

Chapter 1

THE GAIA METHODOLOGY: BASIC CONCEPTS AND EXTENSIONS

Luca Cernuzzi¹, Thomas Juan², Leon Sterling², and Franco Zambonelli³

¹ *Departamento de Ingeniera Electronica e Informatica
Universidad Catlica "Nuestra Seora de la Asuncin",
Campus Universitario, C.C. 1683, Asuncin, Paraguay*
lcernuzz@uca.edu.py

² *Department of Computer Science and Software Engineering,
The University of Melbourne, 3010, Victoria, Australia*
{tlj, leon}@cs.mu.oz.au

³ *Dipartimento di Scienze e Metodi dell'Ingegneria,
Universit di Modena e Reggio Emilia,
Via Allegri 13 - 42100 Reggio Emilia, Italy.*
franco.zambonelli@unimo.it

Abstract In the last few years, there have been several attempts to identify appropriate abstractions for developing Multi-Agents Systems (MAS), and to propose software engineering methodologies accordingly. Among those, the Gaia methodology [Wooldridge et al. 1999] was a pioneering attempt to explicitly focus on using organizational abstractions to drive the analysis and design of closed cooperative multi-agent systems. Actually, a growing trend in agent applications focuses on open MAS based on a great number of agents. The agent interaction is largely dynamic and may produce flexible emergent behavior and distributed intelligence. This chapter presents the original proposal of the Gaia methodology, and three proposals to extend Gaia to better design and build systems in complex, open environments.

Keywords: Agent Oriented Methodologies, Gaia, Open MAS, ROADMAP, AUML

1. Introduction

The challenge of developing Multi-Agents Systems (MAS) has led, in the past few years, to identify appropriate abstractions. These abstractions would then be used as the basis of a systematic methodology for MAS development. Several of these methodologies view MAS as needing to reflect human organizations, and be themselves modeled as organizations or societies of agents. Individual agents will fulfill organizational roles and interact according to specific protocols prescribed by the roles they play.

The Gaia methodology [Wooldridge et al. 1999] was a pioneering attempt to explicitly focus on using organizational abstractions to drive the analysis and design of closed cooperative multi-agent systems.

A growing trend in agent applications is to focus on open MAS based on a great number of agents. Agent interaction in open MAS is largely dynamic and may produce flexible emergent behavior and distributed intelligence.

A typical example of an open MAS composed of self-interested agents is an Agents Marketplace. Electronic marketplaces may assume the form of Web sites, where communities of users interested in specific types of goods and services can meet and arrange their commercial transactions interactively, without the constraint of physical co-location.

The need for considerable human interaction limits the capability and widespread acceptance of such Web sites, as time is often required for seeking and contracting for goods. Intelligent agents are emerging as a powerful solution to this problem. Agents may buy and sell goods and services in the marketplace on behalf of persons, enterprises, or computational organizations. Agents can effectively accomplish the tasks of seeking 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. Thus, we can envision an Internet populated with a variety of special-purpose marketplaces, typically associated with one or more Web sites.

In agent-populated marketplaces, agents interested in specific classes of goods will meet to access an environment made up of "wanted requests" and "sales offers" (possibly based on an auction model). The agents will form an organization of agents playing the roles of "client" and "provider" in a wanted request model as well "bidder" and "supplier" in an auction-based model. Agents fulfilling roles will interact according to specific negotiation patterns. The intrinsic openness of the marketplace, where different agents with different principles, may enter to negotiate, introduces the issues of controlling proper ways to conduct

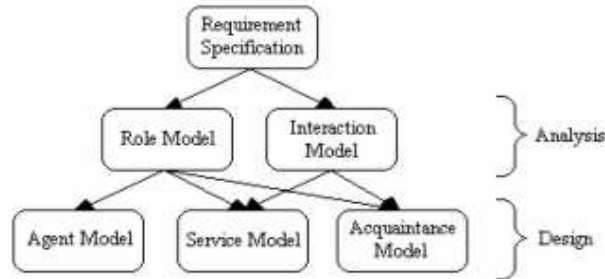


Figure 1.1. Models in the Gaia methodology

negotiations. For example, we wish to avoid agents with self-interested behavior cheating with each other.

Moreover, it is possible to conceive several interacting organizations to co-exist in a marketplace. For example, one could have two independent organizations for dealing with the contracting phase and the subsequent payment and delivery phases.

Consequently, improvements for Gaia have been proposed to better capture the flexibility characterizing systems in complex, open environments. This chapter presents the original proposal of the Gaia methodology, and three proposals to extend Gaia to better model open MAS. The marketplace scenario sketched above is used as a running example for Gaia and the proposed extensions.

2. GAIA IN A NUTSHELL

The basic Gaia methodology has been laid out in [Wooldridge et al. 1999]. It models both the macro (social) aspect and the micro (agent internals) aspect of a multi-agent system. Gaia views a system as a society or organization of agents.

The Gaia methodology is applied after the requirements are gathered and specified, and covers the analysis and design phases. It consists of constructing a series of models as shown in Figure 1.1.

In the analysis phase, the role model and the interaction model are constructed. These two models depict the system as a set of interacting abstract roles. The two models are then used as input to the design stage. The agent model, the services model and the acquaintance model are constructed during initial stages of design. These three models are then taken into the detailed design stage. Gaia does not cover detailed design and relies on conventional methodologies for that purpose.

Table 1.1. Partial syntax of liveness properties

Operator	Interpretation
$x.y$	x followed by y
$x \mid y$	x or y occurs
x^ω	x occurs indefinitely often
$[x]$	x is optional
$x \parallel y$	x and y interleaved

Analysis with Gaia

In the analysis stage, roles in the system are identified, and their interaction is modeled. These roles are abstract constructs used to conceptualize and understand the system thus they have no realization in the implemented system. In Gaia, all roles are atomic constructs and cannot be defined in terms of other roles. Any role schema is intended to be a semiformal description of an agent's behavior and is defined in terms of four attributes: responsibilities, permissions, activities and protocols.

Permissions limit the resources available to the role, usually expressed as some information that the role can read, write or create. The permissions specify both what the role can and cannot use. Activities are tasks or actions a role can take without interacting with other roles, and protocols are tasks or actions a role can take that involve interaction with other roles.

Responsibilities of a role define the role's functionalities. There are two types of responsibilities, liveness properties, and safety properties. The former describes the "lifecycle" or generalized behavior pattern of the role. Liveness properties are represented by a w-regular expression over the sets of activities and protocols the role processes. The syntax of liveness properties used in this chapter is shown in Table 1.

The safety properties are properties that the agent acting on the role must always preserve. Safety properties are expressed as predicates over the variables in the permissions of the role, specifying the legal values these variables can take. The role model is simply a collection of role schema, detailing the attributes for each role in the system.

For our running example of the agent marketplace, 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). Below, three of those schemas are presented.

Role Schema: BIDDER		
Description: Evaluating sellers' offers and making a price offer for a service and/or goods in the Auction model.		
Protocols and Activities: ReceiveCfO, Not-Understood, PriceOffer		
Permissions:		
reads	<i>offers</i>	// <i>offers from the seller</i>
changes	<i>offer_evaluation</i>	// <i>evaluation of the sellers' offers</i>
	<i>price</i>	// <i>price proposed</i>
Responsibilities		
Liveness:		
BIDDER = (ServiceProposal)		
Safety:		

Figure 1.2. Schema for role BIDDER

Role Schema: SUPPLIER		
Description: Proposing a service and/or goods in the Auction model.		
Protocols and Activities: ServiceProposal, ReceiveApproval		
Permissions:		
changes	<i>offer_definition</i>	// <i>service and/or good proposed</i>
Responsibilities		
Liveness:		
SUPPLIER = (ServiceProposal)		
Safety:		

Figure 1.3. Schema for role SUPPLIER

Taking a closer look, in Figure 1.4 an Auctioneer needs to access the offer presented by a seller and to propose to 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 an agent implementing the Auctioneer role receives a proposition of a service (by

Role Schema: AUCTIONEER	
Description: Mediating between suppliers and bidders in the Auction model.	
Protocols and Activities: ServiceProposed, Offers, ReceivePriceOffers, <u>PriceEvaluation</u> , AcceptPrice, AskForNewBid, Inform	
Permissions:	
reads	<i>offer_{definition}</i> // the offer made by the seller
changes	<i>price</i> // the highest proposed price
Responsibilities	
Liveness:	
AUCTIONEER = (ServiceProposed.Offers.ReceivePriceOffers. <u>PriceEvaluation</u> .(AcceptPrice AskForNewBid))*	
Safety:	
■ <i>number_of_price_proposals</i> >= 1	

Figure 1.4. Schema for role AUCTIONEER

means of the ServiceProposed protocol), it then offers (using the Offers protocol) this proposition to the agents fulfilling the bidders role, it receives price offers (using the ReceivePriceOffers protocol) from the set of bidding agents, and then may accept the price or ask for a new bid, using the AcceptPrice or AskForNewBid protocols respectively. The safety expression states that an agent playing the Auctioneer role needs at least one price proposal.

An interaction model is developed based on the initial role model. It contains a protocol definition for each protocol of each role in the system. More attention is paid to the nature and purpose of the interaction than to the precise sequence of execution steps and message exchanges. The protocol definition describes the high level purpose of the protocol, ignoring implementation details such as the sequence of messages exchanged. The protocol definition outlines the initiating role, the responding role, the input and output information, as well as a brief description of the processing the initiator carries out during the execution of this protocol.

For example, Figure 5-5 represents the AcceptPrice protocol.

Analysis within Gaia involves refining the role model and the interaction model iteratively. They serve as the initial input to the design stage.

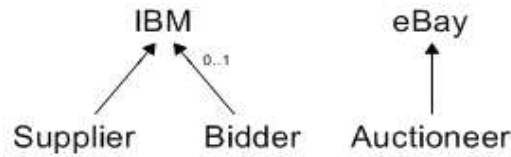


Figure 1.5. A sample agent model

Design with Gaia

In the design phase, the abstract constructs from the analysis stage, such as roles, are mapped to concrete constructs, such as agent types, that will be instantiated at runtime. Gaia requires three models to be produced, see Figure 1.1. The agent model outlines the agent types in the system. The services model outlines the services required by the roles assigned to the agent types. The acquaintance model depicts communication links between agent types.

Assigning the roles to agent types creates the agent model. Each agent type may be assigned one or more roles. For each agent type, the designer annotates the cardinality of agent instances of that type at runtime. Figure 1.5 shows a sample agent model where IBM takes both the Supplier role and the Bidder role.

In Gaia, a service is simply a coherent block of functionality, neutral with respect to implementation details. The service model lists services that agent types provide. The services are derived from the activities and protocols of the roles. For each service, four attributes must be specified, namely the inputs, outputs, pre-conditions and post-conditions. They are easily derived from attributes such as protocol input, from the role model and the interaction model.

The acquaintance model is a directed graph between agent types. An arc from A to B signals the existence of a communication link allowing A to send messages to B. The purpose is to allow the designer to visualize the degree of coupling between agent types. In this model, further details such as message types are ignored.

Limitations

Gaia was designed to handle small-scale, closed agent-based systems. Consequently, it has weaknesses that render it inappropriate for engineering complex open systems like the Agents Marketplace. Gaia has the following limitations:

- 1 Gaia assumes complete specification of requirements and does not address the requirements gathering phase. It does not guide developers to take advantage of richer requirements enabled by agent technologies. The methodology does not facilitate regular changes of requirements typical of open systems.
- 2 Environmental information is implicitly encoded in the permissions and protocols of individual roles. Gaia does not present a holistic model of the execution environment to the developers. This omission renders Gaia inappropriate for engineering applications with dynamic and heterogeneous environments.
- 3 Domain knowledge in the system is implicitly encoded in the attributes of the individual roles. Gaia does not present a holistic model of the structure of the domain knowledge and the interaction and dependencies of knowledge components in the system. This omission prohibits knowledge in the system to be shared, re-used, extended and maintained in a modular fashion.
- 4 Gaia cannot explicitly model and represent important social aspects of a multi-agent system. Among them, Gaia cannot explicitly model the organization structure of the agents in the system, or alternatively, the architecture of the system with merely non-recursive roles. It also lacks the ability to explicitly model the social goals, social tasks or organizational rules within an organization of agents.
- 5 Gaia offers no mechanisms to model dynamic reasoning to allow extension and modification of social aspects at runtime.
- 6 Gaia employs special notation that is different from established software development notations such as UML.

3. ROADMAP

The first extension to the Gaia methodology that we discuss is ROADMAP developed at the University of Melbourne and reported in (Juan et al., 2002).

ROADMAP started as an early attempt to extend Gaia with a dynamic role hierarchy and additional models to describe the agent environment and the agent knowledge, to address the same open issues described in the last section. It inherits the organizational view of computation and the basic definitions of roles, protocols, agents and services from Gaia. However, over time the semantics of these concepts used in ROADMAP become quite different from Gaia. In this light, ROADMAP

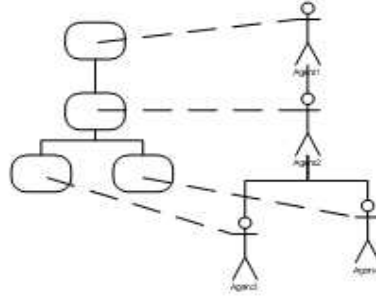


Figure 1.6. System viewed as a computational organization made up of the role hierarchy and the agent hierarchy

may be better considered not as extensions to Gaia, but as a separate methodology.

In ROADMAP, a system is viewed as an organization of agents, consisting of a role hierarchy and an agent hierarchy (Figure 1.6).

The role hierarchy is the specification of the system, representing the correct behaviour of agents. The agent hierarchy is the implementation of the system, providing the actual functionalities. The role hierarchy constraints the agent hierarchy in the same way as organizational structure, responsibilities and business procedures constrains individuals in a human organization.

Roles and protocols are first class entities that have concrete runtime realization in ROADMAP. In a ROADMAP organization, agents interact by message passing, while roles and protocols act as message filters (Figure 1.7).

Figure 1.7 shows an example of official interaction in an organization between Agent A and Agent B. The message from Agent A is first sent to and validated by its role. If all constraints are satisfied, the message propagates to Agent B's role. After the message is validated, Agent B receives the message and can now respond to it. As part of the organizational arrangement, the message is also forwarded to Agent C's role and to Agent C after validation for monitoring purpose. If the message fails to satisfy constraints from any roles concerned, the message will be rejected and actions will be taken to handle the error. This mechanism ensures the interaction respects perspectives of all roles involved. Direct message passing between agents are considered private and does not have the same official status in the organization. In some organizations, private interaction is not desirable and maybe forbidden.

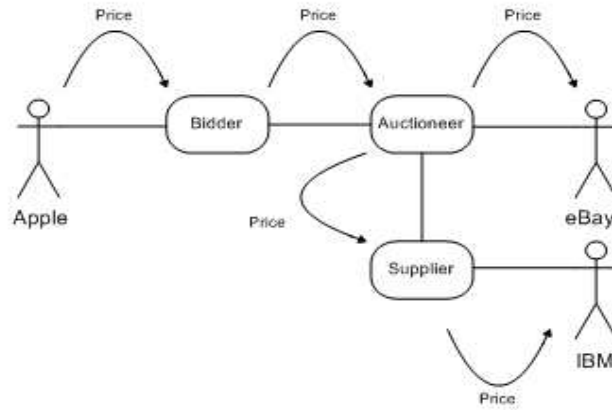


Figure 1.7. Message passing between agents via their roles

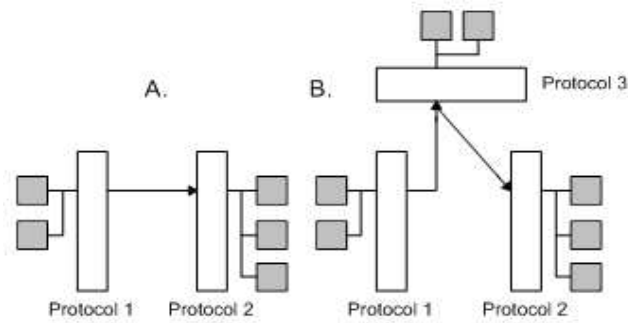


Figure 1.8. Re-routing interaction with protocols

Protocols are reusable message patterns. As concrete runtime entities, they can be reasoned and manipulated, effectively re-routing the interaction without affecting the agent services (in grey) behind the protocols (Figure 1.8).

Figure 1.9 shows an example ROADMAP role. The main improvements from Gaia roles are:

- 1 The new Sub-Roles attributes that use the aggregation semantics for building a role hierarchy recursively. The "involves" keyword relates sub-role attributes to parent role attributes. For example, a parent role safety condition is maintained if and only if all involved sub-role safety conditions are maintained.

- 2 The new knowledge attributes that associate knowledge components with roles.
- 3 Use of keywords "before", "during" and "after" to limit the applicability of attributes to a liveness state or a protocol. This allows pre / post conditions and invariants of protocols to be defined and the implementing services constrained at runtime. fine-grain
- 4 Evaluation functions such as Profit_Margin are specified. The functions serve as official measure of agent performance. The Goals attribute nominate the correct evaluation functions for agents to optimize at a given state. These functions can be implemented in any roles, agents or resources.
- 5 The permission attribute can now include read or modify access to other roles or protocols, allowing the organization to be changed at runtime given the proper authorization.

The key ROADMAP concepts are outlined in the ROADMAP meta-model (see Figure 1.10)

Figure 1.11 shows the structure of the ROADMAP models. The models are grouped into three categories. The environment model and the knowledge model contain reusable high-level domain information. The use-case model, interaction model, role model, agent model and acquaintance model are application specific. The protocol model and service models describe potentially reusable low level soft-ware components.

The ROADMAP methodology borrows its notation from established standards such as UML for use-case modeling and interaction diagrams.

4. THE NEW GAIA

The next extension is called New-Gaia. An underlying philosophy of New-Gaia is to view an organization as more than a simply a collection of roles. Additional explicit abstractions are needed to be devised to consider the organizational perspective (see [Zambonelli et al 2004] for more details). These additional abstractions emerge from the belief that a human organization metaphor is the most appropriate for agent-oriented software engineering.

The most basic abstractions that characterize a computational organization are:

- the roles that different agents play;
- the interaction between roles; and
- the environment in which the MAS is interacting.

Role Schema: Supplier	
Description: The role of proposing a service and/or good in the “Auction” model (the role of the agents class Sellers).	
Sub-Roles:	Marketing, Sales, ServiceStaff //sub-roles
Knowledge:	Sales History //history of previous sales Market Trend //knowledge on the current market demands
Responsibilities:	
Liveness:	Supplier = (SellService Preparation)* SellService = ServiceProposal . ReceiveApproval
Safety:	1. Profit_Margin(offer_definition) >= 10% during ServiceProposal 2. Customer_Awareness (offer_definition) >= 30 % during Preparation involves Marketing . Safety1
Goals:	
ProfitGoal = Max. Profit_Margin during ServiceProposal	
CusGoal = Max. Customer_Awareness during Preparation	
	involves Marketing . CusGoal
According to Prioritize ()	
Protocol and Activities:	
ServiceProposal involves Sales . MakeProposal,	
ReceiveApproval involves Sales . ReceiveApproval,	
Preparation involves Marketing . Research and ServiceStaff . CreateService	
Permissions:	
Changes offer_definition //service and/or good proposed	
Changes Market Trend during Preparation //modify a knowledge component	
Changes This_Role . Protocols during Preparation //allow reflection on all protocols of	
	//this role; self-modification

Figure 1.9. A sample ROADMAP role definition

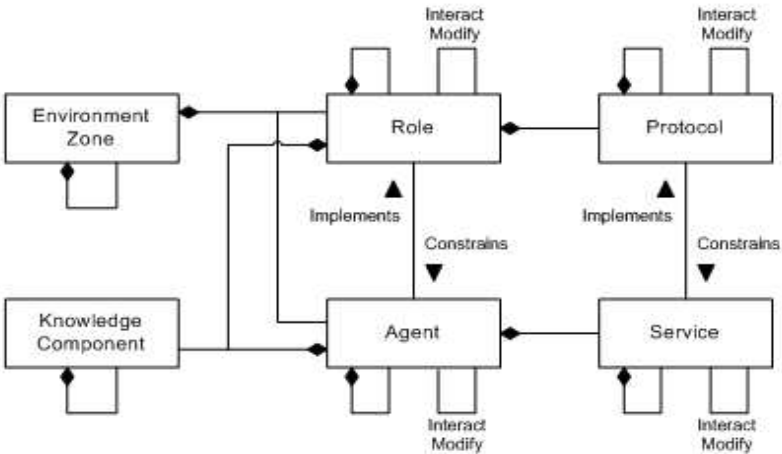


Figure 1.10. The ROADMAP meta-model

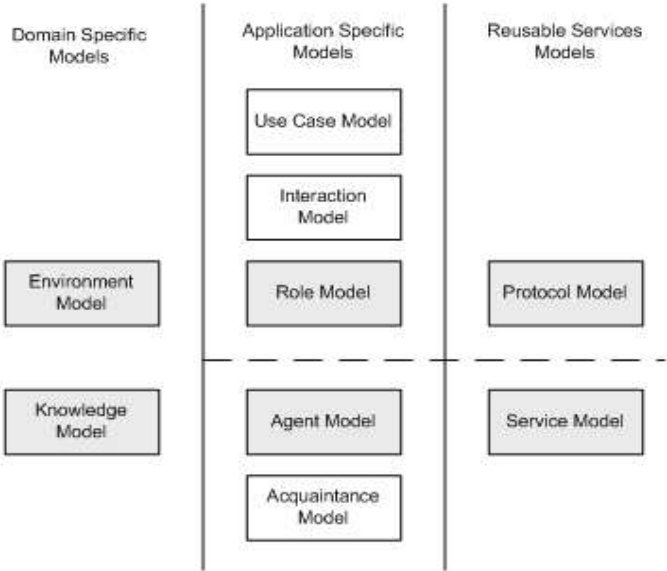


Figure 1.11. Models in the ROADMAP methodology

In the context of New-Gaia the environment abstraction aims to specify all the entities and resources the MAS may interact with to reach the organizational goal. However, the specification of agents with their roles and the interactions among them and with the environment is insufficient to capture the complex and emergent behavior derived from many self-interested agents.

Two other abstractions that are useful to explicitly take into consideration are:

- organizational rules; and
- organizational structure.

Organizational rules are aimed to specify some constraints that the organization will have to observe. Rules may be global (concerned with all the roles and protocols), or just concerned with the relations between roles, between protocols, or between roles and protocols. Rules allow the system designer to explicitly define when and under which conditions a new agent may participate in the organization, which is its position, as well as which behaviors are acceptable as self-interested expressions and which ones have to be prevented by the organization.

Organizational rules may help designer in the identification of organization structures that more naturally adapt to these rules. In this sense, the role model should derive from the organizational structure explicitly defined in the design phase of the MAS. A great number of different organizational structures may be available for designers to better deal with functional and efficiency requirements. Nevertheless, it is highly probable that a reduced subset of these structures are normally adopted. This opens up the opportunity to define catalogs of organizational patterns that may be re-used.

Considering all the above, an overview of the New-Gaia methodology with its models and their relationships is presented in Figure 1.12.

The analysis phase includes the identification of:

- The goals of the organizations that constitute the overall system and their expected global behavior. At this step it is important to identify useful decomposition of the global organization into sub-organizations.
- The environmental model that represents the environment in which the MAS will be situated in term of computational components.
- The preliminary roles model. At this stage, Gaia's notion of roles is abstract from any mapping into agents (this issue will be con-

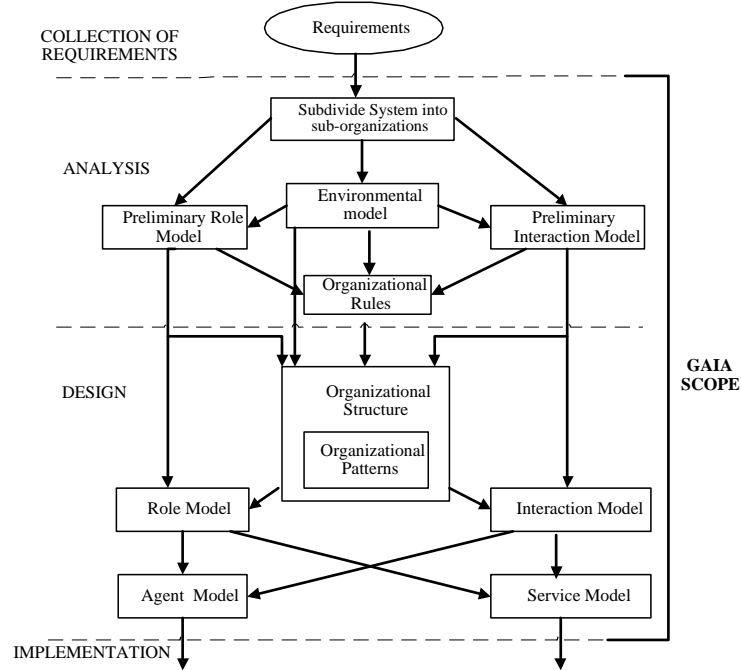


Figure 1.12. Models of the Gaia Methodology and their Relationships

sidered in the design phase) and the analyst has to avoid the imposition of a specific organizational structure.

- The preliminary interaction model, which, while possibly not 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. They are fundamentals to efficiently specify the openness and the general behavior of the developing MAS.

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

The design phase contemplates the following sub-phases:

- Definition of the overall organizational structure respecting the organizational rules. At this stage it is important to exploit well-known organizational patterns that also may influence the design of the final interactions model.

- Revision and completion of the preliminary role and interaction models, considering the adopted organizational structure.
- Definition of the agent model specifying agent types (a set of agent roles) and agent instances.
- Definition of the services model which specifies the main services (blocks of activities with their conditions) related with the agent roles.

The general framework of New-Gaia explicitly covers consistently the abstractions previously defined and undertaken according to the limitations 2 and 4 previously presented in section 2.3. Moreover, New-Gaia is more oriented to designing and building systems in complex, open environments that constitute the new trends in MAS applications. For example, considering the agents marketplace, New-Gaia lets designers model the organization structure as well as the organizational rules.

The Organization

In the example of the agents marketplace, the need to request goods and/or a service usually implies, in real-world organizations, 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. Of course, such a solution can be easily delegated to a MAS that provides the same things with the same organizational structure. However, to improve efficiency, one could 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, as we mentioned above, it is possible to conceive several interacting organizations co-existing in a marketplace: for example, one organization 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 Environmental Model

In a simplified view it is possible to conceive as resources of interest the services (required or offered) and the offers. This leads to the following (minimal) environmental model:

The Preliminary Role Model is the same as proposed in the original Gaia (see section 2.1 and, specifically Figures 1.2, 1.3 and 1.4).

The Organizational Rules

Gaia considers the Organizational Rules in a perspective that is coherent with responsibilities as characteristics of roles but referring 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:

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.

However, an open issue is the notations proposed by Gaia (see item 6 in section 2.3). Notation is addressed by the extension in the following section.

5. APPLYING AUML TO GAIA

Probably due to its simplicity, Gaia notation is poor and unlikely to be widely accepted for industry solutions. The gap from industrial practice is quite evident with respect to the specification of agent interactions, as pointed out in [Sturm and Shehory 2003]. In effect, the Gaia protocol model notation considers all the relevant aspects of a protocol but may be too extensive to specify (one model for every interaction). Further the notation is quite informal and not based on a standard accepted by industry.

Our proposed extension is to re-use Agent UML (AUML) [Bauer et al. 2001, Odell et al. 2000], a set of extensions to UML notation that have been proposed for modeling agent-based systems. AUML builds on the acknowledged success of UML in supporting industrial-strength software engineering. The core part of AUML is the Agents Interaction Protocol (AIP). Protocols in AIP are specified by means of protocol diagrams (extended sequence diagrams) that allow designers to specify extended message semantics, parameterized nested protocols, and protocol templates.

The key ideas of AUML that may be integrated into Gaia to enrich its expressiveness in specify agent interactions are:

- 1 The protocol can be regarded as a whole entity 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.

- 2 Each protocol implies inter-agent interactions that are described using sequence diagram, 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 in order to better capture richer role specification, packages with agent interfaces, deployment diagrams indicating mobility, emergence, etc. However, those notations are less rich than those proposed for AIP and some of them are poor compared with Gaia notations. For example, role specification in Gaia is more expressive, formal and includes more relevant aspects (permissions and responsibilities) than proposed in AUML.

Moreover, AUML is not a thorough methodology and does not cover all the abstraction proposed by New-Gaia. Specifically, AUML [Parunak and Odell 2002] offers a quite poor notation in covering the organizational structures and does not consider the organizational rules. It presents some barriers to adapt to complex and open systems with self-interested behavior.

The Preliminary Interaction Model

The proposed notation of Gaia is quite informal as may appear in the example presented in section 2.1 (see Figure 5-5). 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 agents marketplace 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 package and the activity diagrams for "Auction" protocol, avoiding the statechart.

In Figure 1.13 and 1.14 it is possible to observe the protocol specifying an Auction model with its respective activity diagram. 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 inform the Auctioneer that he has not understood the proposal or may present a

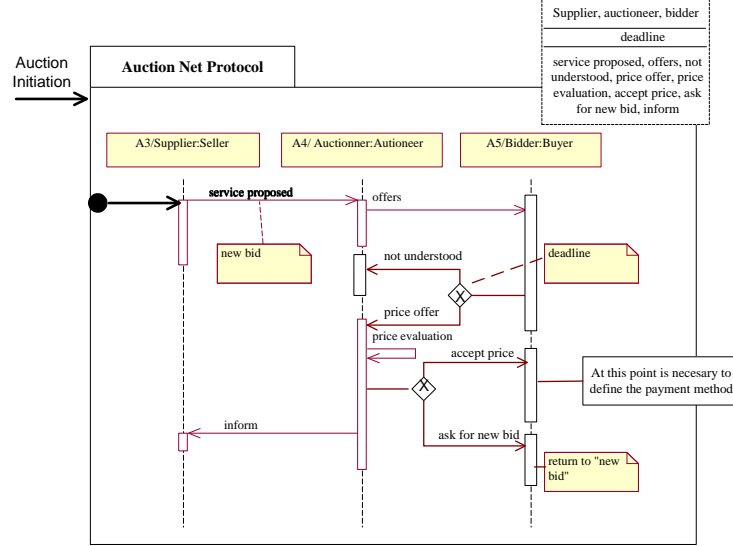


Figure 1.13. The auction model protocol

price offer. Once the Auctioneer has evaluated the price it may accept it informing the Supplier, or reject it asking for new bid.

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 allows the specification of time ordering of messages between agents. Finally, AUML notation introduces an opportunity for agents to select a path in the interaction according to their goals. The latter two aspects are described in Gaia using natural language, introducing possible ambiguities and misunderstandings.

6. OPEN ISSUES

The previous section described an extension to Gaia to better represent aspects of the interaction protocol between agents. However, other notations may be improved. For instance, with regards to the environment specifications, Gaia is centered on agent interactions with the environment. The environment is considered as a provider of computational resources, and Gaia focuses on the interaction with the data

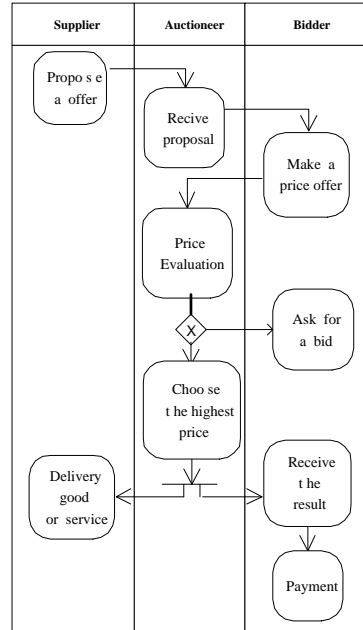


Figure 1.14. The auction model protocol activity diagram

provider, because active components and services may be "agentified". For this purpose Gaia uses a notation inspired by FUSION for operation schemata [Coleman et al. 1994] complemented by means of a graphical representation of the spatial, physical, or logical relationships between environment resources. Other authors give increasing importance to environment modeling, though mostly centered on static aspects of the environment (without explicitly considering the agent interactions with the environment) which may be captured using traditional object diagrams (see for example [Parunak and Odell 2002]). Static environment models evidently adhere more to industrial standards, but do not consider the dynamic aspects of the agent interactions with the environment.

Further, the organizational structure is modeled using a simple, non standard, notation for expressing relationships among roles (or agents) complemented by graphical representations and textual comments aimed at enriching the semantics. Perhaps, also the specification of organizational rules, currently using FUSION-like notation or temporary logic may be improved.

Another important limit, pointed out by in section 3 describing ROADMAP, refers to requirements modeling. ROADMAP proposes the use of case models to capture functional requirements. Other agent oriented method-

ologies put special emphasis on requirements modeling. Among them, Tropos [Giunchiglia et al 2002] seems to be the most formal, complete, and consistent. In the future, it may be useful to explore the opportunity of merging Tropos with Gaia.

7. CONCLUSION

This chapter has presented the Gaia methodology and three of its extensions. The first, ROADMAP, aims to extend Gaia with a dynamic role hierarchy and additional models to describe the agent environment and the agent knowledge. The second, New-Gaia, is a new version of Gaia considering additional abstractions and models that are necessary for designing and building systems in complex, open environments exploiting the organizational perspective. In this sense, additionally to the environment model, organizational rules and structure are explicitly modeled. The third proposes the integration of AUML within Gaia. It leads to a richer notation for the specification of protocols and inter-agent interactions.

Further work is needed to improve other Gaia notations and to include the requirements modeling phase.

Acknowledgments

The second author would like to acknowledge the support of the Smart Internet Technology CRC.

References

- Bauer, B., Miller, J., and Odell, J., "Agent UML: A formalism for specifying multiagent software systems". In Ciancarini, P., Wooldridge, M. (eds.) *Agent-Oriented Software Engineering* (pp.91-103), Springer-Verlag: Berlin, Germany, 2001
- Coleman, D., Arnold, P., Bodoff, S., Dollin, D., Gilchrist, H., Hayes, F., and Jeremas, P., "Object-Oriented Development: the FUSION Method". Prentice-Hall International, Hemel Hempstead (UK), 1994
- 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. and Sterling, L., "The ROADMAP Meta-model for Intelligent Adaptive Multi-Agent Systems in Open Environments". Proceedings of the Fourth International Workshop on Agent Oriented Software Engineering, at 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
- 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
- Wooldridge, M., Jennings, N., and Kinny, D., "A methodology for agent-oriented analysis and design". In Proceedings of Autonomous Agents (Agents 99) (pp. 69-76), Seattle, WA, May, 1999
- Zambonelli, F., Jennings, N., and Wooldridge, M., "Developing Multiagent Systems: The Gaia Methodology". (submitted paper) ACM - Transaction on Software Engineering and Methodology, 2004