Integrated Computer-Aided Engineering 11 (2004) 151–163
IOS Press

On combining organisational modelling and graphical languages for the development of multiagent systems

Álvaro E. Arenas, Juan C. García-Ojeda and José de J. Pérez-Alcázar*
Laboratorio de Cómputo Especializado, Universidad Autónoma de Bucaramanga, Calle 48 No 39 – 234, Santander, Colombia

Abstract. This article illustrates the application of the Gaia methodology to the development of a system for Selective Dissemination of Information on the Web, as well as the integration of Gaia with AUML. This allows a concrete design closer to implementation and the developer offers a better insight into the multiagent system that is being built. Gaia is an agent-oriented organisational methodology that generates an abstract model based on roles describing the system agents, their services and interactions. AUML, as an agent-based extension of the graphical language UML, uses that model as input and breaks it down into a series of diagrams that facilitates its implementation.

1. Introduction

The search and dissemination of relevant information on the Web has become a great challenge, owing mainly to the fact that the Internet can be considered the largest man-made deposit of information in the world. Several authors have proposed techniques and tools to tackle the excess of information on the Web [15,37]. One of these techniques is the Selective Dissemination of Information (SDI), in which the desired information arrives to a user in a selective manner, according to the user's profile (his/her information needs).

This article illustrates the construction of a system for selective dissemination of information based on agent technology. In a periodic form, the system informs a group of users about the publication of new articles from predetermined digital libraries, according to a pre-established profile of the user. The system includes a module for information extraction based on data contained in web pages of the selected digital libraries [12], and combines techniques for content-based and collaborative filtering of information [4,37]. The combination of these two techniques allows a greater flexibility for scaling the system.

The integration of Gaia and AUML in the construction of SDI systems represents the main contribution of this article. Gaia is a methodology based on the view of a multi-agent system as a computational organisation consisting of various interacting roles [43]. The Gaia output consists of a design still too abstract for implementation. To turn this around, a more concrete design is produced using AUML [9,31]. AUML takes the Gaia design as input and breaks it down into a series of diagrams that facilitates its implementation.

The following section describes some of the topics related to the selective dissemination of information, including aspects of information filtering and extracting. Section 3 presents the application of Gaia in our study. Section 4 illustrates the utilisation of AUML diagrams to generate a detailed representation of the agents in the system. Section 5 introduces the implementation of our system. Finally, Section 6 gathers some concluding remarks and discusses possible extensions to our work.

2. Selective dissemination of information

A system for Selective Dissemination of Information (SDI) allows users to continuously receive new filtered

*Corresponding author. Tel.: +57 7 6436111, Ext. 222; E-mail: jperez@unab.edu.co.

ISSN 1069-2509/04/$17.00

documents based on the user profile. In many of these systems, the responsibility of this process lays upon people, who may be overwhelmed by the great amount of information sources. In order to face the information overload, a series of instruments have been developed such as information filtering and information extraction systems. Following, we will describe these techniques.

2.1. Information filtering systems

Information filtering systems remove irrelevant items from a constant stream of documents [1]. Generally, personalized information filtering systems incorporate the interests of the users in a user profile. They use this profile to filter the data sent to the user. The filtering systems are classified into two categories: content-based systems [29], which choose documents based on content characteristics; and social systems, also known as collaborative [37], which select documents based on recommendations or annotations from other users.

2.1.1. Content-based filtering systems

As mentioned before, these systems choose documents according to the characteristics of their contents. The system looks for items similar to those preferred by the user based on a comparison of contents. These systems have certain disadvantages, primarily with regard to the capture of different aspects of the content (i.e., music, videos and images). Furthermore, in the case of texts, most representations capture only certain aspects of the content, which results in a poor performance. Besides the problem of representation, these systems try to operate in such a way that they recommend similar items to the items already displayed. A few examples of content-based filtering systems are CiteSeer [10], Letizia [25] and Lira [5].

2.1.2. Collaborative filtering systems

In contrast to content-based systems, which can be successfully applied to a single user, the collaborative systems assume that there is a team of users using the system. With this technique, advice to the user is based on the reactions of other users. The system looks for users with similar interests and recommends the items preferred by these users. Instead of calculating similarities between items, the system calculates similarities between users. In the collaborative approach there is no analysis of the item's content. Each item is assigned a unique indicator and a user derived rating. This type of systems can be used for non-text data (i.e., video and sound). Some examples of collaborative filtering systems are: Siteseer [36], Phoaks [41] and GroupLens [36].

2.2. Information extraction system

The Web has become a platform for data intensive applications. The majority of the applications have been converted or are in the process of becoming web-based. For this reason, there is a greater demand for web contents amenable to automatic processing. However, there is the problem that most web data is unstructured or semi-structured [2]. In other words, it is directed toward human processing and not automated tools. The recognition and extraction of the implicit information on the web pages is a current problem, important for web data processing by automated tools [28].

A traditional form of extracting data from web pages is the creation of specialized programs called "wrappers" that identify data of interest and converts them to some suitable format. This method has disadvantages mainly due to the difficulty in writing and maintaining the wrappers. Recently, various tools have been proposed to solve these problems, such as RoadRunner [13], DEByE [23], NoDoSe [3] and Minerva [12]. A good review of these tools can be found in [24].

3. Using Gaia for designing SDI systems

In this section we present the design of a SDI system using the Gaia methodology. A SDI system is a typical example of a complex problem [27,39]. Researchers in the area of the agent-oriented software engineering have described a number of mechanisms to handle this complexity. Such mechanisms are [21]: organization, decomposition and abstraction. These mechanisms will be described in the following sections.

Our SDI system automates the task of periodically notifying the users of the updates from different digital-library magazines such as ACM (www.acm.org) or the IEEE (www.computer.org), according to a pre-established profile.

The prototype uses an information extraction model based on the Minerva formalism for developing wrappers. It also combines content-based and collaborative filtering, basing its predictions on a fusion of judgments between these techniques. It is worth noting that the use of these two techniques presents a series of additional advantages. Firstly, the integration of content-based and collaborative filtering resolves the disadvantages mentioned previously. For example, the use of collaborative filtering remedies the failures of the content-based filtering in treating non-text documents. Secondly, the use of wrapper-generating tools such as

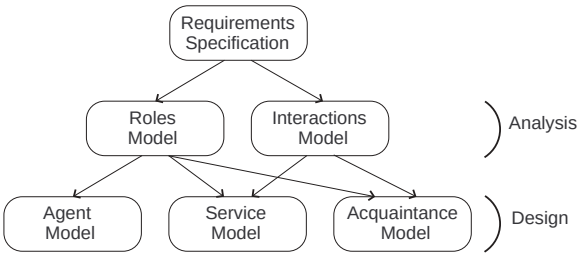

Fig. 1. The Gaia Models.

Minerva may allow a continuous addition of new digital libraries to our system in an automatic or semiautomatic way, thus adding a scaling characteristic to our system.

3.1. The Gaia methodology

We have selected Gaia as our agent-development methodology due to its architectural independence and simplicity [43]. Gaia takes the view that a system can be seen as a society or organisation of agents. It models both the micro-level (agent structure) and macro-level (agent society and organisation structure) of agent development. Figure 1 shows the Gaia models.

In the analysis phase, the role model and the interaction model are constructed. Roles consists of four attributes: responsibilities, permissions, activities and protocols. Responsibilities are of two types: liveness properties – the role has to add something good to the system –, and safety properties – prevent and disallow that something bad happens to the system –. Permissions represent what the role is allowed to do. Activities are tasks that a role performs without interacting with other roles. Protocols are the specific patterns of interaction. These two models depict the system as interacting abstract roles and are then used as input in the design.

In the design phase, the agent model, the services model and the acquaintance model are constructed. The agent model maps roles into agent types, and then creates the right number of agent instances of each type. The service model determines the services needed to fulfil a role in one or several agents. Finally, the acquaintance model represents the communication between agents. Gaia does not cover detailed design, and relies on conventional methodologies for that purpose.

3.2. Analysis phase

As mentioned before, the objective of the Gaia analysis phase is to develop an understanding of the system and its structure, without considering any implementation detail. The analysis consists of a roles model, which identifies the key roles in the system, and of an interaction model, which defines the interactions among the roles.

3.2.1. Identifying the roles in the system

A role can be viewed as an abstract description of an entity's expected function. Typically, a role corresponds to either an individual, a department within an organisation or an organisation itself.

We use an organisational vision in our SDI system, splitting it into the following departments: *Information Retrieval, Selective Dissemination of Information, User Management, Mail Management, Digital Library Management* and *Central Coordination*. Note that the *Mail Management* and *Digital Library Management* are external departments of the organisation (system). We identify the members of each department, each member corresponding to a role, as illustrated in Fig. 2.

3.2.2. The interaction model

The second step in the analysis phase corresponds to the identification and documentation of the protocols associated to each role. The definition of a protocol consists of the following attributes [43]:

- *Purpose.* Brief description of the nature of the interaction.
- *Initiator.* Role(s) responsible for starting the interaction.
- *Responder.* Role(s) that interact(s) with the initiator.
- *Inputs.* Information used by the initiator during protocol execution.
- *Outputs.* Information provided to the responder during the course of the interactions.
- *Processing.* Brief textual description of any processing that the initiator performs during the interaction.

In order to facilitate the understanding of protocol documentation, we focus our attention on all the interactions of the *Coordinator* role, expressed as the five protocols: *Search, Extraction, Representation, Recommendation* and *Dissemination* (Figs 3 and 4).

Figure 3a shows the protocol *Search*, which describes the interactions among the roles *Coordinator, Searcher* and *Librarian* in order to search and download new web pages from the digital libraries. First, the *Coordinator* (initiator) interacts with the *Searcher* (receiver) in order to obtain new web pages. The *Co-*

Departments	Roles	Informal Description
Information Retrieval	Searcher	Searches for new information in digital libraries and download this information into the system
	Extractor	Extracts relevant information from HTML documents such as: author (s), title, abstract, year, volume, number, etc.
	Representative	Makes a representation of the abstracts in order to generate an inverted list.
Selective Dissemination of Information	Recommender	Establishes optimum relationships between recuperated and qualified documents with user needs
	Disseminator	Sends recommended documents by e-mail.
User Management	User Assistant	Receives the actions associated with each user.
	User	Person that requests the service
	User Manager	Person that creates the extraction parameters
Mail Management	Mail Manager	Stores the recommended documents
DL Management	Librarian	Stores information in their digital library(ies).
Central Coordinator	Coordinator	In charge of presenting the first two services in an automated manner.

Fig. 2. Roles of the System.

ordinator initiates this interaction with a waiting time as input, which represents the maximum time allowed before beginning the next request to the *Searcher*, and with the URLs (Uniform Resource Locator) of the latest downloaded web pages as output. Then, the *Searcher* requests a connection with the *Librarian*, with the last found URL as input, in order to connect this URL and download newly referenced pages, which are sent by the *Librarian*. Finally, the *Searcher* informs the *Coordinator* that the search process has finalized and updates, if needed, the latest URL data as well as the recuperated web pages.

Protocol *Extraction* is shown in Fig. 3(b). It describes how the interaction between the *Coordinator* and the *Extractor* enables the extraction of relevant information from the downloaded web pages. First, the *Coordinator* makes the corresponding request to the *Extractor*. The *Extractor* uses the wrapper-generating tool Minerva [12] to extract information from the web pages of a digital library. To this effect, previously generated production rules are used, which describe the frontiers of the relevant information of the web pages. Finally, the *Extractor* sends the extracted information to the *Coordinator*.

Next, protocol *Representation* shows the interaction between the *Coordinator* and the *Representative*, so that the *Representative* may represent the extracted information in a way that is useful for the application of retrieval information techniques [4] (see Fig. 3c).

In the *Recommendation* protocol (Fig. 4a), the *Coordinator* requests to the *Recommender* the application of content-based and collaborative filtering techniques to the documents. On one hand, the content-based techniques evaluates the similarities between the document representation and the user profile [4]. On the other hand, the collaborative technique evaluates the similarities between the users' preferences based on the votes given by the users to the documents. These similarities are calculated, for example, using the Pearson's correlation formula [36]. The combination of these two techniques allows one to obtain more exact results [6, 14,22].

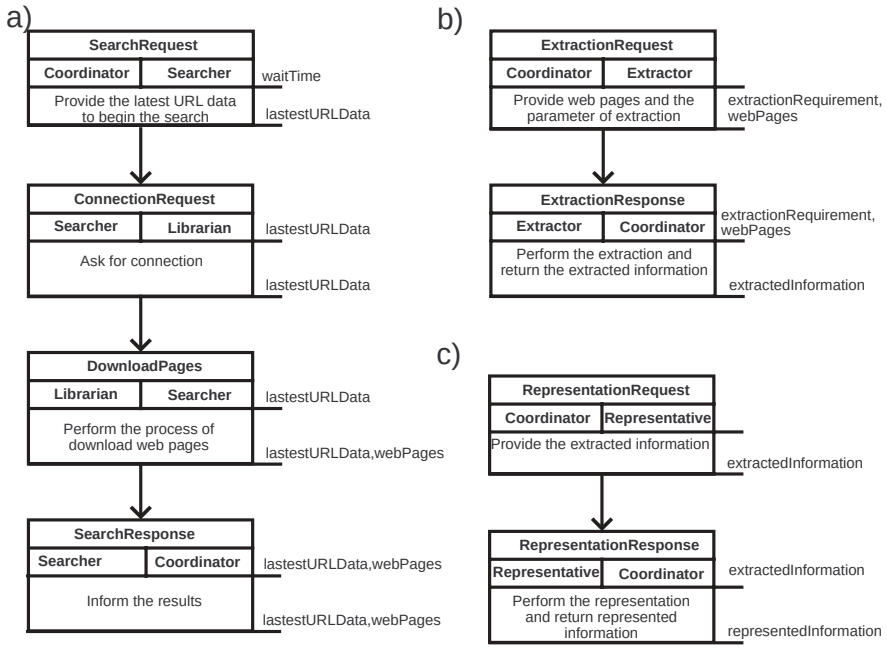

Fig. 3. Definition of protocols associated with the Coordinator role: (a) Search, (b)Extraction, and (c) Representation.

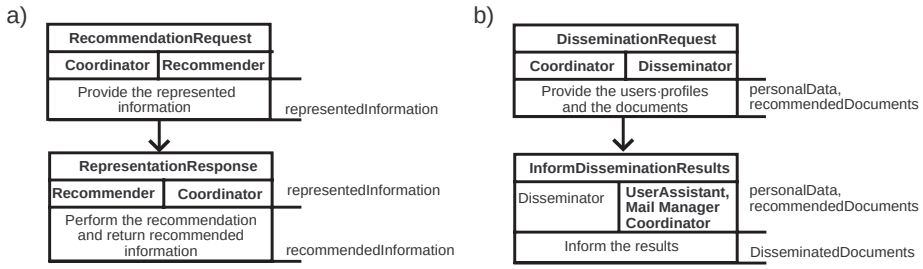

Fig. 4. Definition of protocols associated with the Coordinator role: (a) Recommendation, and (b) Dissemination.

Finally Fig. 4(b) shows protocol *Dissemination*, where the *Coordinator* interacts with the *Disseminator* in order to send the recommended documents to the users via e-mail.

3.2.3. The roles model

The purpose of this stage is to generate an elaborated roles model, representing each role by a textual template that documents its permissions (resources that can be used), responsibilities (functions), protocols (interaction patterns), and activities (private actions in which participate).

Figure 5 shows the textual template of the *Recommender* role. In our system we have identified three permissions associated to the *Recommender* role: *reads*, *generates* and *updates*. The permission *reads* concerns with the information about the user's profile to find the documents according to his/her interests. The responsibilities of a role can be divided into two categories, liveness and safety properties, as we mentioned earlier. In Gaia we specify the liveness properties with regular expressions, which defines the life-cycle of the role (e.g., *ROLENAME* = expression). To distinguish between activities and protocols in the expression, we show an activity underlined. In the case of the *Recommender* role, it participates in two protocols (*RecommendationRequest* and *RecommendationResponse*) and carry out five activities. Safety requirements in Gaia are specified by means of a list of predicates. The safety expressions are listed as a bulleted list, in which each item represents an individual safety responsibility (e.g., to accomplish the permission *reads* the user's profile has to be valid – see expression *ValidProfile* = *True*).

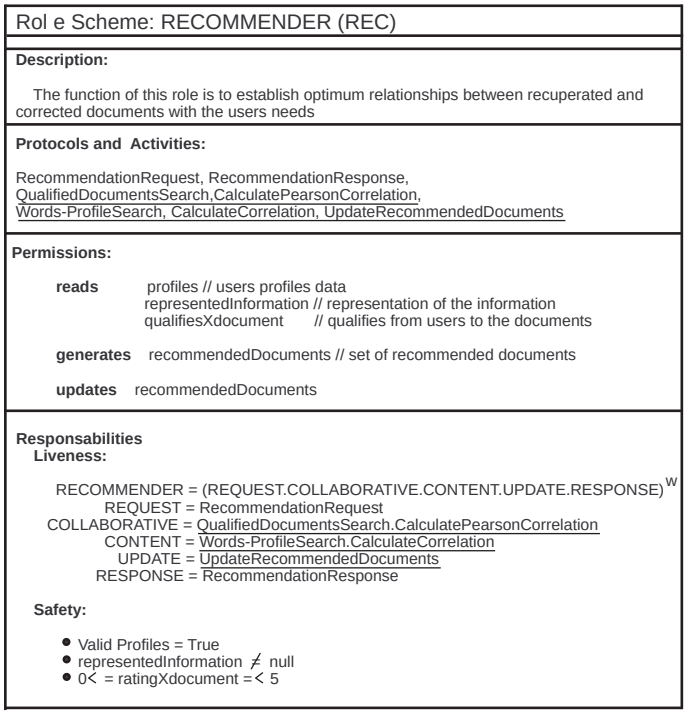

Fig. 5. Schema for role *Recommender*.

3.3. Design phase

The aim of the Gaia design phase is to transform the analysis models into a sufficiently low level of abstraction that traditional design techniques may be applied in order to implement the agents [43]. The design process involves three models. The *agent model* identifies the *agent types* that will make up the system, and the *agent instances* that will be instantiated from these types. The *service model* identifies the principal services that are required to accomplish the roles of the agent. Finally, the *acquaintance model* documents the communication links between the different agents.

3.3.1. The agent model

The *agent model* aims to document the different types of agents that will be used in the system, and the instances that will realize these types of agents at run-time. It is defined by using a simple *agent type tree*, where the root nodes are the agent types and the children are the roles associated with this type of agent.

Figure 6 shows the agents' types and instances that comprise our prototype. The prototype consists of six types of agents, where the *Library Agent* and the *Mail Agent*, which represents the digital libraries and a mail server, are external to the prototype.

The agent instances that will appear in the system are documented by annotating the agent types in the *agent model*. An annotation n means there will be n instances in the run-time system. An annotation $m \cdots n$ indicates there will be between m and n instances in the run-time system ($m < n$). An annotation $*$ means there will be zero or more instances at run-time, and $+$ means there will be one or more instances at run-time. For instance, Fig. 6 shows that there will be one or more instances of *User Agent*.

3.3.2. The services model

The *service model* indicates the services associated with each agent role, and specifies the main properties of these services.

Figure 7 illustrates the services associated with the *Coordinator Agent*. The *Start Search* service, defined from the *Search* protocol, is invoked when the *WaitTime* has been completed, returning the latest URL associated to the new publications and the content of its related web pages. The *Start Extraction* service, defined from the *Extraction* protocol, requests an extraction service to the *Extractor*, obtaining the relevant information from the new publications. The *Perform Representation* service, defined from the *Representation* protocol, takes as input the extracted information and re-

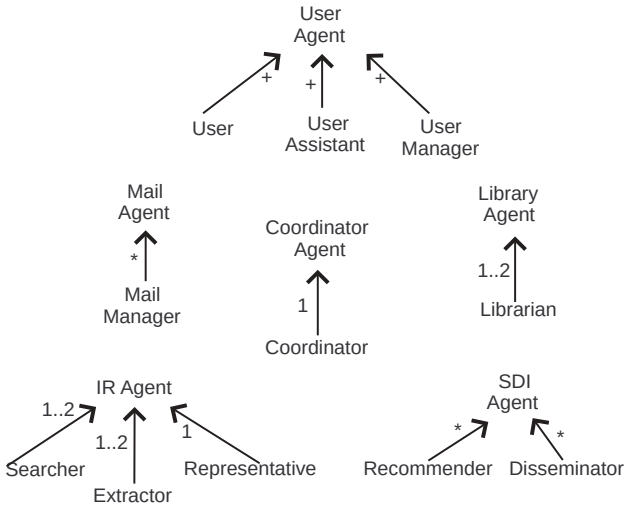

Fig. 6. The Agent Model.

SERVICE	INPUT	OUTPUT	PRE-CONDITION	POST-CONDITION
Start Search	latestURLData	LatestURLData, webPages	(waitTime=30 days) and (latestURLData <> null)	(latestURLData<>null) and (webPages=updated)
Start Extraction	webPages, extraction Requirements	extracted Information	(webPages <> null) and (extractionReqirements <> null) and (compatible(webPages, extractionRequirements)=true)	extractedInformation <>incomplete
Perform Representation	extracted Information	represented Information	extractedInformation <>incomplete	representedInformation <>null
Find Recommendation	profiles, representedInformation, qualifyXDocument	recommended Information	(validProfile=true) and (representedInformation <> null) and (0<= ratingXdocument <= 5)	recommendedInformation <>null
Make Dissemination	profiles, recommended Documents	e-mail, disseminated Documents	(e-mail <> empty) and (Channel = withConnection) and (personalData= valid)	Mail Delivery Receipt

Fig. 7. The Services Model.

quest for representing the information. The *Find Recommendation* service, defined from the *Recommendation* protocol, takes the user profiles, the represented information and the qualification matrix (a matrix where each input represents the qualification given by user i to document j) as input; having as output a list with the recommended documents. Finally, service *Make Dissemination*, defined from the *Dissemination* protocol, uses *PersonalData* (name, last name, user's e-mail) and the recommended documents as input, in order to send e-mails containing the information of the new documents.

3.3.3. The acquaintance model

The final model of a Gaia design in the *acquaintance model*. This model simply defines the communication links that exist between the agent types. Figure 8 shows the interactions between the different types of agents that make up the Selective Dissemination Information prototype.

4. Towards a more concrete model with AUML

As mentioned previously, the Gaia outcome is an abstract design to be refined by traditional design techniques. We have found useful to further refine Gaia with AUML, since it enables one to apply the typical object-oriented properties of UML and new features for specifying other aspects of the agent interaction that are not covered by Gaia.

The Unified Modelling Language (UML) is gaining wide acceptance for the representation of engi-

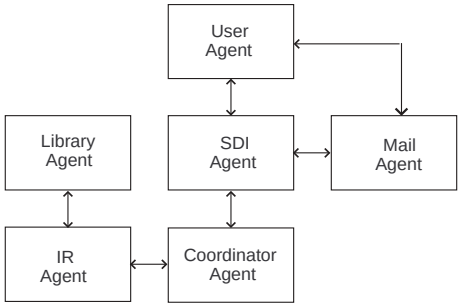

Fig. 8. The Acquaintance Model.

neering artifacts in object-oriented software [38]. The current UML is sometimes insufficient for modelling agents and agent-based systems. However, no formalism yet exist to sufficiently specify agent-system development [31]. Our view of agents, as the next step beyond objects, leads us to explore extensions to UML and idioms within UML to accommodate the distinctive of agents. The result is Agent UML (AUML) [9, 31].

Both FIPA and the OMG Agent Work Group are exploring and recommending extensions to UML such as: specification of Agent Interaction Protocols (AIP), richer role specification, package extension, and deployment diagram extension.

This section describe the first recommended extension called the *Agent Interaction Protocol* (AIP). The AIP describes a communication pattern as an allowed sequence of messages between agents and the constraints on the content of those messages [32].

The first level of the AIP defines communication patterns by means of aggregation concepts such as *packages*. For instance, Fig. 9 describes two packages.

The first package is *DisseminateInformation*, which expresses a simple protocol between the *Coordinator Agent* and the *SDI Agent*. Here, the *Coordinator Agent* sends a *DisseminationRequest*, that the *SDI Agent* responds with a *DisseminationResponse*. The second package is *SendInformation*, which is carried out before sending the *DisseminationResponse* in the first package. The *SDI Agent* sends a *ConnectionRequest* message to the *Mail Agent*, which decides whether to accept or reject the request. If the *Mail Agent* accepts the request, it will receive an e-mail message from *SDI Agent*.

The second level of the AIP represents of interaction among agents. It allows one to integrate information from the Gaia *interaction* and *acquaintance models*. Figure 10 shows the complete interaction sequence of the *Coordinator Agent*.

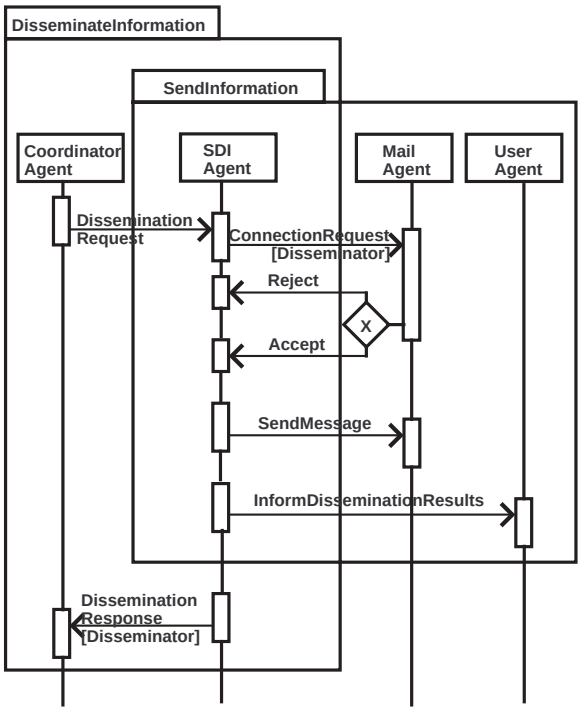

Fig. 9. Using packages to express the*Dissemination* Protocol.

Finally, the third level models the internal process of an agent. We may use *activity diagrams* or *statecharts* to represent such internal process. For instance, Fig. 11 shows the process of information retrieval in the *IR Agent*, carried out by the *Representative* role.

5. Implementation

The implementation and test phases are not part of the Gaia methodology, since they depend on the environment in which the agents will be available and the employed platform of agent development.

The first step in the implementation consisted in defining the system architecture. Since our agent-based system will be available on the web, the selected architecture was the three-layer structure [35]. The three layer architecture consists of a presentation layer, a processing layer (application server) and a database layer. Figure 12 shows the system architecture. The presentation layer is responsible for the visual presentation of the application, as well as the interaction between the user and the system. The database layer contains the data of the application (profiles and documents). Finally, the processing layer describes the agent cooperation in order to achieve a common goal.

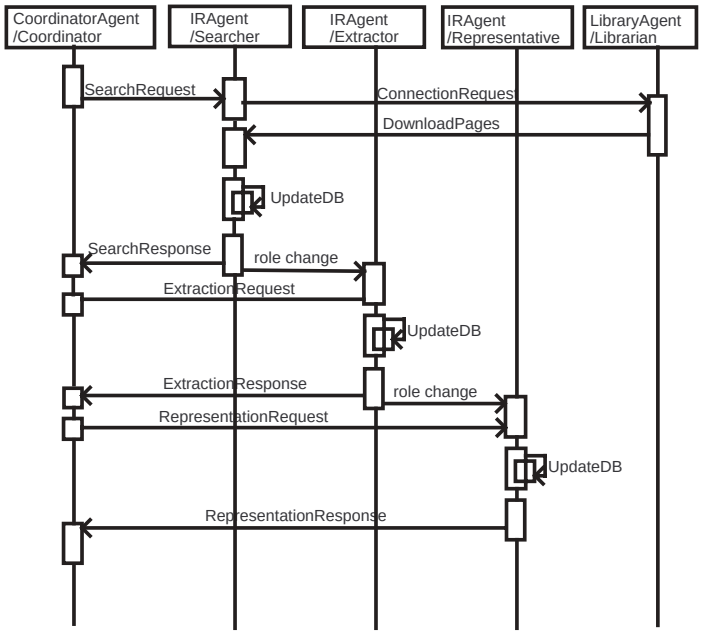

Fig. 10. A sequence diagram depicting an interaction among agents of the System.

The second step in the implementation consisted in the development of the user interface, based on the *web*-flow diagram generated, as an abstraction, from the acquaintance model, as illustrated in Fig. 14. Each interface element activates a process developed by either an agent, a specific class or procedure. The user has access to the prototype through a series of screens, where he/she may administer his/her personal data or profile. The user is also able to see the documents recommended by the agents, as well as to evaluate those documents. In the same way, if the user does not have the possibility to access the system, he/she can easily review the recommended documents through his/her personal mail service (see Fig. 13).

In the third step, we define the parameters to extract the information contained in the web pages from the digital libraries. We use the Araneus Wrapper ToolKit (AWTK version 1.0): Minerva + Editor [12]. Minerva is a formalism to define grammars, in the EBNF style, enriched with an explicit exception-handling mechanism.

The last step consisted in the programming of agents *CoordinatorAgent*, *IRAgents* and *SDIAgents* using Java. Each agent corresponds to a class, and the Java message-passing facility is used for implementing the agent communication.

The current version of the system is used by members of the Information Technology Group at Universidad Autónoma de Bucaramanga. We are working on a new version of the system, using the AgentBuilder platform [40], which has been expanded to include other wrapper-generating tools such as DEByE [23] and we are studying new algorithms to collaborative filtering [18] and new ways to integrate two techniques of information filtering [7,11,33,34].

It is worth noting that having such as detailed design (Gaia + AUML) has been very helpful for the purpose of software maintenance and expandability.

6. Conclusions

The agent-based approach to system development offers a natural means of conceptualizing, designing and building distributed systems. The successful practice of this approach requires robust methodologies for agent-oriented software engineering. This article applies Gaia, a methodology for agent-based system development founded on the view of a multi-agent system as a computational organisation, and AUML, a UML extension for agents, to the problem of Selective Dissemination of Information on the Web.

We have developed each of the models included in Gaia, obtaining a design that identifies the agents of the system and their instances (the agent model), the services associated to each agent (the service model), and the communication links between the agents (the acquaintance model). Further, we have applied the Agent

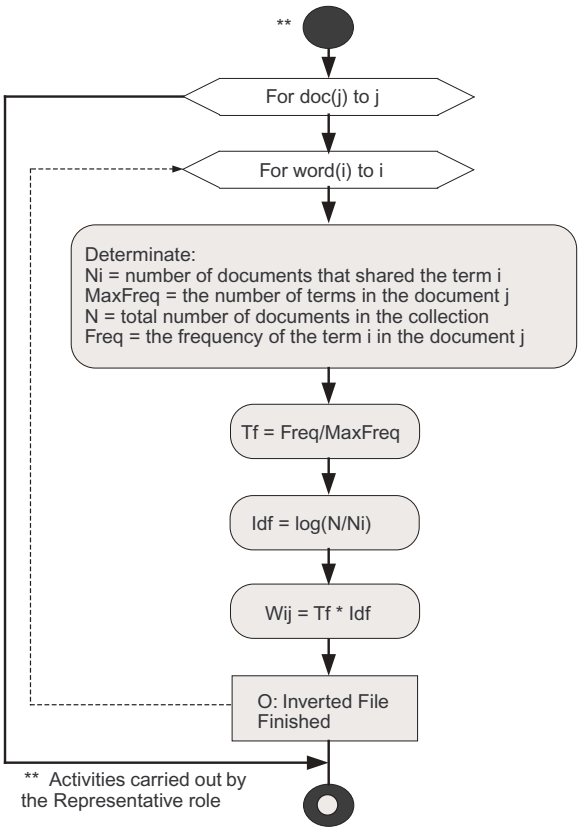

Fig. 11. Internal processing for *IR Agent*.

Interaction Protocol (AIP) of AUML to the Gaia models. The acquaintance model is refined in the first two layers of the AIP, in order to get a deeper view on the interaction among agents as well as their communication protocols. Finally, the service model is refined in the last layer of the AIP, resulting in a detailed description of the internal process of each agent.

Gaia has been successfully applied to diverse domains such as e-commerce [20] and telecommunication [16]; in the same way, AUML has been also applied to the same domains [32]. However, as far as we know, no work has been done, neither on applying Gaia or AUML to the Selective Dissemination of Information problem, nor on integrating both techniques.

The Gaia methodology suggests the application of object-oriented techniques to the generated design in order to make it more concrete. However, we have found that some agent-oriented properties of the interaction and the acquaintance models could be improved, and that such improvements are not possible using traditional object-oriented techniques. This has motivated the integration with AUML. Figure 15 shows the relation between the Gaia models and the layers of the

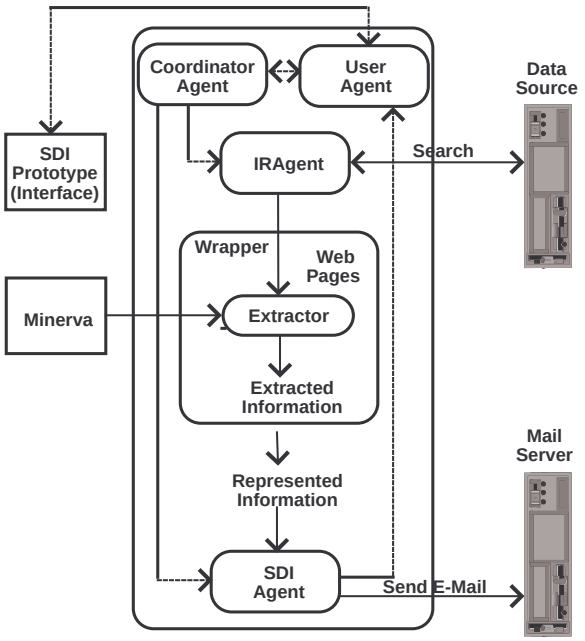

Fig. 12. System Architecture.

AUML's AIP. As future work, we plan to include other AUML extensions into our approach in order to represent internal and external behaviours of the multiagent systems in more detail. These models will support the developers in the implementation phase [8,42].

Another novelty of our system is the combination of filtering techniques (content-based and collaborative) with wrapper-generating tools. As a result, we obtain flexibility and scalability in the SDI on the Web. By contrast, systems like Amalthaea [30] combine filtering techniques with hand-coded wrappers.

It is worth noting that systems such as the one described here constitute a powerful tool for many organisations. They allow the automated process of search and dissemination of information, offering the user the possibility of being notified automatically of new topics of interest. Intended work includes studying the design and implementation of more general tools for dissemination of information.

The work presented here is part of an ongoing project aimed at comparing and extending different methodologies for multiagent systems development, such as MAS-CommonKADS [19] and MASSIVE [26].

MAS-CommonKADs is an agent-based methodology founded on the well-known standard of CommonKADS [17]. However, it has the drawback that some properties feel unnatural for agent development. For instance, the communication model in CommonKADS specifies the communication between

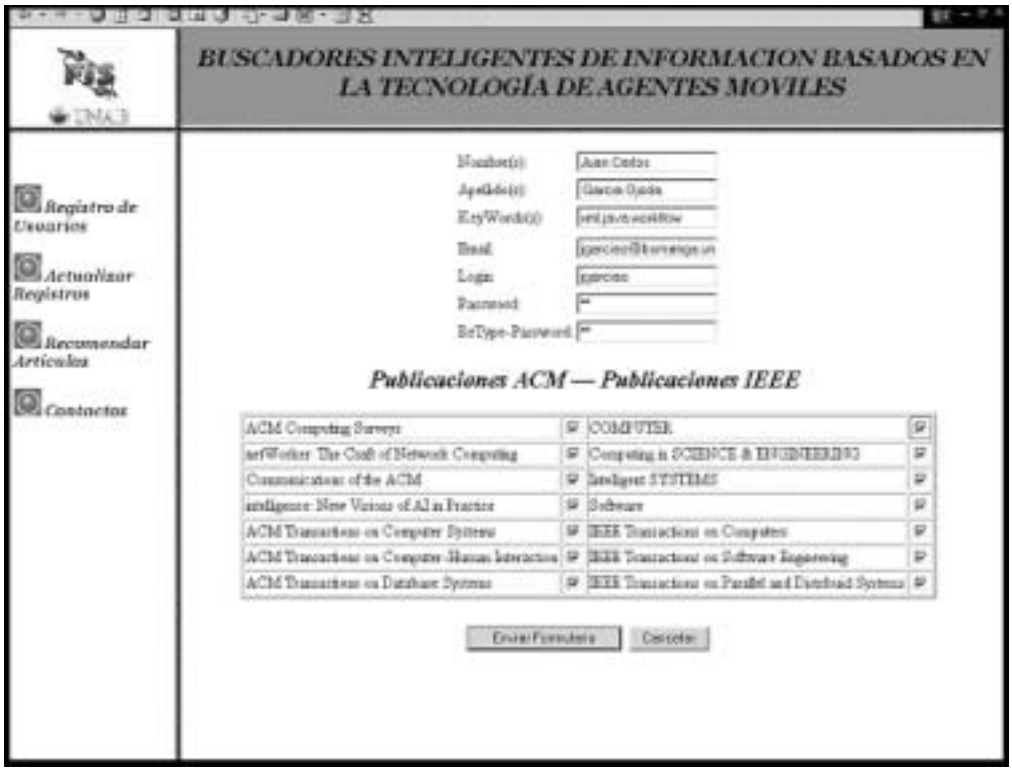

Fig. 13. Interface of the System.

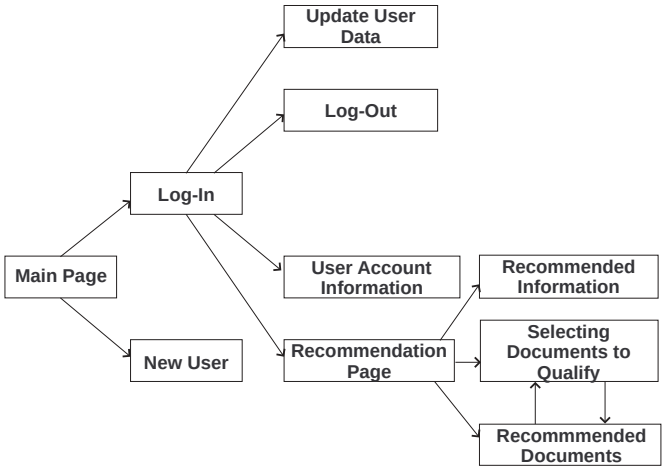

Fig. 14. *Web*-flow diagram of the system.

knowledge systems and human; however, the communication model in MAS-CommonKADS specifies the interaction between human and agents. By contrast, Gaia is not based on any existing methodology. The developers of Gaia have to get the credit for developing a complete new methodology, especially designed for agent-based systems.

MASSIVE follows a view-oriented approach for agent-based development by dealing with the system as a whole and using different views on the system as its main abstraction. Its author claims that it avoids the integration process of model-oriented methodologies – such as Gaia –, since the system is always consistent from any view: changes in one view are always prop-

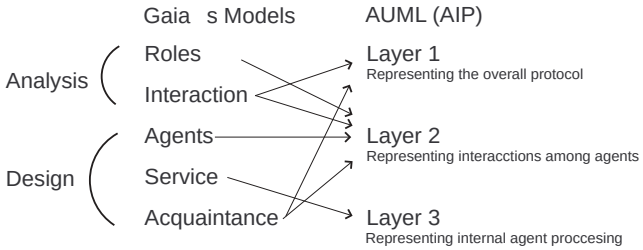

Fig. 15. Relation between the Gaia models and the layers of AUML's AIP.

agated to other views. This offers an interesting perspective for software development, but we have found that its application is not very intuitive.

Acknowledgments

This work was partially supported by the Colombian Research Council, Instituto Colombiano para el Desarrollo de la Ciencia y la Tecnología – Colciencias under Grant 1241-14-11080.

References

[1] K. Aas, *A survey on personalized information filtering systems for the world wide web*, Technical Report 922, Norwegian Computing Center, 1997.

[2] S. Abiteboul, P. Buneman and D. Suciu, *Data on the Web*, Morgan Kaufmann, 2000.

[3] B. Adelberg, NoDoSe: A tool for semi-automatically extracting estructured and semi-estructured data from text documents, *SIGMOD Record* **27**(2) (1998), 238–294.

[4] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, Addison-Wesley, 1999.

[5] M. Balabanovic and Y. Shoham, Learning information retrieval agents: Experiments with automated web browsing, in *Proceedings of the AAAI'95 Spring Symposium on Information Gathering from Heterogenous, Distributed Resources*, Stanford University, 1995, pp. 13–18.

[6] M. Balabanovic and Y. Shoham, Fab: Content-based, collaborative recommendation, *Communications of the ACM* **40**(3) (1997), 66–70.

[7] P. Baudisch, Joining collaborative and content-based filtering, in *Interacting with Recommender Systems, CHI 99 Workshop*, Pittsburgh, PA, 1999.

[8] B. Bauer, Uml classes diagrams and agent-based systems, in *Proceedings of the fifth international conference on Autonomous agents*, ACM Press, 2001, pp. 104–105.

[9] B. Bauer, J. Muller and J. Odell, Agent uml: A formalism for specifying multiagent interaction, in: *Agent-Oriented Software Enginnering*, P. Ciancarinni and M. Wooldridge, eds, Springer, 2001, pp. 91–103.

[10] K. Bollacker, S. Lawrence and C. L. Giles, CiteSeer: An autonomous web agent for automatic retrieval and identification of interesting publications, in: *Proceedings of the Second International Conference on Autonomous Agents*, K.P. Sycara and M. Wooldridge, eds, ACM Press, 1998, pp. 116–123.

[11] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes and M. Sartin, Combining content-based and collaborative filters in an online newspaper, in *Proceedings of ACM SIGIR Workshop on Recommender Systems*, 1999.

[12] V. Crescenzi and G. Mecca, Grammars have exceptions, *Information Systems* **23**(8) (1998), 539–565.

[13] V. Crescenzi, G. Mecca and P. Merialdo, Roadrunner: Towards automatic data extraction from large web sites, in *Proceedings of 27th International Conference on Very Large Data Bases*, Morgan Kaufmann, 2001, pp. 109–118.

[14] O. de Vel and S.A. Nesbitt, Collaborative filtering agent system for dynamic virtual communities on the web, in *Working Notes of Learning from Text and the Web, Conference on Automated Learning and Discovery* (*CONALD-98*), Carnegie Mellon University, 1998.

[15] J. Delgado, N. Ishii and T. Ura, Intelligent collaborative information retrieval, in: *6th Ibero-American Conference on Progress in Artificial Intelligence* (*IBERAMIA-98*), (Vol. 1484), H. Coelho, ed., Springer, 1998, pp. 170–182.

[16] N. dos Santos, F.M. Varej ao and O. Tavares, Multi-agent systems and network management – A positive experience on Unix environments, in: *Advances in Artificial Intelligence* (*IBERAMIA 2002*), (Vol. 2527), F.J. Garijo, J.C. Riquelme and M. Toro, eds, Springer, 2002, pp. 616–624.

[17] G. Schreiber et al., *Knowledge Engineering and Management: The CommonKADS Methodology*, MIT Press, 2000.

[18] C.N. Gonzáles-Caro, M.L. Calderón-Benavides, J. de J. Pérez-Alcázar, J.C. García Díaz and J. Delgado, Towards a more comprehensive comparison of collaborative filtering algorithms, in *Proceedings of the 9th International Symposium on String Processing and Information Retrieval* (*SPIRE'02*), Lecture Notes in Computer Science, Springer, 2002, pp. 248–253.

[19] C.A. Iglesias, M. Garijo, J.C. González and J.R. Velasco, Analysis and design of multiagent systems using MAS-CommonKADS, in: *4th International Workshop on Agent Theories, Architectures, and Languages* (*ATAL-97*), (Vol. 1365), M.P. Singh, A. Rao and M.J. Wooldridge, eds, Springer, 1998, pp. 313–328.

[20] N.R. Jennings, P. Faratin, M.J. Johnson, T.J. Norman, P. O'Brien and M.E. Wiegand, Agent-based bussines process management, *International Journal of Cooperative Information Systems* **5**(2–3) (1996), 105–130.

[21] N.R. Jennings and M. Wooldridge, *Handbook of Agent Technology*, chapter Agent-oriented software engineering. AAAI/MIT Press, 2000.

[22] B. Krulwich, Lifestyle finder, *AI Magazine* **18**(2) (1997), 37–46.

[23] A.H.F. Laender, B. Ribeiro-Neto and A.S. Da Silva, Debye – data extraction by example, *Data and Knowledge Engineering* **40**(2) (2002), 121–154.

[24] A.H.F. Laender, B. Ribeiro-Neto, A.S. Da Silva and J.S. Teixeira, A brief survey of web data extraction tools, *SIGMOD Record* **2**(31) (2002), 84–93.
[25] H. Lieberman, Letizia: An agent that assists web browsing, in: *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, C.S. Mellish, ed., Morgan Kaufmann, 1995, pp. 924–929.
[26] J. Lind, *Iterative software engineering for multiagent systems: The MASSIVE method*, Lecture Notes in Computer Science, Springer, 1994.
[27] P. Maes, Agents that reduce work and information overload, *Communications of the ACM* **37**(7) (1994), 31–40.
[28] G. Mecca, P. Atzeni, A. Masci, P. Merialdo and G. Sindoni, The araneus web-based management system, in *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data*, ACM Press, 1998, pp. 544–546.
[29] D. Mladenic, Text-learning and related intelligent agents: A survey, *IEEE Intelligent Systems* **14**(4) (1999), 44–54.
[30] A. Moukas and P. Maes, Amalthea: An envolving multiagent information filtering and discovery for the WWW, in *Autonomous Agents and Multiagent Systems*, Kluwer Academics, 1998, pp. 59–88.
[31] J. Odell, V.D. Parunak and B. Bauer, Extending uml for agents, in: *Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence*, G. Wagner, Y. Lesperance and E. Yu, eds, 2000, pp. 3–17.
[32] J. Odell, V.D. Parunak and B. Bauer, Representing agent interactions protocols in uml, in: *Agent-Oriented Software Engineering*, P. Ciancarini and M.J. Wooldridge, eds, Springer, 2001, pp. 121–140.
[33] P. Melville, R. Mooney and R. Nagarajan, Content-boosted collaborative filtering, in *The Proceedings of the SIGIR-2001 Workshop on Recommender Systems*, New Orleans, LA, 2001.
[34] G. Polcicova, R. Slovak and P. Navrat, Combining content-based and collaborative filtering, in *ADBIS-DASFAA Symposium 2000*, Prague, Czech Republic, 2000, pp. 118–127.
[35] R.S. Pressman, *Software Engineering. A Practitioner's Approach*, (5th ed.), McGraw Hill, 2000.
[36] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm and J. Riedl, Grouplens: An open architecture for collaborative filtering of netnews, in *Proceedings of ACM'94 Conference on Computer Supported Cooperative Work*, ACM Press, 1994, pp. 175–186.
[37] P. Resnick and H.R. Varian, Recommender systems, *Communications of The ACM* **40**(3) (1997), 56–58.
[38] J. Rumbaugh, I. Jacobson and G. Booch, *Unified Modelling Language Reference Manual*, Addison-Wesley, 1998.
[39] I. Stadnyk and R. Kass, Modeling users' interests in information filters, *Communications of the ACM* **35**(12) (1992), 49–50.
[40] Reticular Systems, Agentbuilder: An integrated toolkit for constructing intelligent software agents, available at http://www.agentbulder.com/Documentation/white_paper_r1_3.pdf, February 1999.
[41] L. Terveen, W. Hill, B. Amento, D. McDonald and J. Creter, PHOAKS: A system for sharing recommendations, *Communications of the ACM* **40**(3) (1997), 59–62.
[42] G. Wagner, Towards agent-oriented information systems, Technical report, Freie University of Berlin, Kaiserswerther Str. 16-18, 14195 Berlin, January 2000.
[43] M. Wooldridge, N.R. Jennings and D. Kinny, The gaia methodology for agent-oriented analysis and design, *Autonomous Agents and Multi-Agent Systems* **3**(3) (2000), 285–312.