Chapter 1

THE GAIA METHODOLOGY: BASIC CONCEPTS AND EXTENSIONS

Luca Cernuzzi[1], Thomas Juan[2], Leon Sterling[2], and Franco Zambonelli[3]

[1] *Departamento de Ingeniera Electrnica e Infromtica*
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@unimore.it

Abstract   The Gaia methodology (Wooldridge et al., 2000) has been the first complete methodology for the analysis and design of multiagent systems. However, the original version of Gaia suffered from the limitations of being suitable for the analysis and design of closed multiagent systems and of adopting non-standard notation techniques. For these reasons, several extensions to the basic Gaia methodology has been recently proposed. In this chapter, we summarize the key characteristics of the original Gaia methodology and present three extensions that have been proposed to Gaia to overcome its limitations and to make it more suitable for the development of open multiagent systems in complex environments.

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

1. Introduction

Gaia has been the first complete methodology being proposed to guide the process of developing a multiagent system from analysis to design.

The first version of Gaia, described in (Wooldridge et al., 2000), emphasizes the necessity to identify proper agent-oriented abstractions around which to center multiagent systems development process. In particular, it outlines the suitability of the organizational metaphor, in which a multiagent systems is conceived as a computational organization of agents, each playing specific roles in the organization, and *cooperating* with each other towards the achievement of a common application (i.e., organizational) goal.

Although the Gaia methodology has been quite influential in the past few years, it suffers from several limitations that may undermine the possibility of effectively adopting it for the majority of real-world multiagent scenarios.

A first limitation refers to the fact that Gaia, in the original proposal, is suited only to drive the analysis and design of *closed multiagent systems*, in which agents must be benevolent to each other and willing to cooperate. Unfortunately, this is not the case of most of today's agent systems, where agents can belong to different stakeholders and can express self-interest in actions.

A second limitation relates to the fact that the *notation* proposed by Gaia to model and represents a multiagent system and its components appears not suitable to tackle the complexities of real-world systems and, even worse, is not inspired by accepted software engineering standards.

As a consequence of the above limitations, *improvements* to the basic Gaia methodology have been proposed in order to both capture the characteristics of open multiagent systems in complex open environments and to improve its notation techniques. This chapter shortly presents the original original version of the Gaia methodology and three proposals for extensions: two proposals extend the basic process of Gaia to make it suitable for open multiagent systems, while a third proposal attempts to integrate in Gaia standard notation techniques.

To exemplify the concepts expressed in this chapter, we exploit a simple running example in the area of *agent-mediated marketplaces*, as a typical example of an open multiagent system in which agents may exhibit self-interested behaviors. In agent-mediated marketplaces, agents interested in buying and selling specific classes of goods will meet together to access an environment made up of "wanted requests" and "sales offers", and in which transactions typically take place in the forms of open public auctions. Agents meeting at a marketplace will form dynamic and open
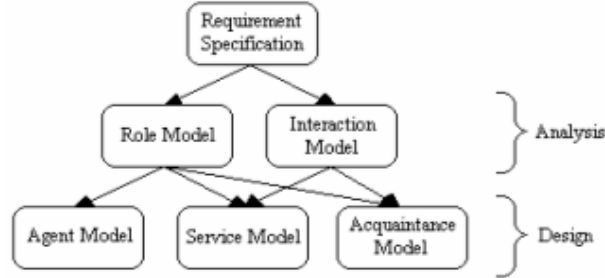
Figure 1.1. Models in the Gaia methodology (Wooldridge et al., 2000)

organizations in which agents themselves can play roles such as "client" and "provider" when publishing requests and offers for goods, as well as roles such as "bidder" and "supplier" when subsequently involved in an auction. The intrinsic openness of the marketplace, where different agents each with its own goals may enter to negotiate, introduces the issues of controlling proper ways to conduct negotiations, i.e., avoiding agents 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 auction phase and the subsequent payment and delivery phases.

2.   Gaia in a Nutshell

The first version of the Gaia methodology is described in (Wooldridge et al., 2000). The scope of the methodology includes the analysis and design phase, and it exclude both collection of specifications (it is applied after the requirements are gathered and specified) and implementation.

In general, the Gaia process consists in orderly constructing a series of models, as shown in Figure 1.1, aimed at describing both the macro (societal) aspects and the micro (intra-agent) aspects of a multi-agent system, generally conceived as an organized society of individuals (i.e., a computational organization of autonomous entities). In the analysis phase, the role model and the interaction model are constructed, to depict the system as a set of interacting abstract roles. These two models are then used as input to the design stage, in which an agent model, a services model, and an acquaintance model are defined to form a complete design specification of the multiagent system to be used for the subsequent implementation phase (not dealt with by Gaia).

4

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

2.1 Analysis with Gaia

In the analysis stage, roles in the system are identified, and their interactions are modeled.

Roles are abstract constructs used to conceptualize and understand the system, with no concrete counterpart in the implemented system. In Gaia, all roles are atomic constructs and cannot be defined in terms of other roles. A role schema is intended to be a semi-formal description of an agent's behavior, and the collection of role schemas for a system define the complete role model. For each role, a role schema is defined in terms of four attributes: permissions, responsibilities, activities and protocols.

Permissions express the environmental resources available to the role, usually in terms of some information that the role can read, write or create. The permissions specify both what the role can and cannot use.

Responsibilities of a role define the role's actual functionality. There are two types of responsibilities, safety and liveness properties. Safety properties are properties that the agent acting on the role must always preserve. These are expressed as predicates over the variables/resources in the permissions of the role, specifying the legal values these variables/resources can take. Liveness properties describe the "lifecycle" or generalized behavior pattern of the role. Liveness properties are represented by a regular expression (see Table 1.1 for the syntax of liveness properties) over the sets of activities and protocols the role executes. There, activities are intended to represent those tasks or actions a role can take without interacting with other roles, while protocols are tasks or actions a role can take that involve interaction with other roles.

For our running example of the agent marketplace, it is possible to identify five possible roles: Client, Bidder, Provider, Supplier, and Auctioneer. In an actual implementation of the system, it is expected that the roles Client and Bidder will be played by some buyer agents, Provided and Supplier by some seller agents, and that the role Actioneer will be played by some auctioneer agents. However, the Gaia analysis

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	*offers*	*// offers from the seller*
changes	$offer_e valuation$	*// evaluation of the sellers' offers*
	price	*// price proposed*

Responsibilities
Liveness:

$$\text{BIDDER} \quad = \quad (\text{ServiceProposal})$$

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	$offer_d efinition$	*// service and/or good proposed*

Responsibilities
Liveness:

$$\text{SUPPLIER} \quad = \quad (\text{ServiceProposal})$$

Safety:

Figure 1.3. Schema for role SUPPLIER

phase abstracts from the presence of agents playing specific roles, an is-
sue that is dealt with in the design phase. Below, three of those schemas
are presented according to the Gaia notation.

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

---

Role Schema: AUCTIONEER

Description:
  Mediating between suppliers and bidders in the Auction model.

Protocols and Activities:
  ServiceProposed, Offers, ReceivePriceOffers, PriceEvaluation, AcceptPrice,
  AskForNewBid, Inform

Permissions:

| reads | $offer_d efinition$ | // the offer made by the seller |
| changes | $price$ | // the highest proposed price |

Responsibilities
Liveness:

$$\text{AUCTIONEER} \; = \; (\text{ServiceProposed.}$$
$$\text{Offers.ReceivePriceOffers.}$$
$$\text{PriceEvaluation.}(\text{AcceptPrice}|\text{AskForNewBid}))^*$$

Safety:
  ■ $number\_of\_price\_proposals >= 1$

Figure 1.4.   Schema for role AUCTIONEER

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 proposal of a service (by means of the ServiceProposed protocol), it then offers (using the Offers protocol) this proposal 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.

In addition to the role model, the Gaia anlysis phase include the definition of an interaction model, including 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 sequence of execution steps and message exchanges. 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
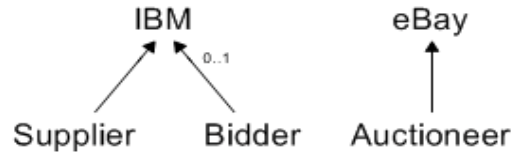
Figure 1.5. A sample agent model

type of information processing taking place during the execution of this protocol.

2.2 Design with Gaia

In the design phase, the abstract constructs of the analysis stage, i.e., the roles and protocols represented in the role and interaction models, are mapped into concrete constructs, i.e., into the agent types that will be instantiated at runtime.

Gaia requires three models to be produced in the design phase (see Figure 1.1): the agent model specifying the types of agents to form the actual system, the service model specifying the services to be implemented by these agent types, and an acquaintance model depicting 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.

2.3   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 agents marketplaces. Specifically, Gaia has the following limitations:

1 Gaia cannot explicitly model and represent important social aspects of a multi-agent system. Among them, Gaia cannot explicitly model the organizational 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. These factors, together with the fact that Gaia implicitly assumes all agents to be cooperative, make it clearly unsuitable for open agents systems.

2 Gaia employs Gaia-specific notations for representing roles and protocols. Also, these notations – although simple and easy to catch – may be somewhat poor for expressing complex problems such as, e.g., complex multi-phase interaction protocols.

In addition, a number of further limitations can be identified. although these will be dealt less extensively by this paper, such as: the lack of requirements modeling phase, the lack of appropriate environmental modeling, and the lack of domain knowledge modeling.

3.   Gaia v.2

The official extension of Gaia, to which we will refer here as Gaia v.2, extends Gaia based on the key consideration that an organization is more than a simply a collection of roles, as it was considered in the first version of Gaia, and that additional organizational abstractions are to be identified (Zambonelli et al., 2003).

In particular, in Gaia v.2, in addition to roles and protocols, the environment in which a multiagent system is immersed is elected to a primary analysis and design abstraction. The environment abstraction explicitly specifies all the entities and resources a multiagent system may interact with to reach the organizational goal. In the original version of Gaia, the description of the environment was implicit in the definition of the permissions associated to role, a choice that does not promote a clear understanding of the overall system and that is of no help to capture those interactions between agents that may occur via the mediation of the environment.

In addition to the environmental model, two further organizational abstractions comes into play in Gaia v.2, namely, the organizational rules and the organizational structures.

Organizational rules aim to specify some constraints that the organization will have to observe. Organizational rules may be global (concerned with all the roles and protocols), or just concerned with the relations between some roles, between some protocols, or between some roles and protocols in the multiagent system. Organizational 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 instead to be prevented by the organization.

Organizational structures, on the other hand, aim to make explicit the overall architecture of the systems (the position of each role in the organization and its relations with other roles) and its choice, a choice that was instead implicitly defined by the role model in the original Gaia.

Organizational rules and organizational structures are strictly related, in that organizational rules may help designers in the identification of organization structures that more naturally suit these rules. In this sense, Gaia v. 2 recognizes that the role model should explicitly derive from the choice of the organizational structure, and not viceversa.

Considering all the above, an overview of the Gaia v. 2 methodology with its models and their relationships is presented in Figure 1.12.

3.1   Analysis in Gaia v.2

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 terms of computational variables/resources) in which the multiagent system will be situated.

- The preliminary roles model. As in the original version of Gaia, the notion of roles abstracts from any mapping into agents (this issue will be considered in the design phase). However, in Gaia v.2, the analysis has to avoid the imposition of a specific organizational structure implicitly defined via the role model. Instead, Gaia v.2 prescribes to leave the role model incomplete (i.e., with some of the inter-role interactions not identified), since only an accurate
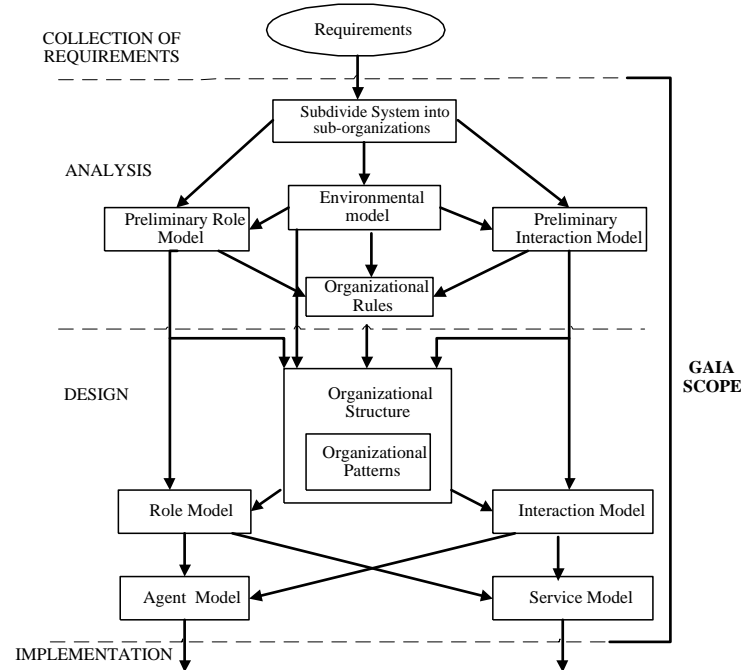
Figure 1.6. Models of Gaia v.2 and their Relationships (Zambonelli et al., 2003)

identification of the organizational structure – to take place in the design phase – will enable to understand exactly which roles will interact with which others.

- The preliminary interaction model, which, as in the case of the role model, abstract away from the organizational structure and has thus to be preliminary (e.g., with some of the partners in a protocol undefined).

- 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 how, in an open multiagent system, external self-interested agents can execute without undermining the overall consistency of the developing multiagent system.

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.

3.2   Design in Gaia v.2

The design phase includes the following sub-phases:

- Definition of the overall architecture of the systems, i.e., of the organizational structure, taking care that it accommodates all preliminary roles and interactions identified in the analysis phase, and taking care that the adopted structure facilitates the enactment of the organizational rules.

- 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 exploit, in this phase, existing catalogs of organizational patterns.

- Revision and completion of the preliminary role and interaction models, on the basis of the adopted organizational structure.

- Definition of the agent model specifying agent types (a set of agent roles) and agent instances, as in the original version of Gaia.

- Definition of the services model, as in the original version of Gaia, to specify the main services (blocks of activities with their pre- and post-conditions) that agent types have to provide.

3.3   Discussion

The general framework of Gaia v.2 exploits consistently novel organizational abstractions to overcomes some of the limitations identified in the original version of Gaia. Specifically, Gaia v.2 is more oriented to designing and building systems in complex, open environments. It is easy to see, in fact, that the explicit adoption of an environmental model, of the organizational rules and of the organizational structures, enables capturing and modeling the agent marketplace example in a more effective and flexible way.

The need to request goods and/or a service usually implies, in agent marketplaces, 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. However, the choice of which specific process to adopt for this transactions represents an important design choice, that should be made explicit, as in Gaia v.2, and that require a proper identification of the environmental characteristics and of the organizational rules .

In a closed system, making buyer and sellers interact directly, without the mediation of any auctioneer agents, may be satisfactory since not self-interested behavior is likely to occur. However, in an open system, specific organizational rules may be identified that should rule the interactions between roles and that should drive the identification of the organizational structures. If we assume that agents are not benevolent to each other (as in real-world open auctions), the need to properly constraints the way agents negotiate (e.g., via a rule requiring buyers to submit bids to sellers in an ordered and monothonically increasing way) may require adopting an organizational structure in which a central role is given to an agent playing the Auctioneer role. Also, the explicit representation of the environment (i.e., a computational environment made up of goods, prices, bids) may enable identifying some problems such as, e.g., the fact that specific bids should not be made public or should not changed arbitrarily by agents.

4. The ROADMAP Methodology

Another extension to Gaia is the ROADMAP one, proposed at the University of Melbourne and firstly described in (Juan et al., 2002). It also is worth outlining that – although coherency of presentation requires to introduce Gaia v.2 before ROADMAP in this chapter – ROADMAP has been proposed earlier in time than Gaia v.2.

ROADMAP started as an attempt to extend the original version of Gaia with: a dynamic role hierarchy (as a way to deal with open agent systems), additional models to explicitly describe the agent environment (as Gaia v.2 does), and the agent knowledge (a feature that is very important in intelligent agent systems and that is neglected by both Gaia and Gaia v.2). From Gaia, ROADMAP inherits – other than the basic underlying process model – the organizational view on multiagent systems, and the basic definitions of roles, protocols, agents and services. However, over time the semantics of these concepts in ROADMAP has become quite different from Gaia, so that ROADMAP can, to some extent, be considered a methodology on its own.

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 structures, responsibilities and business procedures constrains individuals in a human organization. To some extent, the role hierarchy plays in
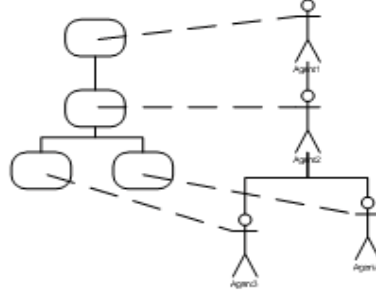
Figure 1.7. System viewed as a computational organization made up of the role hierarchy and the agent hierarchy
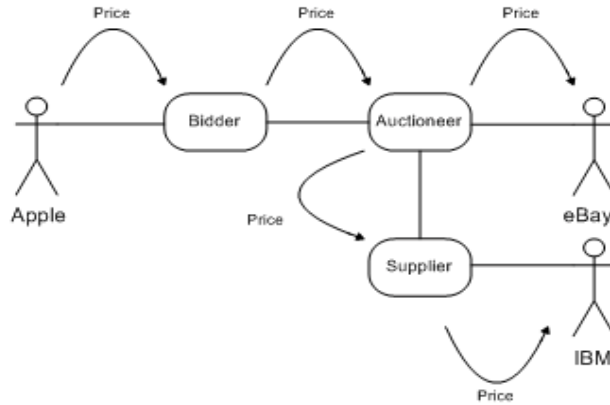
Figure 1.8. Message passing between agents via their roles

ROADMAP a similar role that organizational structures and organizational rules plays, altogether, in Gaia v.2.

Roles and protocols, as in Gaia, are first class entities in ROADMAP that, unlike in Gaia (and in Gaia v.2), have concrete runtime realization in ROADMAP. In a ROADMAP organization, in fact, 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

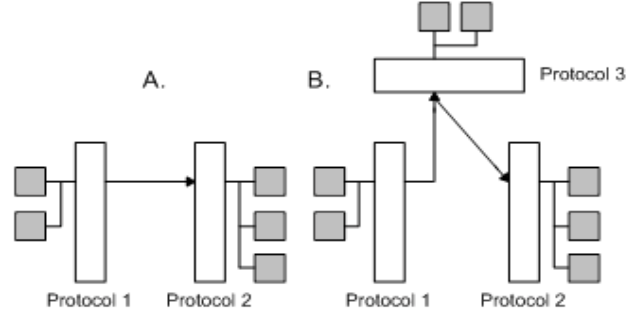*Figure 1.9.* Re-routing interaction with protocols

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.

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). Thus, ROADMAP overcomes the rather poor notation for protocols adopted by Gaia and inherited by Gaia v.2.

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
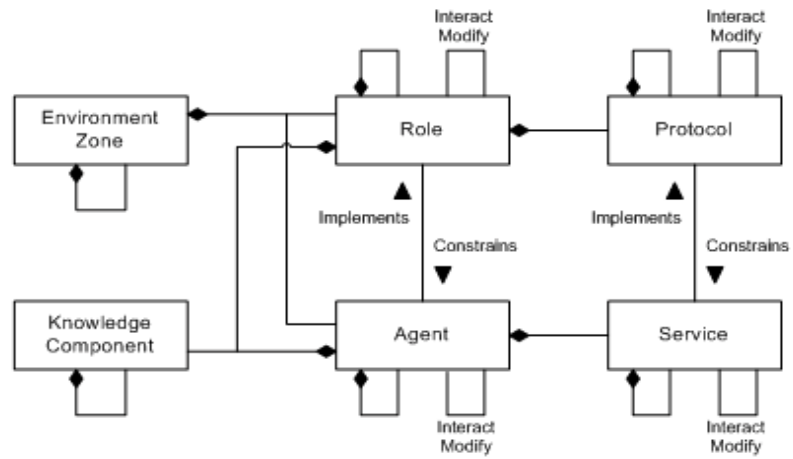
Role Schema: **Supplier**
Description: The role of proposing a service and/or good in the "Auction" model (the role of the agents class Sellers).
Sub-Roles: `Marketing, Sales, ServiceStaff //sub-roles`
Knowledge: Sales History //history of previous sales Market Trend //knowledge on the current market 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 1.10. A sample ROADMAP role definition

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 metamodel (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 software components.

16

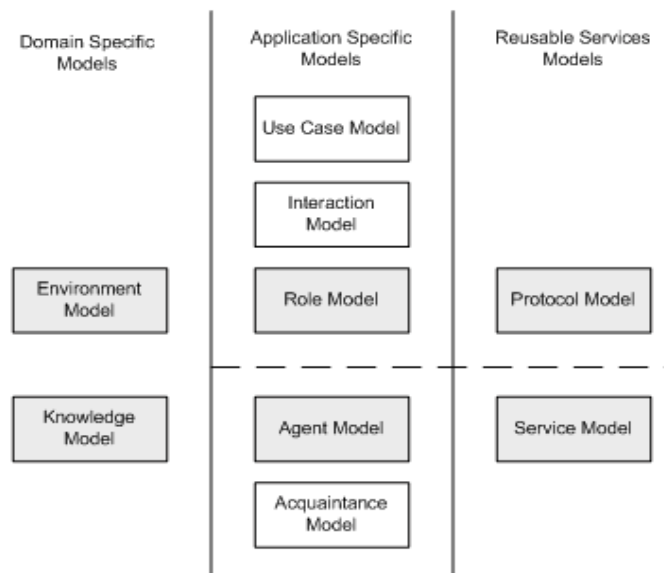*Figure 1.11.*   The ROADMAP meta-model

Figure 1.12.   Models in the ROADMAP methodology

5. Extending Gaia with AUML

Gaia notation for representing roles, protocols, and multiagent organizations in their whole – a notation which is not substantially enriched or extended in Gaia v.2 – is quite 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 (Shehory and Sturm, 2001). 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. Although ROADMAP partially overcomes these problems by introducing richer notations, the possibility to adhere to existing standard notations has to be evaluated.

5.1 AUML: Key Concepts

Our proposed extension is to re-use Agent UML (AUML) (Bauer et al., 2001), 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 then 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 Gaia v.2. Specifically, AUML offers a quite poor notation in covering the organizational structures and does not consider the organizational rules (Parunak and Odell, 2001). It presents some barriers to adapt to complex and open systems with self-interested behavior.

5.2   The Gaia Interactions Model in AUML

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 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 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
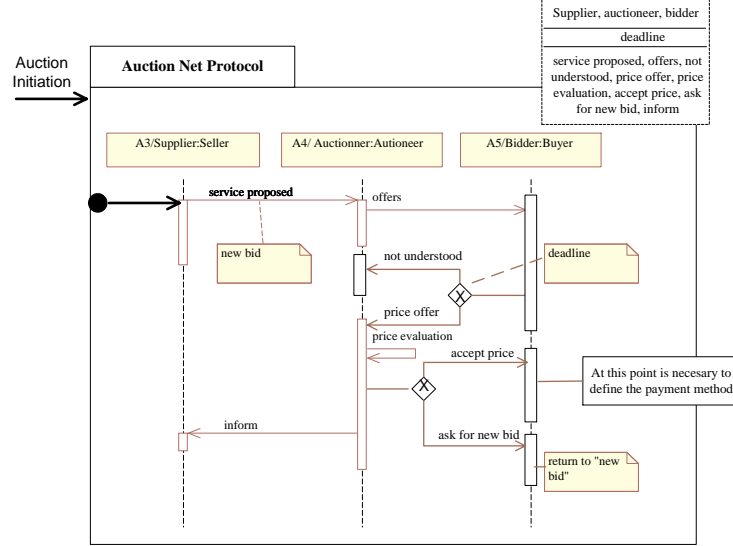
Figure 1.13. The auction model protocol

their goals. The latter two aspects are described in Gaia using natural language, introducing possible ambiguities and misunderstandings.

6. Open Issues

Although the above described extensions are important steps towards the improvement of the Gaia methodology, several other directions for improvements can be identified.

One of these directions related to the modeling and specification of the multiagent systems environment. To model the environment, Gaia uses a notation inspired by FUSION for operation schemata (Coleman et al., 1994) complemented (in Gaia v. 2) by a graphical representation of the spatial, physical, or logical relationships between environment resources and agents. Other authors give increasing importance to environment modeling, which may be captured using traditional object diagrams (see for example (Parunak and Odell, 2001)). However, in most of the cases (as well as in Gaia) these representations fails in properly capturing the dynamic aspects of the environment. Multiagent systems environments may have their own dynamics, which can notably influence the execution of a multiagent system, and such dynamics should be deserved a specific modeling for the analysis and design of a multiagent system to be effective.
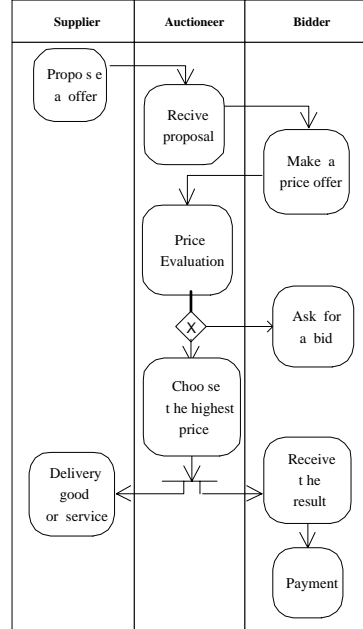
Figure 1.14. The auction model protocol activity diagram

In addition, the organizational structures in Gaia are currently modeled using a simple, non standard, notation, simply expressing relationships among roles (or agents) and complemented by graphical representations and textual comments aimed at enriching the semantics. These representation, as well as the representation of the organizational rules (currently exploiting FUSION-like notation and temporal logic), may be improved.

Another important limitation refers to the lack, in Gaia, of any requirements modeling phase. Although ROADMAP goes in that direction by proposing the use of case models to capture functional requirements, this is not enough. For instance, specific agent methodologies exist that gives the necessary emphasis on the requirements modeling phase. Among them, Tropos (Bresciani et al., 2001) seems to be the most formal, complete, and consistent. In the future, it may be useful to explore the opportunity of exploiting and integrating in Gaia the models and notations already exploited in Tropos for requirements engineering.

7. Conclusions

This chapter has summarized the key characteristics of the original version of the Gaia methodology, and has presented three extensions that have been proposed to it in order to overcome its limitations. On the one hand, ROADMAP and the second version of Gaia, Gaia v.2, extend the original methodology with additional abstractions that are necessary to make Gaia suitable for the analysis and design of complex open agent systems. On the other hand, a proposal to integrate the standard AUML notation in the Gaia process can make Gaia more expressive and easier to be accepted by software engineers.

Although the authors consider Gaia – when enriched with the extensions described in this paper – as one of the most effective methodologies for the analysis and design of multiagent systems, they are also aware of several limitations currently affecting it For instance, the lack of a requirements modeling phase and the need for adopting even richer and expressive notations than Gaia and AUML currently provide may require further research work.

Acknowledgments

References

Bauer, B., Muller, J. P., and Odell, J. (2001). Agent UML: A formalism for specifying multiagent software systems. *International Journal of Software Engineering and Knowledge Engineering*, 11(3):207–230.

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., and Mylopoulos, J. (2001). A knowledge level software engineering methodology for agent oriented programming. In *Proceedings of the 5th International Conference on Autonomous Agents*, pages 648–655. ACM Press, Montreal (CA).

Coleman, D., Arnold, P., Bodoff, S., Dollin, D., Gilchrist, H., Hayes, F., and Jeremas, P. (1994). *Object-Oriented Development: the FUSION Method*. Prentice-Hall International, Hemel Hampstead (UK).

Juan, T., Pierce, A., and Sterling, L. (2002). ROADMAP: Extending the Gaia methodology for complex open systems. In *Proceedings of the 1st ACM Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 3–10. ACM Press, Bologna (I).

Parunak, H.V.D. and Odell, J. (2001). Representing social structures in UML. In *5th international conference on Autonomous agents*, pages 100–101. ACM Press.

Shehory, O. and Sturm, A. (2001). Evaluation of modeling techniques for agent-based systems. In *Proceedings of the 5th International Conference on Autonomous Agents*, pages 624–631. ACM Press, Montreal (CA).

Wooldridge, M.J., Jennings, N. R., and Kinny, D. (2000). The Gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(3):285–312.

Zambonelli, F., Jennings, N. R., and Wooldridge, M.J. (2003). Developing multiagent systems: The Gaia methodology. *ACM Transactions on Software Engineering and Methodology*, 12(3):417–470.