Formal Specification of Holonic Multi-agent Systems Framework

Sebastian Rodriguez, Vincent Hilaire, and Abder Koukam

UTBM,
Systems and Transports Laboratory,
90010 Belfort Cedex,
France,
Tel: +33 384 583 009, Fax +33 384 583 342
vincent.hilaire@utbm.fr

Abstract. Even if software agents and multi-agent systems (MAS) are recognized as both useful abstractions and effective technologies for modeling and building complex distributed applications, they are still difficult to engineer. When massive number of autonomous components interact it is very difficult to predict the behavior of the system and guarantee that the desired functionalities will be fulfilled. Moreover, it seems improbable that a rigid unscalable organization could handle a real world problem. This paper presents a holonic framework where agents exhibit self-organization according to the tasks at hand. We specify formally this framework and prove some properties on the possible evolutions of these systems.

Keywords: Holonic Multi-Agent Systems, self-organised system, formal specification, model checking.

1 Introduction

Even if software agents and multi-agent systems (MAS) are recognized as both useful abstractions and effective technologies for modeling and building complex distributed applications, they are still difficult to engineer. When massive number of autonomous components interact it is very difficult to predict the behavior of the system and guarantee that the desired functionalities will be fulfilled. Moreover, it seems improbable that a rigid unscalable organization could handle a real world problem. The aim of this paper is to present a formally specified framework for holonic MAS which allows agent to self-organise. We prove some pertinent properties concerning the self-organising capabilities of this framework.

The term holon was originally introduced in 1967 by the Hungarian Philosopher Arthur Koestler[7] to refer to natural or artificial structures that are neither wholes nor parts in an absolute sense. According to Koestler, a holon must respect three conditions: (1) being stable, (2) having the capability of autonomy

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3516, pp. 719–726, 2005.

and, (3) being capable of cooperation. Holonic organizations have proven to be an effective solution to several problems associated with hierarchical self organized structures (e.g. [10], [1], [12]). In many MAS applications, an agent that appears as a single entity to the outside world may in fact be composed of several agents. This hierarchical structure corresponds to the one we find in Holonic Organizations. Frameworks have been proposed to model specific problem domains, mainly in Flexible Manufacturing Systems (FMS) and Holonic Manufacturing Systems (HMS), such as PROSA (PROSA stands for Product-Resource-Order-Staff Architecture) , [14] and MetaMorph[9]. However, the Holonic paradigm has also been applied in other fields such as e-health applications [11].

Our framework isn't application domain dependent so it can be easily reused. This framework is based upon organizational concepts which have been successfully used in the MAS domain [6,13]. We base our approach on the Role-Interaction-Organization (RIO) Methodology. RIO uses a specific process and a formal notation OZS that is described in [6]. OZS is a multi-formalisms notation that integrates in Object-Z classes [3] a statechart [5]. OZS classes have then constructs for specifying functional and reactive aspects. We have defined a formal semantics for OZS [4]. This semantic is based upon the translation of Object-Z and statecharts into transition systems and allows the use of theorem proover and model checker.

The paper is organized as follows : section 2 presents RIO and specifies the holonic framework, the section 3 describes proven properties and eventually section 4 concludes.

2 RIO and Holonic Framework Overview

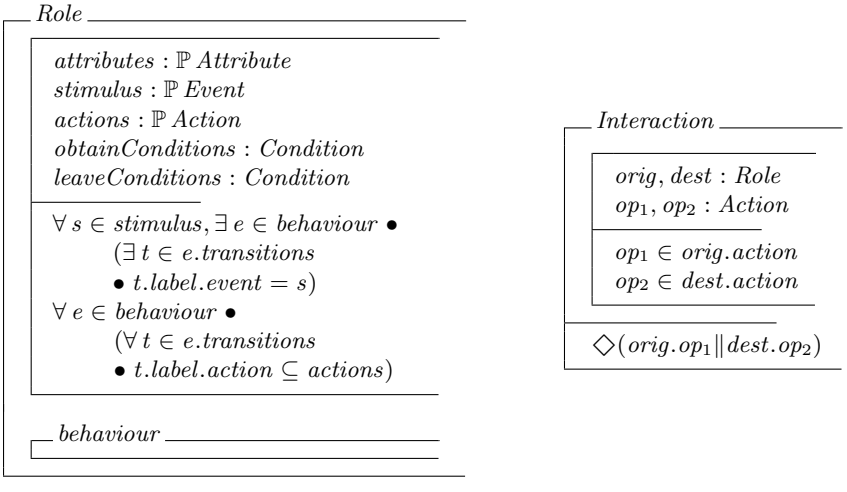
In this section we present the RIO framework and its extension for Holonic MAS. We use the OZS formalism which is based upon the integration of Object-Z and statecharts. Object-Z is an object oriented extension of Z and thus uses the set theory and first order predicate logic. Statecharts add hierarchy of state, parallelism and broadcasted communication to finite state automata. Each concept of the RIO Framework is specified by an OZS class.

2.1 RIO Classes

The RIO Methodology is based on three main concepts. A *Role* is an abstraction of the behaviour of an acting entity. For example, we can see an university as an organization with several roles such as *Researcher*, *Professor*, etc.

We have chosen to specify it by the *Role* class. This class represents the characteristic set of attributes whose elements are of [*Attribute*] type. These elements belong to the *attributes* set. A role is also defined by stimulus it can react to and actions it can execute. They are specified by *stimulus* set and *actions* set respectively. The [*Attribute*], [*Event*] and [*Action*] types are defined as given types which are not defined further.

The reactive aspect of a role is specified by the sub-schema *behaviour* which includes a statechart. It is to say that the *behaviour* schema specifies the different states of the role and transitions among these states. The *obtainConditions* and *leaveConditions* attributes specify conditions required to obtain and leave the role. These conditions require specific capabilities or features to be present in order to play or leave the role. Stimuli, which trigger reactions in the role behaviour, must appear in one transition at least. The action belonging to the statechart transitions must belong to the *actions* set.



An *interaction* is specified by a couple of role which are the origin and the destination of the interaction. The role *orig* and *dest* interacts by the way of operations op_1 and op_2. These operations are combined by the $\|$ operator which equates output of op_1 and input of op_2. The $\Diamond$ symbol is a temporal logic operator which states that eventually the predicate is true. In order to extend interaction to take into account more than two roles or more complex interactions involving plan exchange one has to inherit from the *Interaction* class. For example, the two roles *Proferssor* and *CoursePlanner* interact at the beginning of the year. The *CoursePlanner* sends the schedule to the *Professor* role.

2.2 HMAS Framework

In this section we present a set of roles which contitutes the kernel of the HMAS Framework. They describe the behaviour and interactions of the components of a holonic organization: holons. The holons inside a holonic organization, may join or create other holons to colaborate towards a shared goal. Inside a Holon there is one that acts as the *representative* (Head) and others as members (Part) of the Holon.

In order to enable holons to dynamically change their roles, we define a satisfaction based on the progress of his current task. This satisfaction, called *instant*

satisfaction, depends on the played role and is calculated using the following definition, where R_i is the role played by the holon i.:

Self Satisfaction (SS_i). Satisfaction for the agent i produced by his own work.
Collaborative Satisfaction (CS_i^H). Satisfaction produced for the Agent i by his collaboration with other member agents of the Holon H,
Instant Satisfaction (IS_i). Satisfaction produced by the work done up to the moment

$$\forall\, i \in HMAS\ IS_i = \begin{cases} CS_i + SS_i & if\ R_i = Part \vee R_i = Head \\ SS_i & if\ R_i = Stand - Alone \end{cases} \quad (1)$$

2.3 Framework Specification

The inheritance relationships between the different roles is presented in the figure 1.

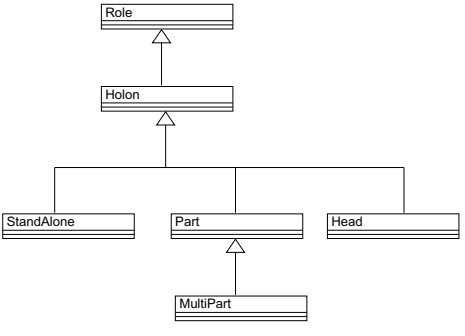


Fig. 1. Inheritance relationships between HMAS roles

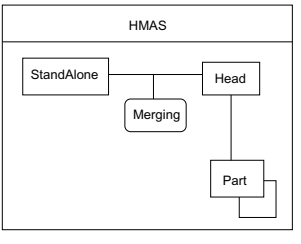


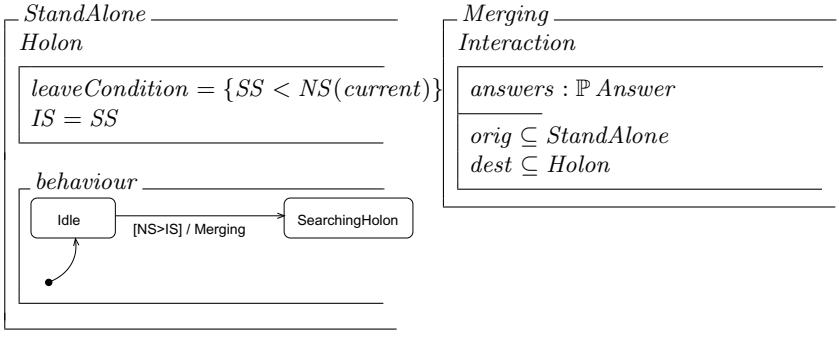
Fig. 2. RIO diagram of the HMAS framework

The RIO diagram of the figure 2 presents the possible interactions between the different roles. A *StandAlone* role player may interact with *Heads* in order to enter a specific holon. This interaction is specified by the *Merging* class which inherits from *Interaction*. *Part* role players interact with their *Head* during the holon's life. These interactions are commands or requests. The *Holon* class inherits from *Role* class and defines the generic elements for all Holonic role players. These elements are the different satisfaction criterions defined in the section 2.2. *SS* stands for self-satisfaction and *IS* stands for instant satisfaction. The current task of the holon is specified by *current* an element of a given type [*Task*]. For each task, the function *NS* associate a threshold. It is the minimum value for the self-satisfaction of the holon in order to pursue the current task. The available services for other holons are specified by the function *availableServices*. All following roles inherit from *Holon* and add specific attributes, operations and behaviours.

Holon
Role

$satisfied : \mathbb{B}$
$SS, IS : \mathbb{R}$
$current : Task$
$NS : Task \mapsto \mathbb{R}$
$availableServices : Service \to \mathbb{B}$

A *StandAlone* role is the entry point of an holonic organization. Each holon which is satisfied by its progress and with no engagement with other holons, plays this role. As soon as its satisfaction is less than the threshold defined by the *NS* function, the *StandAlone* holon searches a holon to merge with.



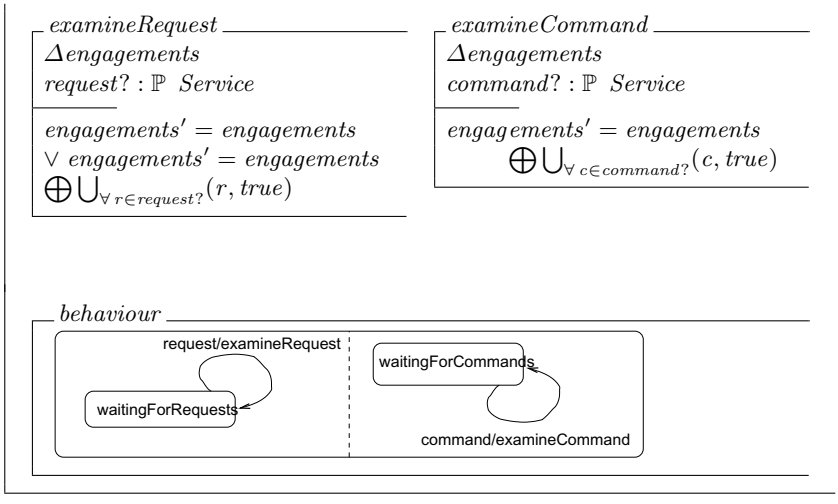
The [Answer] given type specifies the answers given by heads in response to a merge request. This interaction requires that the origin of the interaction is a holon playing the StandAlone role and the destination can be any Holon (a Head or another StandAlone holon).

The *Part* class specifies a role which is part of a holon . A holon part of a bigger holon knows the other members of the holon which are the elements of the *others* set. It also knows the head of its holon, *myHead*. The *Part* role has one more satisfaction criterion than a Holon, *CollaborativeSatisfaction*. It may also be engaged on some of its available services. These engagements are specified by the *engagements* function

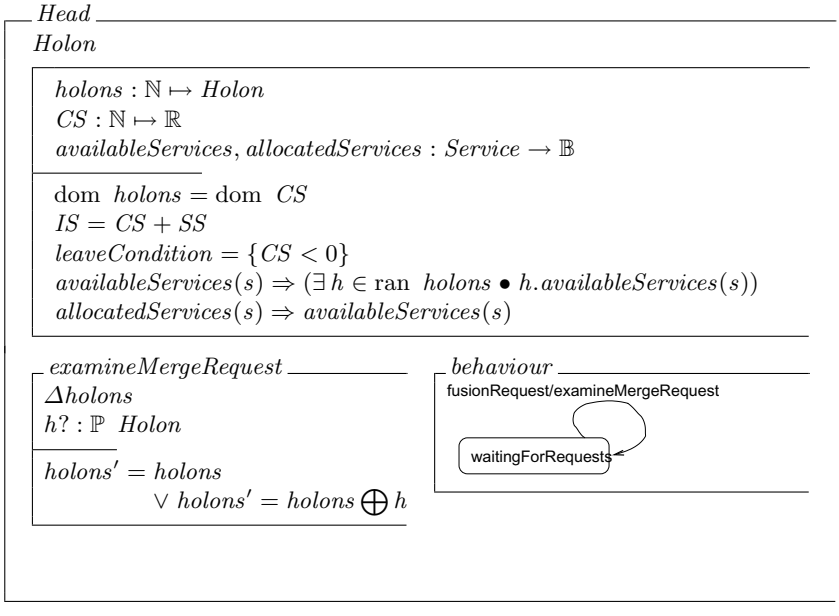
Part
Holon

$others : \mathbb{P}\, Holon, myHead : Head$
$CS : \mathbb{R}$
$engagements : Service \mapsto \mathbb{B}$

$IS = CS + SS$
$\mathrm{dom}\ engagements \subseteq \mathrm{dom}\ availableServices$
$leaveCondition = \{CS < 0\}$



Finally, a holon may play the *Head* role. In this case this holon is the representative of the members of the Holon. Thus, it will examine the request of other holons to join his Holon.



3 Proofs

OZS semantics [4] is based upon transition systems as defined in [8]. It means that for each OZS specification there is an associated transition system. This

transition system represents the set of possible computations the specification can produce. With such transition systems and software tools like SAL [2] one can verify specification properties.

Among the tools proposed by SAL we have chosen the SAL model checker which enables the verification of the satisfiability of a property. The SAL model-checker proves or refutes validity of Linear Temporal Logic (LTL) formulas relatively to a transition system. To establish the satisfiability of history invariant H one must actually establish that $\neg H$ is not valid. This technique is the simplest to use but is limited by the specification state space.

The first property we have proven may be interpreted as "if the holon's satisfaction is not enough then it will try to merge". This property is specified as follows :

$$\forall a : HMASAgent \bullet a.is < a.ns \Rightarrow \Diamond(StandAlone \notin a.playing)$$

It states that for all agent of an holonic MAS if its instant satisfaction becomes less than the necessary satisfaction eventually this agent will not play the StandAlone role. It will try to be engaged in a holon and thus play either the Part or the Head role. In other words, if the agent can not accomplish its task alone, it will create a holon to cooperate with other having a share objective.

The second property we have proven may be interpreted as "if the holon's satisfaction evolves and becomes less than necessary satisfaction the system will try to reorganise". This property is specified as follows :

$$\forall a : HMASAgent \bullet instant = a.playing \wedge a.is < a.ns \Rightarrow \Diamond(a.playing \neq instant)$$

It states that for all agent of an holonic MAS whatever role it plays if its instant satisfaction becomes less than the necessary satisfaction eventually this agent will change the role it is playing.

4 Conclusion

In this paper we have presented a framework for the design of Holonic MAS. This framework is based upon roles the agent can play and satisfactions which characterise the progression of the agent towards achievement of its goals. We have presented this framework through its formal specification using the OZS formalism. The semantics of this formalism enables the verification of properties. We have proven two pertinent properties for this framework. The first property we have proven may be interpreted as "if the holon's satisfaction is not enough then it will try to merge". It's an important property of such self-organised systems. Indeed, it means that if one holon is unsatisfied by its current achievements it will try to merge to find complementary capabilities or services.

The second property may be interpreted as "if the holon's satisfaction evolves and becomes less than necessary satisfaction the system will try to reorganise". This property ensures that if the current holarchy doesn't correspond to the current context it will evolve in order to find a better one.

Other frameworks and methodologies have been proposed [9,14] and, although they have shown to be effective inside specific domains, a more generic framework is needed. Indeed, it is difficult to design a Holonic MAS without clear and specific definitions that can lead from the analysis in terms of holon to the design of the system. Moreover, a framework with predictable properties, such as those we have proven, constitutes a solid foundation for the development of Holonic MAS.

References

1. Hans-Jörgen Bürckert, Klaus Fischer, and Gero Vierke. Teletruck: A holonic fleet management system.
2. Leonardo de Moura, Sam Owre, Harald Rueß, John Rushby, N. Shankar, Maria Sorea, and Ashish Tiwari. SAL 2. In Rajeev Alur and Doron Peled, editors, *Computer-Aided Verification, CAV 2004*, volume 3114 of *Lecture Notes in Computer Science*, pages 496–500, Boston, MA, July 2004. Springer-Verlag.
3. Roger Duke, Paul King, Gordon Rose, and Graeme Smith. The Object-Z specification language. Technical report, Software Verification Research Center, Departement of Computer Science, University of Queensland, AUSTRALIA, 1991.
4. Pablo Gruer, Vincent Hilaire, Abder Koukam, and P. Rovarini. Heterogeneous formal specification based on object-z and statecharts: semantics and verification. *Journal of Systems and Software*, 70(1-2):95–105, 2004.
5. David Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, June 1987.
6. Vincent Hilaire, Abder Koukam, Pablo Gruer, and Jean-Pierre Müller. Formal specification and prototyping of multi-agent systems. In Andrea Omicini, Robert Tolksdorf, and Franco Zambonelli, editors, *Engineering Societies in the Agents' World*, number 1972 in Lecture Notes in Artificial Intelligence. Springer Verlag, 2000.
7. Arthur Koestler. *The Ghost in the Machine*. Hutchinson, 1967.
8. Zohar Manna and Amir Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer, 1995.
9. Francisco Maturana. *MetaMorph: an adaptive multi-agent architecture for advanced manufacturing systems*. PhD thesis, The University of Calgary, 1997.
10. Sebastian Rodriguez, Vincent Hilaire, and Abderrafia Koukam. Towards a methodological framework for holonic multi-agent systems. In *Fourth International Workshop of Engineering Societies in the Agents World*, Imperial College London, UK (EU), 29-31 Octubre 2003.
11. M. Ulieru and A. Geras. Emergent holarchies for e-health applications: a case in glaucoma diagnosis. In *IECON 02 [Industrial Electronics Society, IEEE 2002 28th Annual Conference of the]*, volume 4, pages 2957– 2961, 2002.
12. Gero Vierke and Christian Russ. Agent-based configuration of virtual entreprises.
13. Michael Wooldridge, Nicholas R. Jennings, and David Kinny. A methodology for agent-oriented analysis and design. In Oren Etzioni, Jörg P. Müller, and Jeffrey M. Bradshaw, editors, *Proceedings of the Third Annual Conference on Autonomous Agents (AGENTS-99)*, pages 69–76, New York, May 1-5 1999. ACM Press.
14. J. Wyns. *Reference architecture for Holonic Manufacturing Systems - the key to support evolution and reconfiguration*. PhD thesis, Katholieke Universiteit Leuven, 1999.