GAIA4E: A TOOL SUPPORTING THE DESIGN OF MAS USING GAIA

Luca Cernuzzi
DEI, Universidad Católica "Nuestra Seóra de la Asunción", Campus Universitario, Asunción, Paraguay
lcernuzz@uca.edu.py

Franco Zambonelli
DISMI, Universitá di Modena e Reggio Emilia, Reggio Emilia, Italy
Franco.zambonelli @unimore.it

Keywords: CASE tool, Gaia, Eclipse, Multiagent Systems Design, Software Engineering

Abstract: Different efforts have been devoted to improve the original version of Gaia methodology. The more relevant is the official extension of Gaia, exploiting the organizational abstractions to provide clear guidelines for the analysis and design of complex and open multiagent systems. However, now a day a successful design methodology should include some other strategic factors like the support of a specific CASE tool to simplify the work of the designer. Such a tool supporting the Gaia design process may facilitate the adoption of the methodology in the industrial arena. The present study introduces Gaia4E, a plug-in for the ECLIPSE environment which covers all the phases of Gaia allowing agent engineers to produce and document the corresponding models.

1 INTRODUCTION

The Agent Oriented Software Engineering – AOSE discipline has grown in the last few years and many AOSE methodologies have been proposed (Ciancarini and Wooldridge, 2001). Some of them, (e.g. Giorgini et al. 2005; Gómez and Pavón 2003; Zambonelli et al. 2003) are strongly mentioned in the specialized literature. To achieve an extensive success a methodology should include different strategic factors. Among others, we can mention the use of a standard and well known design language (like UML), the support of a specific CASE tool to simplify the work of the designer and the attention for the automatic production of large parts of code.

Such strategic factors may facilitate a broader introduction of the agent oriented methodologies in the industry practice. Among them, our mainly interest in this paper is to focus on creating and adopting tools that may support engineers in developing multi-agent systems according to a particular methodology as advocated in (Zambonelli and Omicini, 2004).

When developing an agent-based system, several tools are necessary during the different stages of the process. In the design phase we need tools allowing the specification of the system (also using formal languages when needed), its validation and (possibly) offering good automatic code generation capabilities.

We are especially interested in the need of such tools for Gaia due to its large acceptation in the agent community.

The Gaia methodology (Zambonelli et al. 2003), explicitly focus on using organizational abstractions to drive the analysis and design of complex and open multiagent systems (MAS). It models both the macro (social) aspect and the micro (agent internals) aspect of a MAS. The design process proposed in Gaia introduces different models, each of them with its own notation. Some extensions of Gaia considered the adoption of more standard notations based on AUML (Cernuzzi and Zambonelli 2004; Cernuzzi et al. 2004), but no proposal have focused on tools supporting the Gaia process.

In this work, we present the design tool Gaia4E (standing for: Gaia for ECLIPSE) which covers all the phases of Gaia allowing agent engineers to document the corresponding models.

Trying to facilitate a larger acceptance and adoption of such a tool we have implemented it as a

plug-in for the ECLIPSE environment, widely used in the object oriented software engineering community. The ECLIPSE Project is an open source initiative that allows the integration of different tools into a single "environment" (for more information, interested reader may see the ECLIPSE web site: http://www.eclipse.org/). Eclipse may be considered a kind of universal tool platform - an open extensible IDE for almost anything related to software engineering design. The incorporation of new tools into the platform is possible trough plug-ins that provides new functionalities to the environment.

The paper is structured as follow: section 2 briefly introduces Gaia methodology with its process, and its models and their relations; section 3 presents the Gai4E tool using an illustrating example to clarify the concept and notations; and section 4 concludes.

2 GAIA IN A NUTSHELL

Gaia (Zambonelli et al. 2003) 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 possibly compose the whole systems and then, for each of these, to produce four basic abstract models: (i) the environmental model, to capture the characteristics of the MAS operational environment; (ii) a preliminary roles model, to capture the key task-oriented activities to be played in the MAS; (iii) a preliminary interactions 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 input to the architectural design phase which 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 fulfil the MAS goals. The definition of the organizational structure has to account for a variety of factors, including the need of reflecting the structure of the real-world organization, the characteristics of the environment, the need of simplifying the enactment of the organizational rules, as well as the obvious need to keep the design as simple as possible. Once the most appropriate organizational structure is defined, the roles and interactions 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 possibly newly identified interactions and roles.

Past 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.

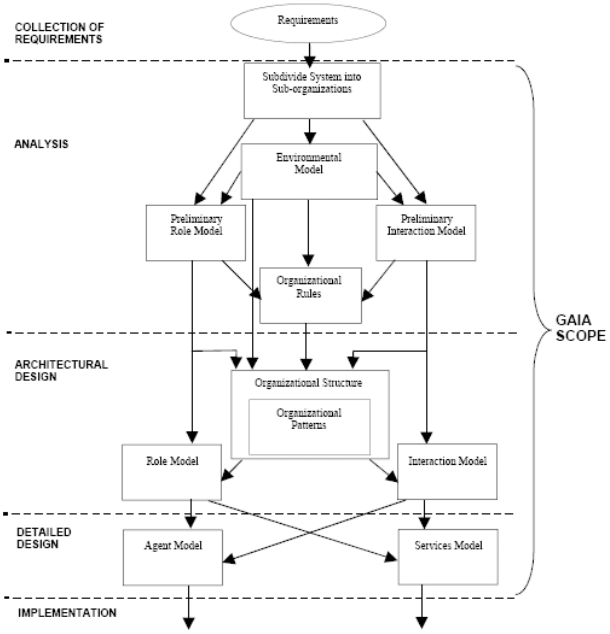

Figure 1: Models of the Gaia methodology and their relations in the Gaia process.

3 GAIA4E: A DESIGN TOOL SUPPORTING GAIA

Now a day a successful design methodology should include some other strategic factors like the support of a specific CASE tool to simplify the work of the designer (Zambonelli and Omicini, 2004).

In this section we present Gaia4E (standing for: Gaia for ECLIPSE), a design tool which covers all the phases of Gaia, allowing agent engineers to document the corresponding models according the specific notations, and which has been implemented as a plug-in for the ECLIPSE environment. ECLIPSE may be considered a kind of open IDE which functionalities could be extended trough plug-ins that are incorporated to the environment.

In order to gain a better understanding of the Gaia4E Plugin, we need to grasp a few concepts related to the Eclipse Platform. The next two paragraphs are straight forward quotations of the Platform Plug-in Developer Guide document that comes within the Eclipse Help Contents.

"The Eclipse platform is structured around the concept of plug-ins. Plug-ins are structured bundles of code and/or data that contribute function to the system. Function can be contributed in the form of code libraries (Java classes with public API), platform extensions, or even documentation. Plug-ins can define extension points, well-defined places where other plug-ins can add functionality."

"The workbench UI plug-in implements workbench UI and defines a number of extension points that allow other plug-ins to contribute menu and toolbar actions, drag and drop operations, dialogs, wizards, and custom views and editors."

The main components in the GUI layout of an Eclipse plug-in are views and editors. An editor is typically used to edit or browse a document or input object (generally a file). A view is more commonly used to navigate a hierarchy of information or display and edit properties for the active editor.

The Gaia4E plug-in makes its main contribution to the UI by extending dialogs and wizards, through one custom editor and two views and arranges them all by providing a specific perspective.

The Gaia perspective includes three different views (see Figure 2). First, located to the left side of the window is the Eclipse Navigator view which allows designer to open an already existing project, to create a new project, etc. The second view, called Gaia properties and situated in the bottom right side, shows the properties of the specific component selected in the active Gaia4E Editor, and allows the user to modify it. The third view is called the Gaia4E Image Viewer and is located in the same space as the former (reachable by clicking its corresponding tab). The purpose of this view is to show an image related to the topology of an Organization in the Architectural Phase.

At last, the editor, known as Gaia4E Editor, lets the user work on a Gaia document file (.gaia extension). The editor is tabbed in the bottom of its window in response to the three phases covered by the methodology (Analysis, Architectural Design and Detailed Design). The user should choose the tab according to the phase she or he wants to work on. After selecting the phase the designer is able to add/delete/modify whatever aspects are involved in that particular phase. For example if the user wants to delete a role specified in the preliminary roles model for the Organization X, the user must click in the lower tab "Analysis", select the X organization in the "Sub-Organizations tree" and go by means of the upper tab to the roles tree, select the role to be deleted and click the delete button in the editor's toolbar.

To better understand the tool we presents each model supported by the Gaia4E using an illustrative example: an auditing agents-based system for the Central Bank of Paraguay (BCP – Banco Central del Paraguay). A multi-layer agent architecture for remote auditing in public institutions has been proposed to solve some deficiencies of the BCP which is in charge of the auditing process of the financial-account management of the credit firms. Figure 3 shows the proposed model. As the reader may observe, the BCP sends different agents to the financial institutions that the BCP needs to audit. In each financial institution the files are validate and modified up to reach the correct information and format. Once the proper agent has completed its auditing activities, the obtained file are compressed and encrypted before being sent back to the BCP. Finally, the BCP sends an electronic receipt to the audited institution.

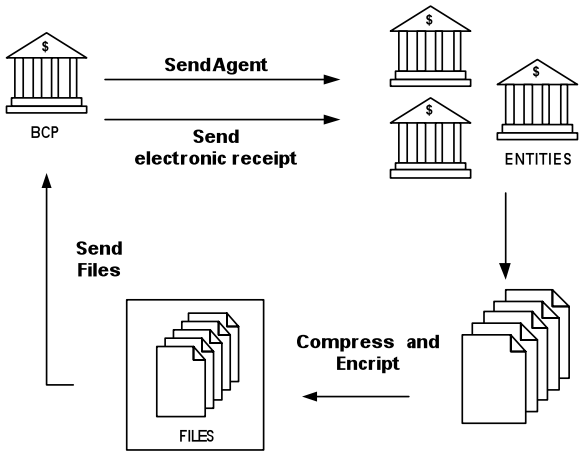

Figure 3: The model of the auditing system for the BCP.

The experimental results achieved with the Auditing Agents BCP system solve the problems identified in the actual working model of the BCP.

3.1 Modelling the BCP Auditing System with GAIA4E

Possible sub-organizations

The auditing system includes four different processes: the correct format verification, the information validity verification, the verification of the relationships between files (also called referential integrity), and the verification of the sum of the debit for each client. Each verification process operates on different data files and includes multiple check activities, one for each of these four different files. Thus, it is natural to think in four different sub-organizations, one for each verification process.

Environmental Model

In this application the environmental model comprises the data bases of the audited banks (accessible if the agent has the corresponding permissions), a set of reports generated by the different agents, and the data base of the BCP.

Preliminary Roles Model

In this model designers capture the basic functional requirements of the application in terms of the roles that the different actors can play. Independently from the organizational structure, the main role is the Auditor which will be in charge of the general auditing process. Moreover, other roles are those related to each of the four verification processes previously identified. Other roles would may appear in the design phase. For space reasons we just present the Auditor role schema (see figure 4).

Preliminary Interaction Model

As for the preliminary roles model, and although this model may experience changes once the organizational structure is defined, some preliminary interaction protocols could be identified. For this model a representative example is presented in figure 5 (for space reasons all the formats, validities and relations verifications are grouped in one agent class for each type).

Organizational Rules

The organizational rules are invariants that agents, playing the corresponding roles, have to observe for maintaining a correct behaviour of the organization.

Different kind of rules may be defined for the BCP auditing system, but for space reasons we just present some examples of the complete set.

Some of these rules may also control the order of the activities. For example, rule 1 means that CreateAgent has to be executed before of ProgrammingAgent; while rule 2 states that ReceiveConfirmation precedes the warning for an ErrorKey or the request for a specific auditing process (in this case the format auditing).

Other rules may specify different constraints for the general behaviour of the organization. For example, rule 4 means that if an Auditor has been validated all the corresponding reports (for Format, Validity, Relation and Sum) must exist.

1. CreateAgent $\rightarrow$ ProgramminAgent
2. ReceiveConfirmation $\rightarrow$ ErrorKey | RequestAuditFormat
3. Key $\neq$ null
4. $\forall$ Auditor, confirmation=true $\rightarrow$ ReportFCA $\neq$ null $\wedge$ ReportFCB $\neq$ null $\wedge$ ReportFCC $\neq$ null $\wedge$ ReportFCD $\neq$ null $\wedge$ ReportVCA $\neq$ null $\wedge$ ReportVCB $\neq$ null $\wedge$ InformeVCC $\neq$ null $\wedge$ ReportVCD $\neq$ null $\wedge$ ReportRCA-CC $\neq$ null $\wedge$ ReportRCB-CA $\neq$ null $\wedge$ ReportRCB-CC $\neq$ null $\wedge$ ReportRCC-CA $\neq$ null $\wedge$ ReportSum $\neq$ null
5. $\forall$ Auditor, home=true $\wedge$ FinalReport $\neq$ null $\rightarrow$ change BD BCP

Organizational Structure

Since the user has the control over the system, she/he is in charge of the creation of the auditor agent. This auditor agent in turn, depends on the checker agent that is in charge of auditor validation. The auditing process starts once the auditor creates all the agents in charge of the different verification processes for the audited bank. Those agents prepare and send a report according to the corresponding validation results. Since the auditor has created all the verification agents, it controls those agents and may eventually decide to eliminate them at any moment.

Thus, a natural organizational structure is presented in figure 6. It is worth to outlines that Gaia4E offers designers to associate graphical files to the documentations. Thus, for the specification of the topology of the organization it is possible to model it with other graphics environments and then insert the model into the Gaia4E corresponding area.

Detailed Design

The detailed design of Gaia considers the Agent Model, in which each role corresponds to a class, and the Service Model. For each auditing process, one Auditor and one Checker agents are instantiated while the other agents (Format, Validity, Relation, and Sum) may or may not be instantiated. If the Auditor agent is not recognized by the Checker these other agents are not instantiated, in all other cases, the verification agents are created when necessary until all errors have been corrected.

In the Service Model the designers identify the services associated to each class specifying inputs, outputs, pre and post-conditions. Inputs and outputs are derived from the Protocols (for the services associated to the agent interaction) and from the Environmental Model (for the services that operate on the resources). Pre and post-conditions derive from the safety properties of roles, from the organizational rules, from constraints on the availability of the resources, and from specific values of resources or data from other agents. For example the service "RequestAuditFormat" needs the "User Configuration" as input; it has the pre-condition "Confirmation=True", and the post-condition "FormatCA≠NULL and FormatCB≠NULL and FormatCC≠NULL and FormatCD≠NULL".

Gaia and Gaia4E do not cover the development and implementation stages. For those activities, in the BCP Auditing System we adopt the Aglets Software Development Kit (ASDK).

4 CONCLUSIONS

In this work, we presented the design tool Gaia4E (standing for: Gaia for ECLIPSE) which supports the work of agent engineers covering all the phases and the models of Gaia. Trying to facilitate a larger acceptance and adoption of such a tool we have implemented it as a plug-in for the Eclipse environment, widely used in software engineering community (especially, in the object oriented arena).

Such strategic factors may facilitate a broader introduction of the Gaia methodology in the industry practice. The Gaia4E CASE was adopted in different industrial projects; one of them was presented in the previous section to illustrate the tool.

Due to the adherence to Gaia, Gaia4E suffer the same limitations. Among them, it does not cover the requirement elicitation phase and the implementation. Moreover, Gaia4E adopts the same notations of the original Gaia. Consequently, relevant improvement to the design tool may be derived from improvements in the Gaia process and notations. In this sense, we are working to include Gaia+AUML (Cernuzzi and Zambonelli 2004; Cernuzzi et al. 2004) into Gaia4E. Additionally, we are exploring methods and notations to be harmonically included into the Gaia process to cover the requirement elicitation phase. Finally, a really attractive open issue which may strongly improve the acceptance of Gaia4E is the capability of the automatic generation of independent agents with their respective software code.

REFERENCES

Cernuzzi, L., Zambonelli, F., 2004. Experiencing AUML in the Gaia Methodology. In: *Proceedings of 6th International Conference on Enterprise Information Systems - ICEIS 2004* (Vol. 3, pp. 283-288), Porto, Portugal.

Cernuzzi, L., Juan, T., Sterling, L., and Zambonelli, F., 2004. The Gaia Methodology. (Chapter book) in *Methodologies and Software Engineering for Agent Systems. The Agent-Oriented Software Engineering handbook*. F. Bergenti, M.-P. Gleizes and F. Zambonelli Editors, (pp. 69-88), Kluwer Publishing.

Ciancarini, P., Wooldridge, M., 2001. Agent-Oriented Software Engineering. *Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering*, Springer Verlag, LNCS, Vol. 1957, 1-24.

ECLIPSE: available in http://www.eclipse.org/

Giorgini, P., Kolp, M., Mylopoulos, J., Castro, J., 2005. Tropos: A Requirements-Driven Methodology for Agent-Oriented Software. (Book Chapter) In: *Agent-Oriented Methodologies*. Idea Group, 20-45.

Gómez, J., Pavón, J., 2003. Agent Oriented Software Engineering with INGENIAS. *Proceedings of the 3rd Central and Eastern Europe Conference on Multiagent Systems*, Springer Verlag, LNCS 2691, pp. 394-403.

Zambonelli, F., Wooldridge, M., Jennings, N. R., 2003. Developing Multiagent Systems: The Gaia Methodology. *ACM Transaction on Software Engineering and Methodology*, vol. 12, Nº 3, 417-470.

Zambonelli, F., Omicini, A., 2004. Challenges and Research Directions in Agent-Oriented Software Engineering. *Journal of Autonomous Agents and Multiagent Systems*, vol. 9, Nº 3, Kluwer Academic Publishers, 253-283

APPENDIX

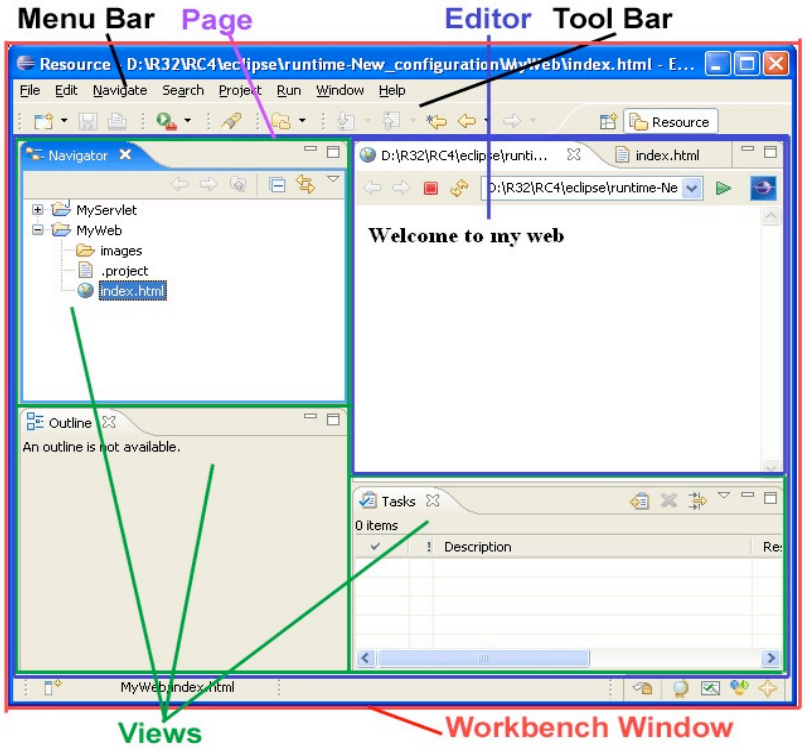

Figure 2: General view of the Eclipse environment.

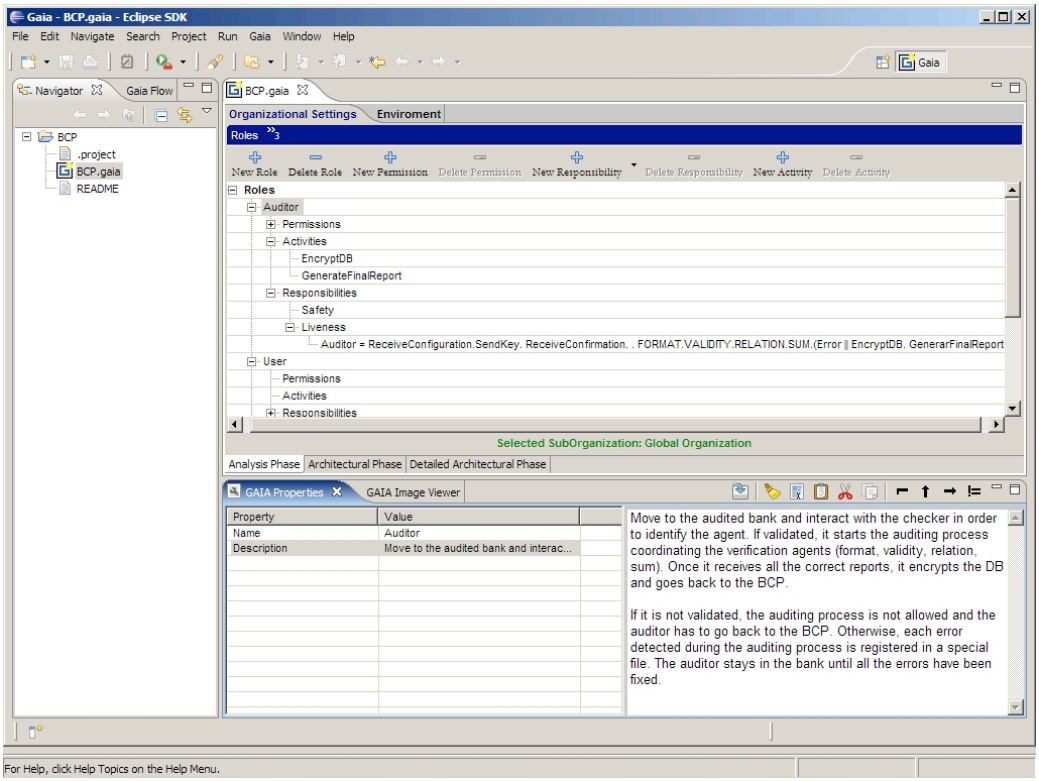

Figure 4: The Auditor Role schemata in Gaia4E

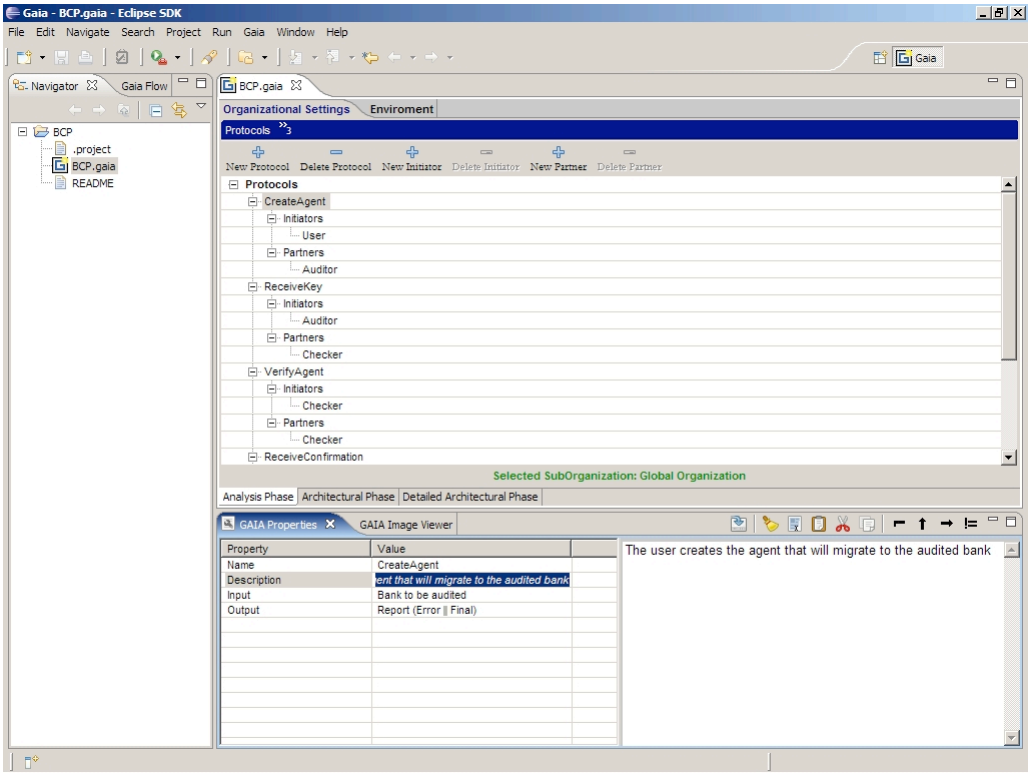

Figure 5: The CreateAgent Protocol Model in Gaia4E

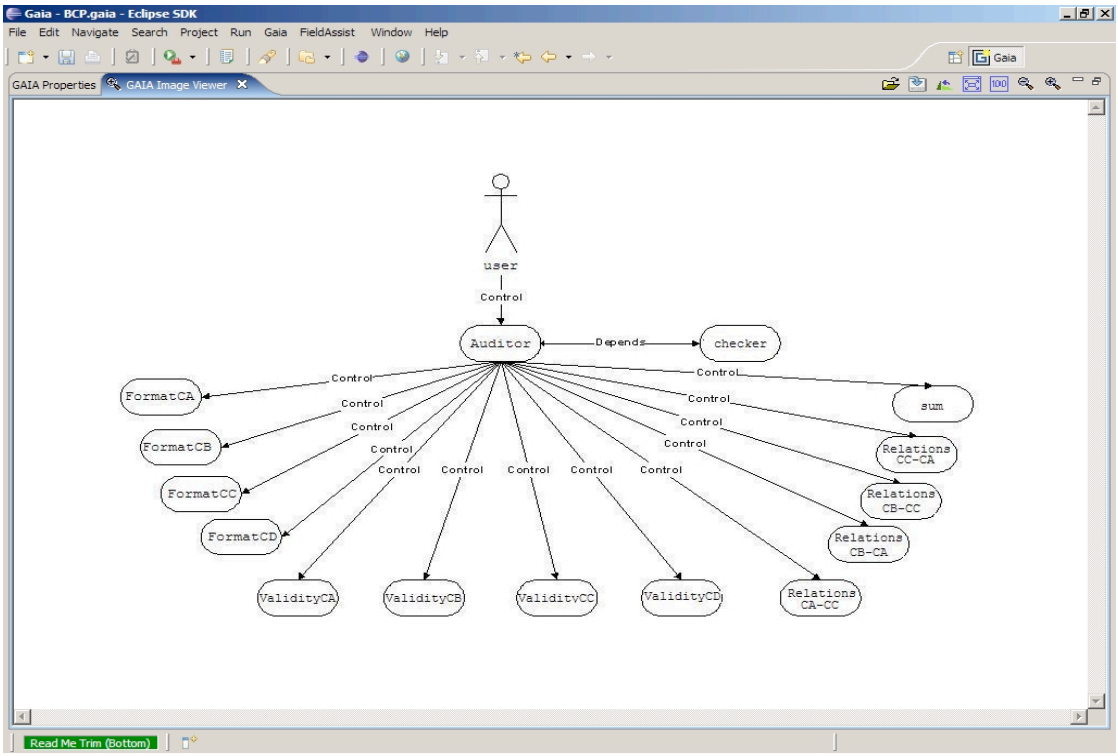

Figure 6: The global view of the Organizational Structure in Gaia4E.