KNOWLEDGE MANAGEMENT SYSTEM DESIGN USING EXTENDED GAIA

Pooja Jain[1], Deepak Dahiya[2]

[1]Jaypee University of Information Technology, Waknaghat, Solan
pooja.jain@juit.ac.in

[2]Jaypee University of Information Technology, Waknaghat, Solan
deepak.dahiya@juit.ac.in

ABSTRACT

An efficient Learning resource centre can be achieved with the help of a network of collaborating, coordinating and communicating software agents. Agent-oriented techniques represent an exciting new means of analysing, designing and building complex software systems. The designing of the interacting agents is done with the help of Gaia, extended for the multiagent systems. Gaia is a methodology for agent-oriented analysis and design proposed by M. Wooldridge [9].

KEYWORDS

Multiagent system, Gaia, learning resource centre

1. INTRODUCTION

Agent oriented techniques have the potential to significantly improve current practice in software engineering and to extend the range of applications that can feasibly be tackled. Multi agent systems

This paper deals with a multi agent learning resource centre. The designing of the various agents is done through Gaia. The remainder of the paper is structured as follows. Section two talks about the various works done in the field of multiagent systems and Gaia. Section three talks about the current work done. Section four deals with the analysis phase of agent design of the Learning resource centre using Gaia. Section five talks briefly about the architectural design. Section six deals with the detailed design. Section seven talks about the future scope of this paper regarding the implementation of the multi agent Learning resource centre.

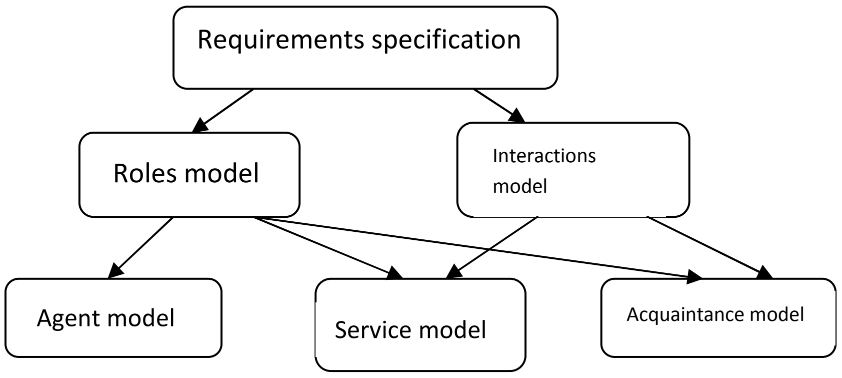
2. RELATED WORK

2.1 Multiagent systems

A multi agent system (MAS) can be abstracted as a set of interacting sub-organizations with a subset of agents possibly belonging to one or more organizations. In each organization, agents can play one or more roles and interact with each other in order to exchange knowledge and coordinate their activities. The MAS is also connected to an environment through some of the agents and these agents interact with the environment through various kinds of sensors and effectors [3].

2.2 Gaia methodology

Gaia [8] was the first complete methodology proposed for the analysis and design of MAS.
Gaia takes the view that a system can be seen as a society or organization of agents. It models both the micro-level (agent structure) and macro level (agent society and organization structure) of agent development. Figure 1 shows the Gaia models. In the analysis phase, the role model and the interaction model are constructed. Roles consist 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 fulfill 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 [1].



However, the original version of Gaia suffered from the limitations of being suitable for the analysis and design of closed MAS and of adopting non-standard notation techniques. Several extensions to the basic Gaia methodology have been recently proposed to overcome these limitations. Various organizational abstractions are necessary for analyzing and designing MASs. There is a need to produce an ordered sequence of steps, an identifiable set of models, and an indication of the interrelationships between the models, showing how and when to exploit which models and abstractions in the development of a MAS.

The new, extended version of Gaia exploits the new organizational abstractions and significantly extends the range of applications to which Gaia can be applied [6]. The Gaia process starts with the analysis phase, whose aim is to collect and organize the specification which is the basis for the design of the computational organization. This includes the identification of:

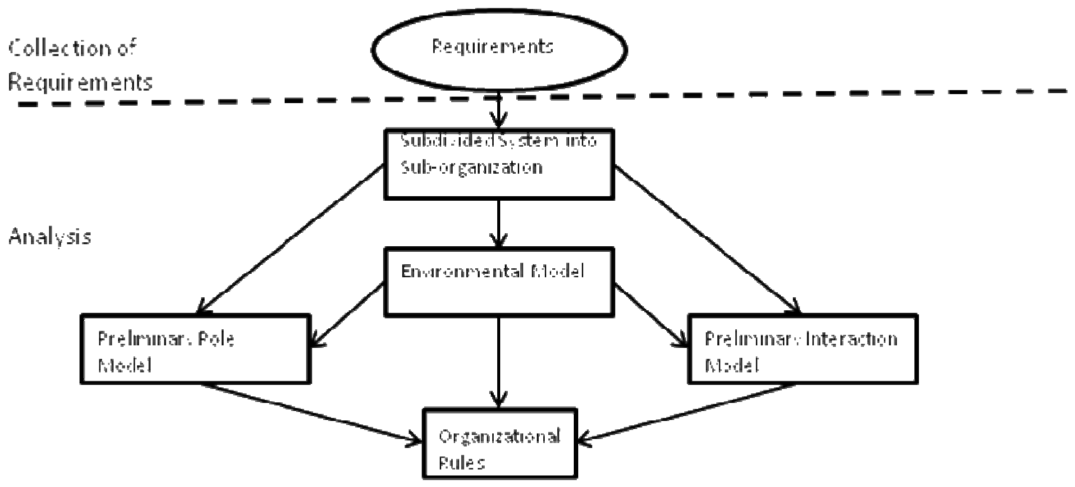
—*The goals of the organizations that constitute the overall system and their expected global behavior*. This involves identifying how to fruitfully decompose the global organization into loosely coupled sub organizations.
—*The environmental model*. Intended as an abstract, computational representation of the environment in which the MAS will be situated.
—*The preliminary roles model*. Identifying the basic skills required by the organization. This preliminary model contains only those roles, possibly not completely defined, that can be identified without committing to the imposition of a specific organizational structure. Also, the notion of roles, at this stage, is abstract from any mapping into agents.
—*The preliminary interaction model*. Identifying the basic interactions required to accomplish the preliminary roles. Again, this model must abstract away from the organizational structure and can be left incomplete.
—*The rules that the organization should respect and enforce in its global behavior*. Such rules express constraints on the execution activities of roles and protocols and are of primary importance in promoting efficiency in design and in identifying how the developing MAS can support openness and self-interested behavior [9].

The output of the analysis phase—consisting of an environmental model, a preliminary roles model, a preliminary interactions model, and a set of organizational rules—is exploited by the design phase, which can be logically decomposed into an architectural design phase and a detailed design phase. The architectural design phase includes:
—*The definition of the system's organizational structure in terms of its topology and control regime*. This activity, which could also exploit of catalogues organizational patterns, involves considering: (i) the organizational efficiency, (ii) the real-world organization (if any) in which the MAS is situated, and (iii) the need to enforce the organizational rules [7].
—*The completion of the preliminary role and interaction models*. This is based upon the adopted organizational structure and involves separating— whenever possible—the organizational-independent aspects (detected from the analysis phase) and the organizational-dependent ones (derived from the adoption of a specific organizational structure). This demarcation promotes a design-for-change perspective by separating the structure of the system (derived from a contingent choice) from its goals (derived from a general characterization).

Once the overall architecture of the system is identified together with its completed roles and interactions model, the detailed design phase can begin. This covers:
—*The definition of the agent model*. This identifies the *agent classes* that will make up the system and the *agent instances* that will be instantiated from these classes. There may be a one-to-one correspondence between roles and agent types, although a number of closely related roles can be mapped into in the same agent class for the purposes of convenience or efficiency.
—*The definition of the services model*. This identifies the main services—intended as coherent blocks of activity in which agents will engage—that are required to realize the agent's roles, and their properties.

3. CURRENT WORK: AGENT DESIGN FOR LEARNING RESOURCE CENTRE

The library system that's being considered here is a college library. The users are the students and the faculty of the college. The Learning resource centre (LRC) is a MAS comprising of various independent, reactive and autonomous agents. The agents coordinate, collaborate and coordinate with each other to form an efficient LRC. In MASs, applications are designed and developed in terms of autonomous software entities (*agents*) that can flexibly achieve their objectives by interacting with one another in terms of high-level protocols and languages [10]. These characteristics are well suited to tackling the emerging complexities for a number of reasons. Such a MAS can be designed using Gaia, as described in the previous section.


Collection of Requirements
Requirements
Subdivided System into Sub-organization
Analysis
Environmental Model
Preliminary Role Model
Preliminary Interaction Model
Organizational Rules

This paper will concentrate only on the analysis phase as depicted in the above figure. The figure is inspired from the paper by Wooldridge [9].

A library has a large number of books related to different fields and topics. The user can search for the books as per his desire. Depending upon his search, a list of books is displayed. The list will have the attributes like, title of the book, author/authors, publishing house and the number of available copies. When the user wants to have a particular book, the system will check the number of books already issued to the user, as there is a limit on the number of books that a user can be issued. If the user has the sufficient balance, then the book will be given to the user, and the number of copies of that particular book will be reduced by one.

If the user asks for a book which is present in the library, but currently none of the copy is available, then the user can see the people to whom the copies have been issued. The system will give an option, whether the user wants to send an email to them, to return the book. Depending upon the choice of the user, an email can be send to some/all of the people having the copies of the desired book.

This LRC can be considered as an intelligent system, due to its one unique feature. If suppose, the user searched for a topic and that topic is not available in any of the books present in the library, then the user will be given a choice by the system for the web search. The web search is done by a topical web crawler. A web crawler is a program or an automated script which browses the World Wide Web in a methodical automated manner [5]. A focused crawler or topical crawler is a web crawler that attempts to download only web pages that are relevant to a topic or set of topics. Topical crawling generally assumes that only the topic is given, while focused crawling also assumes that some labeled examples of relevant and not relevant pages are available. Two major components of a crawler have been considered-collecting agent, and searching agent [11]. The collecting agent downloads web pages from the WWW and indexes

the HTML documents and storing the information to a database, which can be used for later search. Collecting agent includes a simple HTML parser, which can read any HTML file and fetch useful information, such as title, pure text contents without HTML tag, and sub-link. The searching agent **- s**earching agent is responsible for accepting the search request from user, searching the database and presenting the search results to user. When the user initiates a new search, database will be searched for any matching results, and the result is displayed to the user, it never searches over WWW but it searches the database only. A high level architecture of a web crawler [2] has been analyzed as in figure 1 for building web crawler system on the client machine. Here, the multi-threaded downloader downloads the web pages from the WWW, and using some parsers the web pages are decomposed into URLs, contents, title etc. The URLs are queued and sent to the downloader using some scheduling algorithm. The downloaded data are stored in a database.
When the user comes to return the book, then the system accepts the book and updates the number of copies of that particular book. The books can be recommended by the faculty as well. When a recommendation request reaches the LRC, it prepares a quote and sends it to the accounts department, so that the book can be ordered.

4. ANALYSIS PHASE

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

4.1 The Organizations

The first phase in Gaia analysis is concerned with determining whether multiple organizations have to coexist in the system and become autonomous interacting MASs. In our system, the organizations can be easily identified as:-

- The one that takes care of issue and return of books
- The one that keeps all the information about the available books

4.2 The Environmental Model

The environment is treated in terms of abstract computation resources, such as variables or tuples, made available to the agents for sensing. Following such identification, the environmental model (in its simplest form) can be viewed as a list of resources; each associated with a symbolic name, characterized by the type of actions that the agents can perform on it, and possibly associated with additional textual comments and descriptions.

reads	*book_catalogue*	the collection of all the books can be read
	copies	the no of copies of the book available can be read
changes	*book_catalogue*	changes whenever a new book is added or deleted
	copies	the no of copies of the book will be changed by the agent whenever the book is issued or returned

reads	*recommended_books*	the list of books recommended by the faculty
reads	*book_balance*	the total of the number of books issues to a user can be read
changes	*reg_details*	register the students and the faculty

The book_catalogue is represented by a data structure including information such as author of the book; title of the book, the year in which the book was published, the publishing house etc. Copies is the total number of copies of a book available in the library

4.3 The Preliminary Role Model

Given the identification of the basic skills and of their basic interaction needs, respectively, the analysis phase can provide a *preliminary definition* of the organization's roles and protocols.
To represent (preliminary) roles, Gaia adopts an abstract, semiformal, description to express their capabilities and expected behaviors. These are represented by two main attribute classes, respectively: (i) permissions and (ii) responsibilities.
In the Learning resource centre, the roles can be identified as:-

1. Book_manager
2. Register
3. Authorization
4. Solve_query
5. Display
6. Issuer
7. Web_crawler
8. Recommend
9. Return
10. Send_email

Book_manager role keeps track of all the books available in the library. It also keeps all the information about the book_catalogue and the copies of a particular book present. It will also keep the entire information about a particular user, like the books issued in his name and his transaction history of 3 months

The register role is responsible to register the new students and new faculty. The user will give his enrollment number, name and mother's name and accordingly it will allot a default user id and a password. When the user logs to the system, it will give an option of exiting user or a new user. If the user is new then the registration process is handled by this agent otherwise the request is send to the authorization agent.

The authorization role is used to get the username and the password from the user. The details are matched with the database and correspondingly authorization is done. Once the authorization is done, a message is passed to the solve_query agent.

The solve_query agent takes the query from the user and solves it. It will send a message to the display role, which will display all the books/journals/magazines matching the search results. If the display agent is unable to display anything, or in other words, the book/topic searched by the user is not available in the library, then the solve_query role will give the option to the user for searching the topic from the web. If the user agrees, a message is sent to the web_crawler role, which will take care of the topic to be searched and store the results in a folder. This folder is private for the user and he can see the downloaded material at later point of time. This data is available only for a month. The data of this folder is also displayed by this very display role.

If the display agent is able to display the books and its details in coordination with the book_manager agent, then the user can select any book which he wants. Once the user has selected a particular book, the Issuer role checks the book balance of the user and accordingly issues the book.

The display role is used to display the results to the user. These results can be-

1. Books/ Journals/ magazines search result
2. Search results stored by the web crawler

Once a book is issued by the issuer agent, a mail is send to the user through the send_email role confirming the issue of the book. It will also show the list of all the books currently issued to the user, collaborating with the book_manager role.

The recommend role keeps a track of recommended books. The books can be recommended by the faculty or a request can be made by the student, as well, for the purchase of a book. Then this role will form a quotation of the book and send an email to the accounts department through the send_email agent.

Return role concerns the returning of the books by the users. As the user returns the book, the "copies" attribute is incremented by one. Once the book is returned by the user, a mail is send to the user through the send_email agent confirming the return of the book. It will also show the list of all the books currently issued to the user, collaborating with the book_manager role.

The permissions of the book_manager can be depicted as:

Reads	*book_catalogue*	can read all the information about the various books available
Changes	*book_catalogue*	when any book is added or deleted
Reads	*user_book_details*	list of all the books currently issued to the user

The permissions of the Issuer can be depicted as:

Changes	*copies*	When the book is issued, its number of copies is decremented by one
Reads	*user_book_balance*	Checks the balance of the user, so that the book can be issued if he has sufficient balance

The permissions of the Display agent can be depicted as:

Reads	*book_catalogue*	Displays the information regarding the books
Reads	*query_result*	Displays the results of the queries

The permissions of the recommend agent is

Read	*recommended_books*	the list of recommended books
Changes	*book_quote*	a quotation of the recommended book is formed

4.4 Responsibilities

These attributes determine the expected behavior of a role and, as such, are perhaps the key attribute associated with a role. Responsibilities are divided into two types: liveness properties and safety properties.
Liveness properties intuitively state that “something good happens,” that is, describe those states of affairs that an agent must bring about, given certain conditions. In contrast, safety properties are *invariants*. Intuitively, a safety property states that “nothing bad happens,” that is, that an acceptable state of affairs is maintained. In other words, safety expressions indirectly express abnormal situations to which an agent should be able to react.

Liveness expressions of the different agents used are:-

Issuer = (Read_book_catalogue, Read_user_book_balance, update_copies, update_user_balace)

Web_crawler= (get_topic, search, store_result)

Display= (<u>read_book_catalogue</u>, display_book_detail, <u>read_crawler_result</u>, display_query_results)

We now turn to safety requirements. These can be specified by means of a list of predicates, typically expressed over the variables listed in a role’s permissions attribute. The safety requirements of the various agents are:-

- *Student_bal NOT > limit*
- *Faculty_bal NOT > limit*

By convention, we simply list safety expressions as a bulleted list, each item in the list expressing an individual safety responsibility. It is assumed that these responsibilities apply across *all* states of the system execution.
With all these definitions in place, it is now possible to precisely define the Gaia roles model. This is used in the analysis phase to define preliminary roles and in the design phase to give the

complete specification of all the roles involved in a system. A roles model is comprised of a set of *role schema*, one for each role in the system. A role schema draws together the various attributes [4]

The role schema of some of the roles is:-

Role schema:book_manager
Description: This preliminary role involves keeping a track of all the books available in the library. It uses a data structure called book_catalogue that contains all the necessary fields of a book. It also stores all the information about each user. His details like name, email id etc and the number and name of the books issued in his name.
Protocols and activities Change_book_catalogue, read_book_catalogue, change_user_details, read_user_details
Responsibilities Liveness: Book_manager= (Change_book_catalogue, read_book_catalogue, change_user_details, read_user_details)

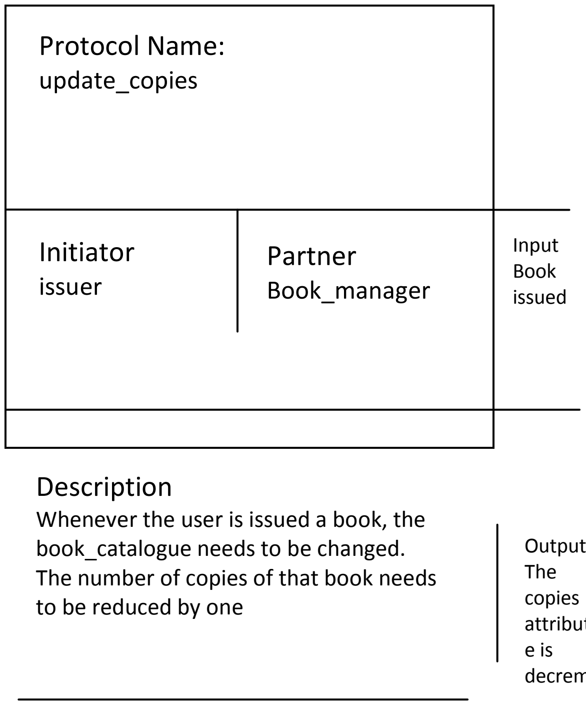
Role schema:Issuer
Description: This preliminary role involves issuing the book to the user on the condition that the copy of the book is available and the user has sufficient balance to issue a book
Protocols and activities Read_book_catalogue, Read_user_book_balance, update_copies, update_user_balace
Responsibilities Liveness: Issuer = (Read_book_catalogue, Read_user_book_balance, update_copies,update_user_balace) Safety: • Student_bal NOT > limit • Faculty_bal NOT > limit

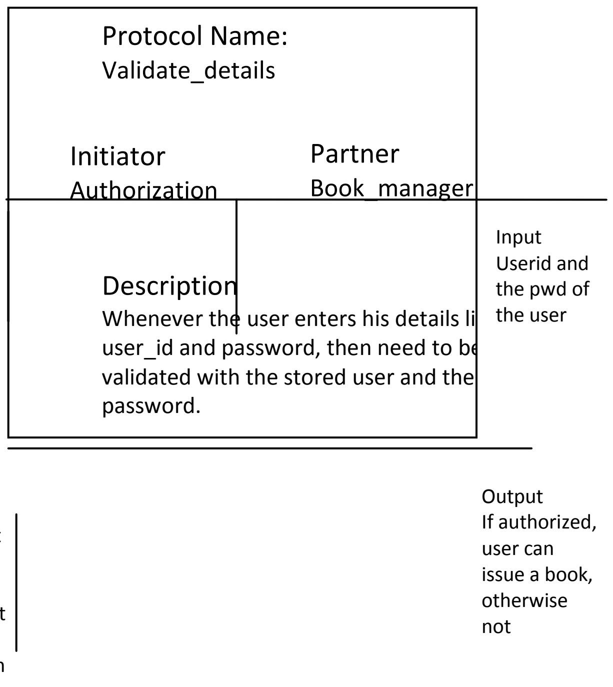
Role schema: authorization
Description: This preliminary role involves authorizing the user. The user id and the password taken from the user are compared with the ones already stored in the database & correspondingly validation is done. After this the user is asked to enter the search criteria. Accordingly a query is sent to the book_manager
Protocols and activities Receive_userid, receive_password, validate_details, send_query
Responsibilities Liveness: Authorization= (Receive_userid, receive_password, validate_details) Safety: • User_id = stored user_id • Password=stored password

Role schema: send_email
Description: This preliminary role involves sending the mail to the user when he issues or returns a book. It selects the template of the mail to be sent. Then it interacts with the book_manager to get the details of the user. Once it's available, it sends the mail to the user with the details of the book issues/ returned by him and also the list of all the books issued in his name.
Protocols and activities Select_template, get_user_details, send_email
Responsibilities Liveness: Send_email = (Select_template, get_user_details, send_email) Safety: • Student_bal NOT > limit • Faculty_bal NOT > limit

4.5 The Preliminary Interaction Model

This model captures the dependencies and relationships between the various roles in the MAS organization, in terms of one protocol definition for each type of inter role interaction. The protocol definition of some is as follows:-





4.6 The Organizational Rules

The preliminary roles and interaction models capture the basic characteristics, functionalities, and interaction patterns that the MAS system must realize, independently of any predefined organizational structure. In Gaia, the perspective on organizational rules is consistent with that on roles' responsibilities: organizational rules are considered as *responsibilities* of the organization as a whole. Accordingly, it is possible to distinguish between safety and liveness organizational rules [12].

The most basic liveness rule can be that a book can be returned only when it has been issued. This can be depicted as

$$\begin{tikzcd} \text{Issuer} \arrow[r] & \text{Return} \end{tikzcd}$$

Another can be that a book can be issued only when the user has been authorized by the authorization role.

$$\begin{tikzcd} \text{Authorization} \arrow[r] & \text{Issuer} \end{tikzcd}$$

5. ARCHITECTURAL DESIGN

The output of the Gaia analysis phase systematically documents all the functional (and to some extent non functional) characteristics that the LRC 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 roles and interactions models.

While the analysis phase is mainly aimed at understanding what the MAS will have to be, the design phase tells the actual characteristics of the MAS. Many factors influence the actual design of the agents and the interaction between them. Like in the case of LRC, it may happen that the limit of books to be issued is different for different faculty. For example, the limit for a lecturer can be less than the limit for an associate professor, which in turn may have a limit less than the professor. Sometimes it may also happen in some of the colleges that the user doesn't need a user_id and a password, since the book is issued by a librarian. The user himself doesn't need to log onto the system. Only on the basis of the roll number of the student and employee code of the faculty, the books are issued. In such a case, the authorization role will not come into the picture. So, the architectural design will entirely depend upon the actual implementation of the Learning resource centre, i.e. on the basis of the requirement specifications given by a particular college.

6. DETAILED DESIGN

6.1 Definition of the Agent Model

There is one to one correspondence between the roles and the agents. In the LRC, it can be said that the different agents can be

- Book_manager
- Register
- Authorization
- Solve_query
- Display
- Issuer
- Web_crawler
- Recommend
- Return
- Send_email

It is desirable to have a book manager agent that takes care of the book_manager role and keeps the entire information about all the books in the library as well as the complete information about the users.

The register agent takes care of the register role and registers a new user in the system. The authorization agent is responsible of validating the user_id and the password of the users, so that the books can be issued and returned.

Similarly, other agents can be designed on the basis of the roles, already described above.

6.2 Service model

The aim of the Gaia *services* 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. Thus, the services model applies both in the case of static assignment of roles to agent classes as well as in the case where agents can dynamically assume roles. For each service performed by an agent, it is necessary to document its properties. The *inputs*, *outputs*, *preconditions* and *post conditions* should be known for all the services performed by an agent. Inputs and outputs can be obtained from the protocols model as described above. Pre- and post conditions represent constraints on the execution and completion, respectively, of services.

6.3 Outcome of the Design Phase

After the successful completion of the Gaia design process, developers are provided with a welldefined set of agent classes to implement and instantiate, according to the defined agent and services model.

7. CONCLUSION AND FUTURE WORK

This paper discusses the agent design of a Learning resource centre to be used in a college. This paper talks about the analysis phase in depth but the architectural design and the detailed design are still to be dealt in detail. In future, the author plans to implement this model in JADE. Once the design of a multi agent system is done the implementation can be very effectively done using the JADE platform.

8. REFERENCES

1. Arenas, J. C. Garcia, and J. J. PerezAlcazar. On combining organizational modeling and graphical languages for the development ofmultiagent systems. Integrated Computer-Aided Engineering Journal (To be published), 2003.

2. Baden Hughes, “Web Crawling” , Department of Computer Science and Software Engineering, UniversityofMelbourne.
3. Ciancarini, p., Omicini, a., and Zambonelli, F. 2000. Multiagent systems engineering: The coordination viewpoint. In Intelligent Agents VI: Agent Theories, Architectures, and Languages. Lecture Notes in Computer Science, vol. 1767. Springer- Verlag, New York, pp. 250–259.

4. Huang, W., El-Darzi, E. and Jin, L. (2007) "Extending the Gaia Methodology for the Design and Development of Agent-based Software Systems", Proceedings of the 31st IEEE Annual International Computer Software and Applications Conference (COMPSAC 2007), Beijing, China, pp. 159-168.

5. Rajashree Shettar, Dr. Shobha G,Proceedings of the International MultiConference of Engineers and Computer Scientists 2008 Vol II,IMECS 2008, 19-21 March, 2008, Hong Kong,ISBN: 978-988-17012-1-3,IMECS 2008

6. J. C. Garcia-Ojeda, J. De J. Perez-Alcazar, and A. E. Arenas. Extending the Gaia methodology with Agent-UML. In Proceedings of the Third International Joint Conference on Autonomous

7. 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 (Bologna, Italy, July). ACM, New York, pp. 3–10.

8. Wooldridge, M., Jennings, N. R., and Kinny, D. 2000. The Gaia methodology for agent-oriented analysis and design. J. Autonom. Agents Multi-Agent Syst. 3, 3, 285–312.

9. Wooldridge M Zambonelli F, N. R. Jennings, , "Developing Multiagent Systems: The Gaia Methodology," ACM Transactions on Software Engineering and Methodology, Vol. 12, No. 3, Jul. 2003, pp. 317-330.

10. Wooldridge M and N. R. Jennings (1995) “Intelligent agents: theory and practice” The Knowledge Engineering Review 10 (2) 115-152.

11. Xiaoming Liu & Dun Tan, “A web crawler”, 2000.

12. Zambonelli, F., Jennings, N. R., and Wooldridge, M. 2001b. Organizational rules as an abstraction for the analysis and design of multi-agent systems. Int. J. Softw. Knowl. Eng. 11, 3 (Apr.), 303–328.