ITACASE: A TOOL SUPPORTING THE DESIGN OF MAS USING THE GAIA METHODOLOGY

Luca Cernuzzi
Universidad Católica “Nuestra Señora de la Asunción”, DEI, Asunción, Paraguay, 1364
lcernuzz@uca.edu.py

Javier Ferrés
Universidad Católica “Nuestra Señora de la Asunción”, DEI, Asunción, Paraguay, 1364
javier_ferres@uca.edu.py

Alethia Hume
Universidad Católica “Nuestra Señora de la Asunción”, DEI, Asunción, Paraguay, 1364
alehume@gmail.com

Francisco Milleres
Universidad Católica “Nuestra Señora de la Asunción”, DEI, Asunción, Paraguay, 1364
fmilleres@gmail.com

Franco Zambonelli
Universitá di Modena e Reggio Emilia, DIMGI, Reggio Emilia, Italia, 42100,
franco.zambonelli@unimo.it

Abstract
Gaia was one of the first agent-based methodologies to exploit the organizational abstractions, to provide clear guidelines for the analysis and design of complex multiagent systems. Moreover, the clear processes and models Gaia proposes have awakened a great interest for such methodology in the agent research arena. Nevertheless, a current successful design methodology should include some other strategic factors to obtain its adoption in the industrial environment. Among them, the support of a specific CASE tool to simplify the work of the designer according to the methodology design process may encourage the industrial software engineers to use it in their real world projects. For this purpose, the present study introduces ITACASE, a CASE which covers all the phases of Gaia allowing agent engineers to produce and document the corresponding models.

Keywords: CASE tool, Gaia, Multiagent systems Design, Software Engineering.
1 Introduction

Since the 1980s, agent-based computing has attracted an increasing amount of interest from the research and business communities, due to its potential to significantly improve the development of high-quality and complex systems. With the increasing acceptance of agent-based computing, a great deal of new research related to the identification and definition of suitable models, tools, and techniques to support the design and development of agent-based systems has emerged. This research area is generally identified as Agent-Oriented Software Engineering (AOSE). Among the others, a great deal of efforts in the AOSE area focuses on the definition of methodologies to guide the process of developing multi-agent systems (MAS) [2],[9] and [10]. However, there is a clear lack of standards in notations, methods, tools, and practices, which introduces some barriers to convert the agent paradigm in a mainstream for the software engineering industry. More specifically, the design and construction of multi-agent systems suffers from a lack of tools supporting these processes [8]. A possible way to reduce the gap between the software engineering industry and the agent paradigm, is by creating and adopting tools that may support engineers and designers in the construction of multi-agent systems.

Several tools are necessary during the different stages in the design and construction of a multi-agent system (some authors tend to classify them in three categories: design, implementation and deployment tools) [15]. In particular, in the design phase we need tools that will allow the specification of the system (also using formal languages when needed), its validation, and (possibly) its capability of offering good automatic code generation. This work focuses on one of these needs, proposing a design tool for multi-agent systems specification.

Specifically, among the AOSE methodologies we devote special attention to the Gaia proposal, [16] which was the first complete methodology to guide the process of developing a MAS from analysis to design, and which reaches a widespread acceptation among the agent community. Moreover, we consider that Gaia may be especially significant for the analysis and design of open MAS. However, some limitations of Gaia have been pointed out [6] [8]. Among them, the lack of a specific Computer Aided Software Engineering (CASE) tool to support the work of the designer according to the Gaia methodology. For this purpose, the present study focus on ITACASE, a CASE which covers all the phases of Gaia, allowing agent engineers to produce and document the corresponding models.

The remainder of this paper is organized as follow. Section 2 briefly introduces the Gaia methodology. Then Section 3 presents the ITACASE, an illustrating example of the utilization of the tool. The section 4 is devoted to analyze the advantages and limitations of the proposal and to discuss some related works. Finally, Section 5 concludes and outlines open research issues.

2 A brief introduction to GAIA

Gaia [16] focuses on the use of organizational abstractions to drive the analysis and design of MASs. Gaia models both the macro (social) aspect and the micro (agent internals) aspect of a MAS, and devotes a specific effort to model the organizational structure and the organizational rules that govern the global behavior of the agents in the organization. Gaia proposes three mainly phases, namely, the analysis, the architectural design, and the detailed design.

The goal of the analysis phase in Gaia, covering the requirements in term of functions and activities, is to firstly identify which loosely couple sub-organizations could possibly compose the whole system, and then for each of these, produce four basic abstract models: (i) the environmental model, to capture the characteristics of the MAS operational environment; (ii) a preliminary role model, to capture the key task-oriented activities to be played in the MAS; (iii) a preliminary interaction model, to capture basic inter-dependencies between roles; and (iv) a set of organizational rules, expressing global constraints/directives that must underlie the MAS functioning.

The above analysis models are used as an input to the architectural design phase. In particular, the architectural design phase is in charge of defining the most proper organizational structure for the MAS, i.e., the topology of interactions in the MAS and the control regime of these interactions, which most effectively
enables to fulfill the MAS goals. Once the most appropriate organizational structure is defined, the role and interaction models identified in the analysis phase (which were preliminary, in that they were not situated in any actual organizational structure) can be finalized, to account for all newly identified interactions and possibly for newly identified roles.

Besides the architectural design phase, the detailed design involves identifying: (i) an agent model, i.e., the set of agent classes in the MAS, implementing the identified roles, and the specific instances of these classes; and (ii) a services model, expressing services and interaction protocols to be provided within agent classes. The result of the design phase is assumed to be something that could be implemented in a technology neutral way. In figure 1 we can see a general view of Gaia with its phases and models.

![Figure 1. Models of the Gaia methodology and their relations in the Gaia process.](image)

### 3 ITACASE: a design tool supporting Gaia

ITACASE was conceived as a designing tool supporting the Gaia methodology. This tool aims to guide the designing process through a friendly user interface, and to fulfill the absence of designing tools supporting Gaia. Thus, it tries to increase Gaia’s widespread acceptance among agent orientation paradigm and to pinpoint its major drawbacks through real testing and utilization.

The design tool was developed with the Microsoft .NET 2.0 framework. The Integrated Development Environment (IDE) used for the development was the Microsoft Visual C# Express Edition 2005 ([http://msdn.microsoft.com/vstudio/express/visualcsharp/default.aspx](http://msdn.microsoft.com/vstudio/express/visualcsharp/default.aspx)). We decided to adopt the Rational Unified Process (RUP) [11] for the development of the ITACASE for different reasons. Despite the stages, and consequently the components of the CASE tool, were well defined by the Gaia methodology, the specific behavior of each stage was quite open. Moreover, ITACASE has to offer a great flexibility to agent engineers in order to analyze and design different types of MAS. Thus, an evolutionary and (partially) spiral process was more adequate than a linear one for dealing with such open environment.

The utility was deployed as a stand-alone application. It deploys all services locally, and stores the information generated by the user in XML files.

The appearance of the application (see for example figure 2) and how the user interacts with it, is defined according to the different phases of Gaia. There is a progress bar, which reflects the coverage of the different
phases and guides the designer according to the methodology. First, the designer defines the different elements in the Analysis Phase, i.e. sub-organizations, environment’s resources, preliminary roles, preliminary interactions and the organizational rules. Second, in the Architectural Design Phase, the designer has to define the organizational structure and patterns, and to complete the Role Model and Interaction Model according to the organizational structure adopted. Finally, in the Detailed Design Phase, the designer defines the agents which play the different roles and the corresponding services.

To provide a better description of the graphical user interface (GUI) of ITACASE, we will focus on its main windows components, menu bar items, tool bar buttons, and its functionality. The main window has three components (see figure 2), the progress bar on the right side, the properties panel on the left side and the main panel on the center. The progress bar contains a graphical representation of Gaia’s models described above. In the main panel, the user defines the attributes according to the active model in the progress bar. The properties panel covers the definition of component (e.g. roles, resources, protocols, etc.) properties defined in the main panel. In the toolbar several buttons can be devised, namely the new project, save, open project, print, cut, copy, paste, and help buttons. The menu bar items are the file, view, project and help menu. The file menu contains all file related functionalities. The view menu can personalize the visualization of the different main panels, for instance, the designer can allow or disallow the visualization of the progress bar or the properties panel. The project menu covers the different phases of the progress bar, and the generation of a graphical representation of the project. Finally, the help menu contains a detailed guide of ITACASE use and functionalities.

To better introduce ITACASE we put its work on a case study. We considered an agent-based system for supporting the management of an international conference. Setting up and running a conference is a multiple phase process, involving a considerable number of individuals and groups. Authors have to submit their papers in the submission phase, and the reviewers have to evaluate the papers in order to accept or reject them. Then, authors need to be notified about the acceptance or rejection of their work. In case of acceptance, authors have to prepare a camera-ready version for the final proceedings. Once the general goal and functionalities of the system-to-be are defined, we start with the analysis phase.

3.1 Analysis Phase

The main goal of the analysis phase is to organize the collected specifications and requirements for the system-to-be into environmental, preliminary role and interaction models, and a set of organizational rules, for each of the sub-organizations composing the overall model [16].

![Figure 2. Definition of sub-organizations.](image-url)
In the conference management organization example, several sub-organizations can be devised. In figure 2 we can see the definition of the different sub-organizations such as submission, review and publication. The user defines the sub-organization in the main panel, and sets its properties such as description and name in the properties panel. For space reasons we reduce our example to the reviewing sub-organization.

The environmental model can be viewed as a list of resources; each associated with a symbolic name, characterized by the type of actions that the agents perform on it, and possibly associated with additional textual comments and descriptions [16]. The designer elicits the different resources of the overall conference management. The name, a short description, and the type of actions that an agent will have on the selected resource can be defined in the resource properties panel. Thus, in the example of figure 3, the designer can specify the ReviewForms resources (one per each assigned paper) which may be changed (i.e. fulfilled) by reviewers.

![Definition of Resources](image)

Figure 3. Definition of Resources.

The preliminary role model is intended to identify some characteristics of the systems that are likely to remain the same. These characteristics are the basic skills of agents such as roles. The user defines them in the role properties panel with a name, a short description, the protocols and activities that they implement, permissions that they have over the resources and the responsibilities rules (Liveness and Safety). In figure 4 we present a graphical view of such a model in ITACASE. For instance, the designer adds an agent playing the Reviewer role which implements the ReviewPaper activity and the ReceivePaper protocol. The Reviewer defined has permissions over papers and review forms, and its mayor responsibility is that the number of papers has to be equal to the number of review forms.
The Preliminary Interaction Model captures the dependencies and relationships between the various roles in the MAS organization. In the conference management, specifically in the review sub-organization, several protocols and activities can be defined. In figure 5 the user defines the Receivepaper protocol with a name, a description, an input, an output, initiators and partners. These properties are defined in the protocol properties panel, and new relationships and dependencies can be added in the main panel.

General relationships between roles, protocols and between roles and protocols are best captured by organizational rules. In figure 6 we can see, for example, the two kinds of organizational rules. Liveness rules capture the dynamic of the organization. For instance, the designer sets that a reviewer can not review his own
paper. Safety rules define global time independent invariants that must be respected. In our example we can see that the user defines that the ReviewCatcher, ReviewCollector and Reviewer roles have to be played with a specific order.

Figure 6. Definition of Organizational Rules.

3.2 Architectural Design Phase

The output of the Gaia analysis phase systematically documents all the functional (and to some extent the nonfunctional) characteristics that the MAS has to express, together with the characteristics of the operational environment in which the MAS will be situated. These structured specifications have to be used in architectural design to identify an efficient and reliable way to structure the MAS organization, and to complete accordingly the preliminary role and interaction models [16]. The definition of the organizational structure consists in the definition of inter-role relationships. In figure 7 the designer defines that the role ReviewCatcher has to control the Reviewer in order to get the reviews previously submitted.
Additionally, ITACASE allows depicting a graphical representation of the organizational structure using other graphical software like Microsoft Visio™. In figure 8 it is possible to observe the structure adopted for the reviewing organization of the conference management case study.

After the definition of the organizational structure, the user completes the definition of the role and the interaction models identified in the analysis phase. In this phase, a critical decision concerns the selection of an adequate organizational structure, possibly coherent with the real/world reviewing organization. Such decision may introduce new roles and perhaps new interaction protocols among roles. Thus, with a better understanding of how roles will interact according to the organizational rules and organizational structure, the
user can add new roles (in the same way explained in the Preliminary Role Model; see figure 4) or refine them accordingly. At this stage, ITACASE allows the designer to edit and modify previously defined roles and protocols.

3.3 Detailed Design Phase

An agent is an active software entity playing a set of roles. Thus, the definition of the agent model accounts for identifying which agent classes are to be defined to play specific roles, and how many instances of each class have to be instantiated in the actual system [16]. Since the different roles for the sub-organization have been already defined, ITACASE allows designers to retry them from a list and directly assign them to the agent under definition.

In our conference management example in figure 9, the designer defines an agent with the name PC Member in the Agents windows in the main panel. In the properties panel, the user sets the agent properties such as its name, a short description, the roles the agent will play, the services that the agent will give, and the minimal and maximal number of instances that the agent will present in the MAS.

On the other hand, as its name suggests, the aim of Gaia service model is to identify the services associated with each agent class or, equivalently, with each of the roles to be played by the agent classes [15].

Once again, to facilitate designers, ITACASE allows retrying a specific service from a list of services that have been already defined and directly assign them to the agent under definition.

As an example, consider the PC-Chair agent. The PC-Chair agent implements the service `collectandcontrol` defined in the service model (figure 10). The service `collectandcontrol` receives a request for reviewing as an input and a completed review form as an output. Its precondition is that the agent has not played already that service a `max_instances` number of times. As a postcondition, we can see that the review form has been correctly completed. The agent or agents implementing this service are specified in the service properties panel, as the different properties listed above and may be retried from the Agents database of ITACASE.

![Figure 9. Agent Model specification.](image)
4 Discussion and Related works

ITACASE is the result of an incremental development process mainly carried out at the Catholic University “Nuestra Señora de la Asunción”. It has been used in several case studies to fix the first limitations until obtaining the actual version. As we have seen in the previous sub-section, ITACASE allows the documentation of the functionalities as well as the overall MAS organization according to Gaia’s notations and models. It is a design tool with no intention to support agent engineers in all the life cycle MAS process. Despite this, it might provoke that agent engineers adopt the Gaia methodology for their real-world projects and offers some improvement to agent designers.

Since ITACASE supports the Gaia process, it also inherits some Gaia limitations, such as:
- Gaia adopts a waterfall like software development process model which implies some kind of incrementality, but that is too linear and consequently not useful for changing requirement situations [5];
- Gaia notations are far to be standards and indeed accepted in the industrial arena, for this purpose a particular extension to Gaia has been proposed, namely Gaia+AUML [7] [8];
- Gaia uncovers all the software development life cycle and particularly does not deal with the requirement engineering phase [5].

With regards to the first limitation, ITACASE supports a non-linear approach to software development. In effect, it is possible to go back and forth between the different phases and freely switch among the different models proposed by Gaia. For example, the possibility of going back from the design to the analysis phase gives the user the opportunity to fix some wrong assumptions or ambiguous requirements, consequently, this improves the linear approach implicitly adopted by Gaia.

The second limitation is very common for all the AOSE methodologies with regards to the organizational structure notations. To this purpose, ITACASE provides a partial solution to this problem allowing designers to specify the organizational structure using other software graphical tools such as Microsoft Visio™. Clearly, the use of a particular graphical tool does not imply any improvement on the notations, but introduces more flexibility and allows designer to produce such a model in an easier way, using different notations. Nevertheless, the notation problem is not reduced to the organizational structure one. Thus, in the future, ITACASE has to support more standard notations like AUML [1].

Probably, the main issue for the use of a CASE tool is the opportunity of automatic generation of code. In this sense, since ITACASE follows the Gaia process, it limits itself to the analysis and design support. Such limitation avoids the possibility of supporting a model driven development approach which is a very
prominent discipline in software engineering nowadays. Consequently, it has to be overcome in the future for the really use of ITACASE in the industrial arena.

Other interesting CASE tools and development platforms have been proposed in the AOSE community. On one hand, among the platforms environments we can mention Zeus [12], Jacks [13], MAGE [18], etc. Nevertheless, it is hard to compare such development environment with ITACASE since they do not support a specific methodology process. Consequently, independently from the adopted methodology for the analysis and design of a MAS, agent engineers, may chose some of such environment for the development of the system-to-be. On the other hand, there are some CASE tools which support the corresponding methodologies. Among them, we can mention the TAOM4E [3], which supports Tropos [4], and the AgentTool [14] which support MASE [14]. It is hard to compare those tools with ITACASE without comparing the corresponding methodologies. Since this issue is out of the scope of the present work, interested readers may find some insights in [5].

However, it is worth to observe a trend of implementing these tools based on the Eclipse platform (http://www.eclipse.org). In effect additionally to TAOM4E, also the authors of MASE are preparing a new version based on this well known platform (http://macr.cis.ksu.edu/projects/agentTool/agentool3.htm). This trend suggest us a future improvement for ITACASE to develop a plug-in for Eclipse instead of a stand-alone application. Such solution may facilitate a greater acception of ITACASE in the software engineering community.

5 Conclusion and future works

A possible way to reduce the gap between the software engineering industry and the agent paradigm is creating and adopting tools that may support engineers and designers in the construction of multi-agent systems.

This paper presents ITACASE, which is a designing tool supporting the Gaia methodology. This tool aims to guide the designing process through a friendly user interface and to fulfill the absence of designing tools supporting Gaia.

ITACASE allows the documentation of the functionalities as well as the overall MAS organization according to Gaia’s notations and models. It is a design tool with no intention to support agent engineers in all the life cycle MAS process. Nevertheless, to some extent it facilitates agent engineers into adopting the Gaia methodology for their real-world projects.

ITACASE supports a non-linear approach to software development which is not covered by Gaia. In effect, going back and forth is possible between the different phases as well as freely switching among the different models proposed by the Gaia methodology.

ITACASE allows designers to specify the organizational structure using other software graphical applications. Evidently, the use of a particular graphical tool does not imply any improvement on the notations, but it introduces greater flexibility, and allows designers to produce in an easier way such a model using different notations.

Different future improvements are being envisionned. A first natural extension has to incorporate the Gaia + AUML notations [7], [8]. Moreover, we are looking for richer notations for modeling organizational abstractions (i.e. organizational structure and organizational rules). In addition, a new version integrated to the Eclipse environment will provide greater acception in the software engineering community. Finally, the more relevant contribution will come from allowing code generation starting from the design models.

Acknowledgements

This work was partially supported by the Italian MUR in the frame of the PRIN project "MEnSA - Agent oriented methodologies: engineering of interactions and relationship with the infrastructures" and also by the Universidad Católica "Nuestra Señora de la Asunción" project "Diseño y Modelado de Sistemas Agentes y Multi-Agentes: Propuestas de Mejoras". Moreover, the authors are grateful to Juan José Bareiro, Oscar Rafael Benítez, Rodney Rodríguez and Miguel García, all of them students at the Universidad Católica “Nuestra Señora de la Asunción”, for valuable comments and support during development.
References


