EXPERIENCING AUML IN THE GAIA METHODOLOGY

Luca Cernuzzi
DEI Universidad Católica "Nuestra Señora de la Asunción", Campus Universitario, C.C. 1683, Asunción, Paraguay

Franco Zambonelli
Dipartimento di Scienze e Metodi dell'Ingegneria, Università di Modena e Reggio Emilia, Reggio Emilia, Italy

Keywords: Agent Oriented Methodologies, Gaia, AUML, Open Multi-Agent Systems

Abstract: In the last few years a great number of AOSE methodologies have been proposed, some of which centered on organizational aspects to better capture the behavior of agents societies. Those methodologies may be considered very useful for modeling open systems composed of a great number of interacting autonomous agents. Gaia exploits organizational abstractions to provide clear guidelines for the analysis and design of complex and open Multi-Agent Systems (MAS). However, the notation of the Gaia methodology is probably less powerful (and perhaps less acceptable for industry solutions) than others (like AUML). In this perspective, this aims at performing a preliminary exploration towards the potential application of the AUML notation into the Gaia methodology: it explores the above issues using an application example and pays specific attention to the problem of modeling the complexity of open MAS and emergent behaviors.

1 INTRODUCTION

In the last few years a great number of AOSE methodologies have been proposed trying to model specific agents architectures, or extending accepted techniques and methods from traditional OO engineering paradigm. However, they are dependent on abstractions and tools that may be unsuitable for modeling new trends in agent-based systems..

Other researchers have recently proposed to identify appropriate abstractions for Multi-Agent Systems (MAS), and to propose software engineering methodologies accordingly. Some of such methodologies are Gaia (Zambonelli et al., 2003), MESSAGE (Caire et al., 2001), TROPOS (Giunchiglia et al., 2002), and ROADMAP (Juan et al., 2002). All of these methodologies share the idea that a MAS may be viewed as an organized society of individual agents with their roles and different kinds of interactions among them according to specific protocols that are related to the roles of the interacting agents. Some of those methodologies introduce different abstractions however, very few of them explicitly focus on organizational ones.

Among those, we consider that the new Gaia proposal (Zambonelli et al., 2003) (the first proposal was centered on closed communities of cooperating agents) may be specially significant when used in the analysis and design of open MAS. In effect, Gaia exploits organizational abstractions that are necessary for designing and building systems in complex, open environments. Our considerations are reinforced by the evaluation of Gaia as presented in (Sturm and Shehory, 2003).

However, probably due to its simplicity, Gaia notations are poor and far to be widely accepted for industry solutions (unlike UML in OO software engineering). This aspect seems to be quite evident in the specification of agent interactions (Sturm and Shehory, 2003). In fact, the Gaia protocol model while rigorously specify actors and input and output of the protocol, use informal natural language to specify its semantics, including the dependency and speech-act interactions involved. Given this, it may be very interesting to re-use richer notations founded on a more consolidated paradigms, like object orientation with its proven techniques and solutions, and adapt them to MAS specification needs.

In this paper we try to apply AUML to Gaia. AUML is not per se a thorough methodology However it models in a very rich and expressive way the Agents Interaction Protocols (AIP) that constitute a central aspect for open MAS. Thus, AIP may naturally replace the Gaia protocols model.

Cernuzzi L. and Zambonelli F. (2004).
EXPERIENCING AUML IN THE GAIA METHODOLOGY.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 283-288
DOI: 10.5220/0002618802830288

The remainder of this paper is organized as follow. Section 2 introduces the idea of open MAS and an illustrating example. Section 3 synthetically presents the new Gaia proposal. Section 4 analyzes the AUML notation and describes how it is possible to use it within Gaia by means of the analysis specification of the illustrating example. Section 5 discusses related work in the area. Finally, Section 6 concludes and outlines open research directions.

2 OPEN MAS: AN EMERGENT PARADIGM

Agents may be conceived as stand-alone entities that accomplish particular tasks on behalf of a user (e.g., personal digital assistants, e-mail filters, buy assistants). However, usually, the environments where agents accomplish their tasks are populated with other agents. In these MAS, the global behavior of the system derives from the interaction (co-operation, co-ordination, negotiation, etc.) among the existing agents.

So, a MAS is based on two types of aspects: individual and collective. The former set comprises those aspects found in classical systems but is the latter that includes those constituting a new dimension: the organizational aspects.

It may be possible to distinguish between two main classes of MAS:

- distributed problem solving systems in which agents are explicitly designed to co-operatively achieve a specific goal in a closed way. That is, all agents are defined a priori, they are co-operative to each other and, therefore, they can trust one another during interactions;
- open systems in which each agent has its own aims and objectives and is not designed to share a common goal with others. Moreover, since the objectives of different agents may be in opposition (competitive), agents should not necessarily trust each other.

Actually, a growing trend in agent applications focuses on open MAS based on a great number of agents whose interactions may produce an emergent behavior and distributed intelligence. Examples of them may be found in most Internet-based systems (e.g., agents for information retrieval and web service agents) in which the agents have to exploit services, knowledge, and capabilities offered by other agents. Also, it includes all those systems that involve interactions between agents on behalf of different stakeholders (e.g., e-commerce agents).

Critical aspects in open MAS applications are the organizational structure (if any) and specially the organizational rules that control the behavior of self-interested agents. For this reason it is very important to focus on methodologies that may support the modeling of those aspects, among the agents with their roles and interaction.

2.1 Illustrating Example: the Agents Marketplace

A representative example of open MAS are agent marketplaces. A marketplace can be considered any place where some proactive entities (e.g. persons, enterprises, or computational organizations) can go to put on sale services and/or goods and, vice versa, where other proactive entities go to buy good and service they are in need of.

Currently, marketplaces may exist in the form of Web sites, where specific communities of users interested in specific classes of goods can meet and arrange their commercial transactions in an interactive way and without the constraint of physical co-location. Still, the need of some form of interaction limits the capability and widespread acceptance of such Web sites, due to the amount of time which may be required for looking and contracting for goods.

Agents can effectively accomplish the tasks of looking for goods on behalf of clients, selling goods on behalf of providers, and negotiating with each other directly, without direct intervention of involved humans and/or enterprises (apart from the interaction with its agents). Thus, we can envision the Internet will be populated with a variety of special-purpose marketplaces.

In such marketplaces, agents interested in specific classes of goods will meet to access an environment made up of "sales offers" (possibly based on an auction model) and of "wanted requests". Such agents, in a given marketplace, will form an organization made up of agents playing the roles of "client" and "provider" in a wanted request model as well as those of “bidder” and “supplier” in an auction-based model, and interacting with each other according to specific negotiation patterns. The intrinsic openness of the scenario, where different agents may enter the marketplaces to negotiate, introduces the issues of controlling negotiations in a proper way so as to avoid agents, which have a self-interested behavior, cheat to with each other.

3 GAIA IN A NUTSHELL

A first overview of the Gaia methodology with its models and their relationships is presented below.

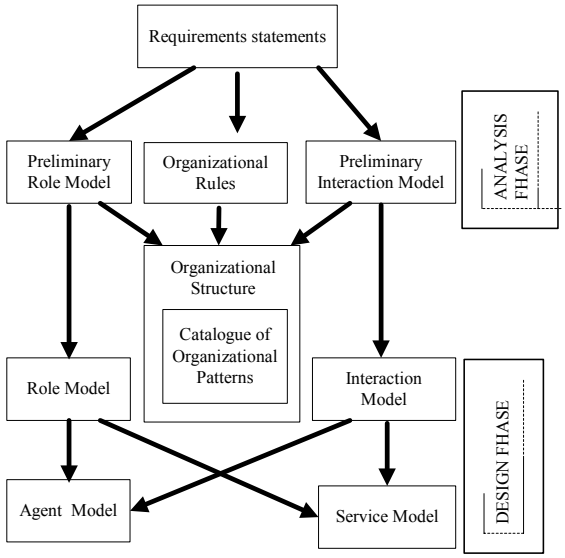

Figure 1: The Gaia Methodology

Gaia does not consider the collection of requirements stage, starting right at the analysis phase which aims to collect all the specifications of the computational organization of the MAS. This includes the identification of (see Figure 1):
- The goals of the organizations that constitute the overall system and their expected global behavior. This will be refined during the design phase in order to identify useful decomposition of the global organization into sub-organizations.
- The preliminary roles model that captures the basic skills required for different types of agents to reach the organization goals. At this stage, Gaia's notion of roles is abstract from any mapping into agents (this issue will be considered in the design phase) and the analyzer have to avoid the imposition of a specific organizational structure.
- The preliminary interaction model that captures the basic needed interactions from the identified preliminary roles. Also this model, possibly not being completely defined, must abstract away from the organizational structure.
- The organizational rules that govern the organization in its global behavior. Such rules impose constraints on the execution activities of roles and protocols. Moreover, they are fundamental to efficiently specify the openness and the general behavior of the developing MAS.

The output of the analysis phase consists of three basic models: (i) a preliminary roles model; (ii) a preliminary interactions model; and (iii) a set of organizational rules.

The design phase is aimed at producing a complete specification of the MAS. To this end, the design phase contemplates the following sub-phases:
- Definition of the overall organizational structure considering the organizational rules. At this stage it is important to exploit well-known organizational patterns that may also influence the design of the final interactions model.
- Considering the adopted organizational structure a revision and completion of the preliminary role and interaction models.
- Definition of the agent model specifying agent types (a set of agent roles) and agent instances.
- Definition of the services model which specify the main services (blocks of activities with their conditions) related with the agent roles.

The specification of agents with their roles and the interactions among them and with the environment, are not enough to capture the complex and emergent behavior derived from many self-interested agents applications. For this reason, Gaia spent additional effort in modeling the organizational structure as well as the organizational rules. Those aspects, as already stated in section 2, are very relevant in order to specify open MAS.

4 INTEGRATING AUML INTO GAIA

Different attempts in the past few year have tried to extend UML notation for agent-based systems one of them is Agent UML (AUML). AUML is not per se a thorough methodology, however it builds on the acknowledged success of UML in supporting industrial-strength software engineering.

The core part of Agent UML is the Agents Interaction Protocol (AIP), that constitute a central aspect for open MAS, specified by means of protocol diagrams. Protocol diagrams extensions to UML include agent roles, multithreaded lifelines, extended message semantics, parameterized nested protocols, and protocol templates.

Still, the key ideas of AUML that may be integrated into Gaia methodology are:
- The protocol can be regarded as a whole entity (expressed by mean of an enriched sequence diagram) and treated as a package. AUML considers an AIP as a template, whose parameters may be roles, constraints, and communication acts. This template approach expresses in a more compact way and UML-like notation the same semantics of the Gaia protocol notation, but it is easier to visualize.
- Each protocol implies inter-agent interactions that are described using sequence diagrams, activity diagrams, and statecharts. AUML extends sequence

diagram notation in order to represent Agents (and eventually their Class) and their Roles, and to support concurrent threads of interactions. The activity diagram, particularly useful for complex interaction protocols that involve concurrent processing, and statecharts are used to specify the internal behavior of an agent.

AUML proposes other extensions to UML such as packages with agent interfaces, deployment diagrams indicating mobility, emergence, etc. However, those notations are less richer than those proposed for AIP and some of them are poor compared to Gaia notations. For example, the role specification in Gaia is more expressive, formal and includes more relevant aspects (permissions and responsibilities) than AUML proposal.

Moreover, AUML does not cover all the abstractions proposed by Gaia. Specifically, AUML (Parunak and Odell 2002) offers a rather poor notation in covering the organizational structures and does not consider the organizational rules. Thus, it presents some barriers to adapt to complex and open systems with self-interested behavior.

4.1 The Agents marketplace modeled in Gaia + AUML

In this section the illustrating example of Agents Marketplace is specified using Gaia and replacing the preliminary interaction model with the AUML notation. For space reasons we present just a few examples for each model of the analysis phase in which it is possible to appreciate the main benefits of the integration.

The Organization

In the example of the agents marketplace, the need to request goods or services usually implies the request for a set of offers by sellers, and receipt of the offers, and the evaluation by the buyer, after which the service provision is assigned to the winner. Such a solution can be easily delegated to a MAS with the same organizational structure. However, it could also be possible to adopt a different structure. For instance, to improve efficiency one can adopt a descending price auction for requesting services at the best price. The choice for an auction-based negotiation requires re-thinking the organizational structure and, for instance, introduces the need for agents to interact with the mediation of an auctioneer in charge of enacting negotiation rules. Moreover, it is possible to conceive several interacting organizations to co-exist in a marketplace. For example, one could think of two separated organizations: one for dealing with the contracting phase and another for dealing with the subsequent payment and delivery phases. In the next, we pay attention just to the organization for dealing with the contracting phase.

The Preliminary Role Model

In the agents marketplace example, it is possible to identify five possible roles corresponding to three classes of agents: Client and Bidder (for the agents class Buyers); Provider and Supplier (for the agents class Sellers); and Auctioneer (for the agents class Auctioneers, see Figure 2).

Role Schema: `Auctioneer`
Description: Mediator between supplier and bidder in the "Auction " model (the role of the agents class Auctioneers).
Protocol and Activities: `ServiceProposed, Offers, ReceivePriceOffers, PriceEvaluation, AcceptPrice, AskForNewBid, Inform`
Permissions: Reads *offer_definition* //the offer made by the seller Changes *price* //the highest proposed price
Responsibilities: Liveness: `Auctioneer= (ServiceProposed.Offers.ReceivePriceOffers.PriceEvaluation.AcceptPrice!AskForNewBid)*` Safety ▪ *number_of_price_proposal >= 1*

Figure 2: Schema for role Auctioneer

Any role schema is intended to be a semiformal description of an agent's behavior and specifies permissions and responsibilities corresponding to that role. Responsibilities may be specified in terms of liveness (desirable) and safety (avoided undesirable) properties expressed using regular expressions. Those expressions include a set of activities (actions that the agent may perform without any interaction with other agents) and protocols (activities that do require interaction among agents). For example, in Figure 2 an `Auctioneer` needs to access the offer presented by a seller and to propose the seller the highest price offered by bidders, as stated in its permissions. The liveness expression, that may occur 0 or more times, specifies that whenever the Auctioneer receives a proposition of a service (by means of the `ServiceProposed` protocol), then `Offers` this proposition to `Bidders`, `ReceivePriceOffers` from `Bidders`, and then may `AcceptPrice` or `AskForNewBid`. The safety expression states that the `Auctioneer` needs at least one price proposal.

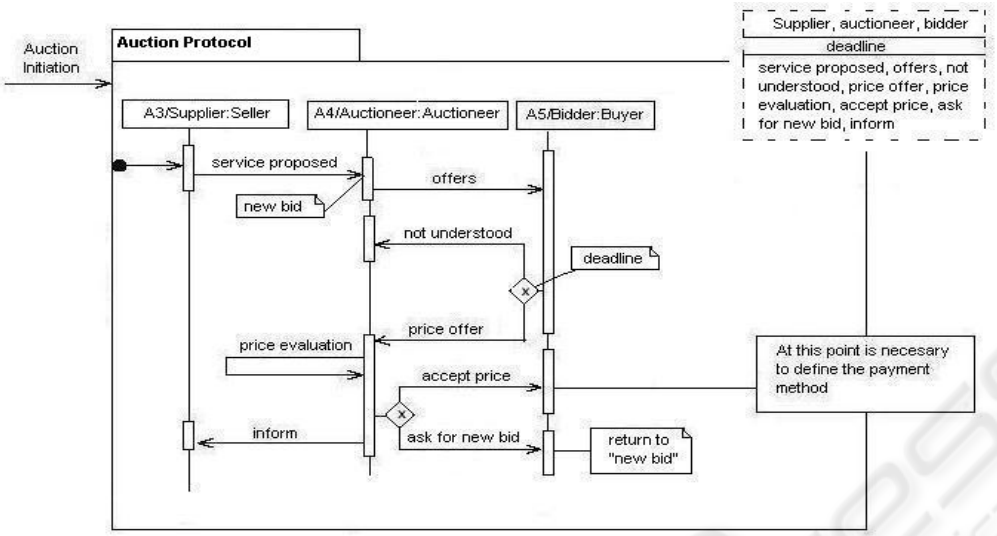

Figure 3: The Auction model protocol

The preliminary interaction model

This model consists of a set of protocol definitions, one for each protocol of each role in the system. More attention is paid to the nature and purpose of the interaction than to the sequence of execution steps. In fact, the protocol definition describes the high-level purpose of the protocol, ignoring implementation details such as the sequence of messages exchanged. In particular, the protocol definition is a simple table detailing the role initiating the protocol, the role in charge of responding to it, the input and output information processed in the protocol, as well as a brief textual description of the type of information processing taking place during the execution of this protocol

The proposed notation of Gaia for protocols is quite informal. Thus, instead of the Gaia notation, we introduce the use of AIP notation proposed by AUML. This implies that a protocol must be described as a sequence of actions and message interactions and may contain a set of atomic protocols as defined in Gaia. In the example we have considered two protocols for the contract phase: one for the "Wanted Request" model and one for the "Auction" model. For space reasons we present just the packagefor "Auction" protocol (see Figures 3), avoiding the activity diagram and statechart.

It states that when the Supplier proposes a service, the Auctioneer offers it to the Bidder looking for the best price. Meanwhile, a Bidder may present a price offer or may inform the Auctioneer that he has not understood the proposal. Once the Auctioneer has evaluated the price it may accept it, informing the Supplier, or reject it asking for new bid.

The Organizational Rules

Gaia considers the Organizational Rules in a perspective that is coherent with responsibilities characteristics of roles but referred to the organization as a whole. Accordingly, it is possible to distinguish between safety and liveness organizational rules. As an example of a safety rule, the Auctioneer cannot participate as a Bidder in an auction it is moderating:

$$\neg \texttt{Auctioneer}\,(\mathit{offers}\,(x)) \mid \texttt{Bidder}\,(\mathit{offers}\,(x))$$

Also, in the case of a negotiation based on the English auction mechanisms, bidder agents must not be allowed to interact directly with the seller, so as to avoid collusions aimed at artificially heightening the selling price of a good.

4.2 Discussion

A possible weakness of Gaia is the notations it proposes and, among them as pointed out in (Sturm and Shehory, 2003), specifically that related to the protocol model. In fact, the proposed notation considers all the relevant aspects of a protocol but may be too extensive to specify (one model for every interaction), and it is quite informal and not based on a standard accepted by industry.

As previous section highlights, the integration of AUML within Gaia leads to a richer notation for the specification of protocols and inter-agent interactions since the AUML notation introduces different advantages. First, it specifies a distinguished set of agent instances satisfying the agent role and class it belongs to; while Gaia just

specifies the role. Second, it is more compact specifying in a single diagram a sequence of actions and messages interactions which may contain a set of atomic protocols as defined in Gaia. Third, AUML is more formal and let to specify the time ordering of messages between agents. Finally, AUML notation introduces the opportunity for agents to select a path in the interaction according to their goals. The latest two aspects are described in Gaia using natural language and so introducing possible ambiguities and misunderstandings.

5 RELATED WORK

Other attempts to extend Gaia for better modeling open MAS include Roadmap (Juan et al., 2002) and Skeleton (Juan et al., 2003) methodologies.

The Roadmap methodology aims to support the engineering of large-scale open systems promoting the view of software systems as computational organizations. Roadmap extends the Gaia methodology by introducing use-cases for requirement gathering, explicit models of agent environment and knowledge, and an interaction model based on AUML interaction diagrams (Juan et al., 2002). However, the interaction model proposed is just a statement without further details or examples and it seems to be an interesting possible idea more than a real conceptualized model. For this reason, it is quite unclear for designers how to accomplish this integration.

The Skeleton methodology proposes an integration of the common elements identified from Prometheus (Padgham and Winikoff, 2002) and Roadmap. It inherits from Roadmap the interaction model with its advantages and the same drawbacks mentioned above.

6 CONCLUSIONS AND FUTURE WORKS

This work proposes the integration of AUML within the Gaia methodology to improve modeling of open MAS. Specifically, we replaced the protocol model of Gaia with the Agents Interaction Protocol (AIP) of AUML, specified by means of protocol diagrams. This extensions to UML enrich Gaia in four main aspects: (i) a richer notation for specifying agent instances of a particular class satisfying the agent role; (ii) a more compact notation that represents in a single diagram a sequence of actions and message interactions; (iii) a more formal notation that reduces possible ambiguities and allows to specify messages between agents; and (iv) multithreaded lifelines that permit agents to select a path in the interaction according to their goals.

Moreover, AUML builds on the acknowledged success of UML in industrial software engineering and it is reasonable to think that it may reduce the distance between researchers' proposals and industry practices. Nevertheless, the main pitfall in using Gaia integrated with AUML to design MAS in an industrial environment is that there are no CASE tools available which implement this methodology.

REFERENCES

Caire, C., Garrijo, F., Gómez, J., Pavón, J., Leal, F., et al., "Agent oriented analysis using MESSAGE/UML". *Proceedings of Agent-Oriented Software Engineering – AOSE 01*, Montreal Canada, May, 2001

Giunchiglia, F., Mylopoulos, J., and Perini A., "The Tropos Software Development Methodology: Processes, Models and Diagrams", *Proceedings of Agent-Oriented Software Engineering (AOSE-2002)*, Bologna, Italy, July 2002

Juan, T., Pearce, A. and Sterling, L., "ROADMAP: Extending the Gaia Methodology for Complex Open Systems". *Proceeding of Autonomous Agents and Multi-Agent Systems - AAMAS '02* (pp. 3-10), Bologna, Italy, July 15-19, 2002

Juan, T., Sterling, L. and Winikoff, M., "Assembling Agent Oriented Software Engineering Methodologies from Features". *Autonomous Agents and Multi-Agent Systems-AAMAS'03*, Melbourne, Australia, July, 2003

Odell, J., Parunak, H. v. D., and Bauer, B., "Extending UML for Agents". *Proceedings of Workshop on Agent Oriented Information Systems – AOIS 2000*, Austin, USA, 2000

Padgham, L. and Winikoff, M., "Prometheus: A Methodology for Developing Intelligent Agents". (poster) *Proceedings of Autonomous Agents and Multi-Agent Systems - AAMAS '02* (pp. 3-10), Bologna, Italy, July 15-19, 2002

Parunak, H. v. D., and Odell, J., "Representing Social Structures in UML". In: Wooldridge M., et al. (eds.): *Agent Oriented Software Engineering – AOSE II* (pp. 1-16), Springer-Verlag, Berlin, Germany, 2002

Sturm, A. and Shehory, O., "A Framework for Evaluating Agent-Oriented Methodologies". *Agent Oriented Information Systems – AOIS 2003 at AAMAS '03*, Melbourne, Australia, July 14, 2003

Zambonelli, F., Jennings, N., and Woolddridge, M., "Developing Multiagent Systems: the Gaia Methodology". *ACM Transactions on Software Engineering and Methodology*, 12(3) (pp. 417-470), July 2003.