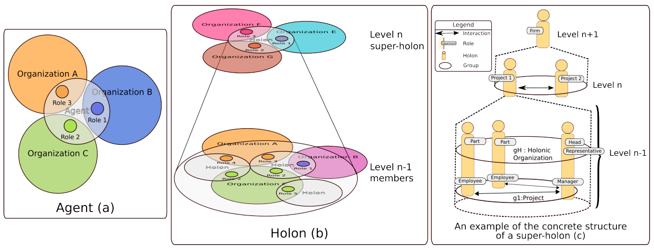

Fig. 6. Agent and Holon symbolic representation

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). We have adopted this management structure due the wide range of configurations it allows. In a moderated group, a subset of the members will represent all the sub-holons with the outside world. Four roles are defined to describe the status of a member inside a super-holon. These roles inherit from the Holonic Role introduced in figure 4 and aren't represented in this figure for clarity reasons.

***Head*, decision maker :** it represents a privileged status confering a certain level of authority..

***Representative*, interface of the holon :** it's a part of the visible face of a super-holon, he's an interface between the outside world (same level or upper level) and other members of the holons. He will represent others members for taking certain decisions or accomplishing certain tasks (i.e. recruit member, translate information). The *Representative* can be played by more than one member at the same time.

***Part* :** Classical members. Normally in charge of doing task affected by head, a *Part* can also have an administrative load, and being implied in the decision making process. It depends on the configuration choosen to model the super-holon. The *Part* role represents members belonging to only one super-holon.

***Multi-Part* :** extension of *Part*, this role is played by sub-holons shared by more than one super-holon.

Depending on the level of abstraction a super-holon can be seen 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 related to that contract in the production organization at level n-1); the AbstractAgent existence in this case enables the interactions among the different roles. ii) (external) interactions (mostly communications) between roles (at different layers) of different agents. For instance the Head (layer n) responsible for accepting a contract asks to worker roles (layer n-1) of providing the service.
Figure 5(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.
In order to maximize its goal achievement expectation, an agent has to be able to estimate the competences of its future partners and to identify the most appropriate collaborators. The AbstractCapacity concept allows us to represent the competences of an agent or of a set of agents. An AbstractCapacity describes what an *(Abstract)Agent* should be able to do in order to satisfy the requirements it is responsible for. This means that the set of AbstractCapacities obtained by refining the ProblemCapacity of the Problem Domain, becomes the specification of the system requirements in the Agency Domain. Indeed, AbstractCapacities 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 others 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 others 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 [11], JADE, FIPA-OS, in Janus the concept of Role is considered as a first class entity. It thus allows 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 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 consist 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. A framework for HMAS analysis and design is proposed and supported by a development platform, namely *Janus*.
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's also able to support more "traditional" multiagent system.
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

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

applications. Future works will deepen the meta-model concepts and associate a methodology to guide the developer during his work of modelling and implementing a complex (and possibly holonic) multi-agent system.

References

1. E. Adam. *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.
2. Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Jade - a fipa-compliant agent framework. Technical report, CSELT, 1999.
3. H. Van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters. Reference architecture for holonic manufacturing systems: Prosa. *Computers in Industry*, 37:255–274, 1998.
4. H.J. Bürckert, K. Fischer, and G.Vierke. Transportation scheduling with holonic mas - the teletruck approach. In *Conf. on Practical Applications of Intelligent Agents and Multiagents*, pages 577–590, 1998.
5. K. Carley and M. Prietula, editors. *Computational Organization Theory.* 1994.
6. Kelly Christine Correa e Silva Fernandes. *Systèmes Multi-Agents Hybrides: Une Approche pour la Conception de Systèmes Complexes.* PhD thesis, Université Joseph Fourier- Grenoble 1, 2001.
7. M. Cossentino. *Agent-Oriented Methodologies*, chapter IV : From Requirements to Code with the PASSI Methodology, pages 79–106. 2005.
8. Dipankar Dasgupta. *Artificial Immune Systems and Their Applications.* Springer-Verlag, November 1998.
9. V. Dignum and F. Dignum. 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.
10. M.E. Epstein and R. Axtell. *Growing Artificial Societies: Social Science from the Ground Up.* MIT Press, 1996.
11. J. Ferber, O. Gutknecht, and F. Michel. From agents to organizations: an organizational view of multi-agent systems. In *AOSEIV@AAMAS03*, LNCS 2935, pages 214–230, 2004.
12. Les Gasser. Mas infrastructure definitions, needs, prospects. In *Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems*, 01.
13. Christian Gerber, Jörg H. Siekmann, and Gero Vierke. Holonic multi-agent systems. Technical Report DFKI-RR-99-03, Deutsches Forschungszentrum für Künztliche Inteligenz - GmbH, Postfach 20 80, 67608 Kaiserslautern, FRG, May 1999.
14. J.H. Holland. *Hidden order: how adaptation builds complexity.* 1995.
15. C.A. Iglesias, M. Garijo, and J. Gonzalez. A survey of agent oriented methodologies. In *Workshop on Intelligent Agents V: Agent Theories, Architectures and Languages (ATAL-98)*, volume 1555, pages 313–327. Springer-Verlag, 1999.
16. N.R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 177(2):277–296, 2000.
17. Francisco Maturana. *MetaMorph: an adaptive multi-agent architecture for advanced manufacturing systems.* PhD thesis, The University of Calgary, 1997.
18. S. Poslad, P. Buckle, and R. Hadingham. FIPA-OS: the FIPA agent platform available as open source. In *Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 2000)*, pages 355–368, 2000.

19. M.J. Prietula, K.M. Carley, and L.E. Gasser. *Simulating Organizations: Computational Models of Institutions and Groups.* AAAI Press, 1998.
20. S. Rodriguez, N. Gaud, V. Hilaire, S. Galland, and A. Koukam. An analysis and design concept for self-organization in holonic multi-agent systems. In *Engineering Self-Organising Systems*, volume 4335 of *LNAI*, pages 15–27. Springer-Verlag, 2007.
21. S. Rodriguez, V. Hilaire, and A. Koukam. Towards a methodological framework for holonic multi-agent systems. In *Workshop of Engineering Societies in the Agents World*, pages 179–185, 2003.
22. Sebastian A. Rodriguez. *From analysis to design of Holonic Multi-Agent Systems: a Framework, methodological guidelines and applications.* PhD thesis, Université de Technologie de Belfort-Montbéliard, 2005.
23. O. Simonin and J. Ferber. Modélisation des satisfactions personnelle et interactive d'agents situés coopératifs. In *JFIADSMA'01: 9èmes Journées Francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents*, pages 215–226, 2001.
24. M. Ulieru and A. Geras. Emergent holarchies for e-health applications: a case in glaucoma diagnosis. In *IEEE IECON 02*, volume 4, pages 2957– 2961, 2002.
25. J. Wyns. *Reference architecture for Holonic Manufacturing Systems - the key to support evolution and reconfiguration.* PhD thesis, Katholieke Universiteit Leuven, 1999.
26. F. Zambonelli, N. Jennings, and M. Wooldridge. Developing multiagent systems: the gaia methodology. *ACM Trans. on Software Engineering and Methodology*, 12(3), 2003.