A Multi-Agent System for Integrating Information from Heterogeneous Data Sources

Carlos R. Jaimez-González, Wulfrano A. Luna-Ramírez, Luis A. Ramírez-Colín

Departamento de Tecnologías de la Información, Universidad Autónoma Metropolitana - Cuajimalpa, Av. Constituyentes No. 1054, Col. Lomas Altas, C.P. 11950, México D.F.
{cjaimez, wluna, 207363003}@correo.cua.uam.mx

Abstract. The Internet has changed the way in which information is stored, visualized and shared. In many cases the information we need is on the Internet in different formats and comes from heterogeneous data sources. The tasks of retrieving and integrating information with those features have to be done either manually or by developing a system to do it automatically according to certain rules and criteria. This paper presents a multi-agent system modeled under the BDI approach that integrates information from heterogeneous data sources. Every agent in the group has different abilities and activities, such as retrieving automatically information about call for papers for conferences from different Web sites, which have different structures; extracting information; concentrating it in their own storage system; and visualizing it through a flexible Web site with different search options. Additionally, the multi-agent system provides a Web service interface which allows other systems to retrieve call for papers from it, establishing a certain degree of interoperability to the full system by providing its information in a standard XML format.

Keywords. Multi-agent systems, heterogeneous data sources, call for papers, information retrieval, web objects in xml, information integration.

1 Introduction

The Internet is an increasing source of a great amount of information, where a lot of data can be retrieved from the Web and there is a daily effort for updating and supporting it. The way in which information is stored, visualized and shared is changing due to the continuous generation of content. Unfortunately, despite the availability of specialized Web sites, the registration, search and retrieval of useful information is still a difficult task. In the academic world there are different types of information that are continuously required, such as that related to academic and scientific conferences, which is needed in an accurate and up to date way.

In this paper, we present a multi-agent system (MAS) for automatic web search and retrieval of information of conferences and their corresponding call for papers (CFP). The agents retrieve information about conferences, from different specialized conference Web sites, in which the information is structured in many formats; so an important task of the agents is to extract and format the relevant information that ap-

Advances in Computer Science and Engineering
Research in Computing Science 58, 2012 pp. 271–284
Paper Recived 20-08-2012 and Acepted 01-11-2012

pears in them in order to concentrate it in a centralized storage system; and visualize it through a flexible Web site with some search options. Additionally, the MAS offers a Web service that provides information about the stored CFP in a standard XML format, providing interoperability with other systems. The MAS performs three main activities: a) searching for information, b) collecting, formatting and storing it, and c) showing it to the user in a summarized way. The work presented in this paper is based on the system architecture described in [1].

The rest of the paper is organized as follows. Section 2 provides the preliminaries for this paper. In section 3, we describe three specialized Web sites that handle CFP, which are used by our multi-agent system. Section 4 presents the architecture of the multi-agent system, where the four agents of our system are described. The different interfaces of our system are presented in section 5. Finally, section 6 gives some conclusions and future work.

2 Preliminaries

A practical way to deal with the web search of CFP information across the Web is to design a software system for doing it automatically. In order to implement this system we use the abstraction level provided by the intelligent agents approach; due to the features exhibited by them, such as autonomy, interactivity, proactiveness, rationality, among others [2]. Such system would also be capable of performing goal-driven work, decreasing or even minimizing the external intervention, self-controlling the system actions, reducing conflicts between goals, having a behavior oriented to achieve their goals [3], etc. This approach is an excellent choice for this domain, where it is needed to search, extract, format, store, and visualize heterogeneous information about relevant CFP for conferences.

In order to design such a system, it is convenient to take advantage of the intentional-agent approach; which implies that an agent can have propositional attitudes and mental states [3]. The MAS exposed in this paper is composed of a group of four agents, designed by the Belief-Desire-Intention (BDI) approach [3], where each agent has mental states of three types: beliefs (the real status of the agent in a certain time), desires (its goals or states to achieve), and intentions (the set of actions to perform in order to get its goals). Additionally, according to the BDI approach, each agent has information to execute its tasks, its procedural knowledge (a set of action plans and a queue of events for registering the actions and environment changes during its operation), and a set of abilities to accomplish its goals and carry out the assigned tasks.

3 Call For Papers

This section describes three specialized Web sites dedicated to concentrate Calls For Papers (CFP) from conferences about different topics: *WikiCFP* [4], *ConferenceAlerts* [5], and *DBWorld* [6]. They have different features and options to handle CFP, which will be highlighted in the following subsections.

3.1 WikiCFP

WikiCFP is a Web site for registering manually and searching for conference CFP. It allows different search options and facilities, such as listing popular CFP, popular categories, CFP by category, CFP added recently, etc. It also allows searching for CFP by year and keyword, where the keyword can be any text that is contained in the information of the CFP. There are two types of users in *WikiCFP*: non-registered and registered. A non-registered user can only browse all the categories and CFP; a registered user can additionally create a list of their favourites CFP, and receive notifications about the status of the CFP being watched.

The navigation and the format of the information contained on a CFP are simple. Figure 1 shows a screenshot of the *WikiCFP* Web site with a list of CFP, where every CFP in the list contains the following basic information: name of the conference, URL, when and where the conference takes place, submission deadline, notification date. Internally, the information of the CFP is semi-structured, and is contained in HTML tables.

Fig. 1. WikiCFP Web site.

3.2 ConferenceAlerts

This Web site allows different search options in two main categories; one is by interest topic, which is classified by areas; and the other is by country, which will order the CFP by geographical regions. The Web site also allows registering a new event, promoting an event, and subscribing to receive updates. Some of the information about CFP that is provided is the following: day, event name, city, country, among others. When selecting a specific event, we can find complementary information that is transcendental to maintain informed users. It should be noticed that this Web site not only allows the registration of academic conference CFP, but also of different types of events, such as meetings, commercial conferences, etc. Figure 2 illustrates a screenshot of the *ConferenceAlerts* Web site.

Fig. 2. ConferenceAlerts Web site.

3.3 DBWorld

The *DBWorld* Web site is also used for registering manually and searching for conference CFP. It has a very simple graphical user interface, where all the conferences are contained in a list, and it allows the following actions: searching for CFP, geographic position of the conference on a map, access to old CFP, and registration of a new conference CFP. There is a map with a searching tool, where it is possible to search for a CFP using filters or keywords. It shows the results on the map and basic information about the CFP, such as name of the conference, dates, submission deadline, place of the conference, URL, notification date, etc. This Web site also uses the format of a wiki, where the information about the CFP is in a list with a specific pattern. Although the information is semi-structured, it has a common pattern to represent the information of a CFP in tables. A screenshot of the *DBWorld* Web Site is shown in Figure 3.

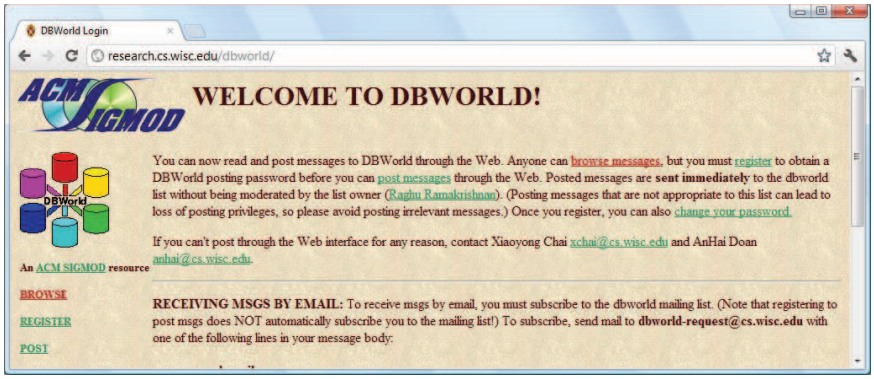

Fig. 3. DBWorld Web site.

4 Multi-Agent System Architecture

The architecture of our multi-agent system is composed of four intentional agents, designed using the BDI approach [3], [7]; and the Prometheus methodology, using the Prometheus Design Tool (PDT) [8], a graphical editor for specifying, designing and implementing intelligent agents. The development process following this methodology has three stages:

1. **System specification**: oriented to identify the human-agent interaction described by perceptions and actions, defining the goals, and some steps of functionality based in the defined roles.
2. **Architectural design**: where the agents of the system are defined, the MAS entire structure is determined, and the system dynamics is captured by means of interaction protocols.
3. **Detailed design**: where the plans, data, capabilities and actions of the agents are defined. PDT was used for designing this detailed design of agents.

Figure 4 illustrates the System Overview Diagram of PDT, with the architecture of the MAS, where it can be seen the agents and the three main processes mentioned above, which are associated with them.

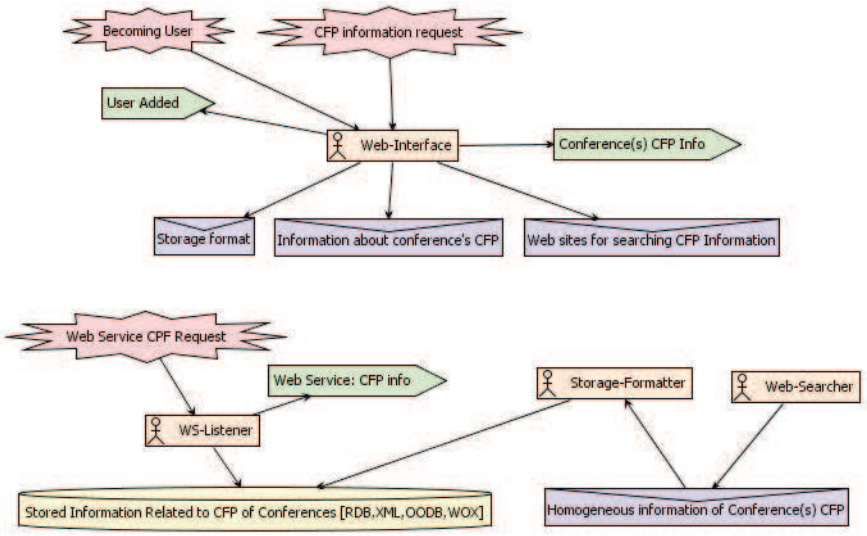

Fig. 4. The MAS architecture, with the Web-Interface, Web-Searcher, Storage-Formatter and the WS-Listener agents. The CPF information is stored in different formats: relational, object oriented, XML and Web Objects in XML.

In the first place it is depicted the *Web -Interface* agent, which has the following activities: it provides the communication with the users of the system; it gives to the

other three agents the instructions for searching information on the Internet, it specifies the format for storing the heterogeneous information collected from the Internet; and it shows to the users the results of their requests for CFP information. The *Web-Searcher* agent is in charge of retrieving CFP from different Web sites, and downloading this information to the system. The *Storage-Formatter* agent takes the downloaded CFP and cleans them up, by converting them into XHTML files and finally takes the clean CFP, reads them, extracts their information, and stores them in the storage system. Additionally, the *WS-Listener* is the agent responsible for providing the responses of Web Services requests of external systems; it executes the queries provided in the Web Service and sends the result to the requesting system.

The group of BDI agents were implemented using *Jason* [9], which is an interpreter of the agent-oriented language *AgentSpeak(L)* [3], wrote in the Java programming language. *Jason* is a graphical environment for agent development, which allows the edition and execution of agent programs, and includes the opportunity to extend the abilities of each agent through Java programs (called internal actions) that implement a variety of tasks in order to achieve certain goals.

The following three subsections explain in detail the activities carried out by the two main agents of our system: *Web-Searcher* and *Storage-Formatter*.

4.1 Retrieving a CFP

The *Web-Searcher* agent retrieves a set of CFP from the Web sites described in previous sections (*WikiCFP, ConferenceAlerts* and *DBWorld*). In order to carry out this task, the agent uses the `wget` command, which is launched automatically once a day, according to the agent goals. The following code fragment shows a sample of the *Web-Searcher's* execution of the `wget` command to retrieve only a specific CFP from the *WikiCFP* Web site.

```
wget -N --html-extension
http://www.wikicfp.com/cfp/servlet/event.showcfp?eventid=$a
```

It can be seen that the `wget` command has some parameters to specify how to download a specific CFP, which is contained in a Web page. The `-N` parameter is used to specify how deep the command goes and retrieves the contents of the Web page; the `--html` and the `-extension` parameters specify to store the Web page retrieved as an HTML file; and finally the rest of the string represents the URL of the Web page (CFP) that will be downloaded. The command uses the `$a` variable to represent the `id` of the CFP from the *WikiCFP* Web site (this id goes from 1 to 19,000 approximately, and this number is being increased daily as new CFP are added to the Web site). In order to illustrate a real example of how this agent retrieves a Web page with a CFP, we will use the CFP of the *12th Congress on Computing (CORE 2012),* which is already registered in the Web site. The agent retrieves the Web page representing this CFP from the following URL:

```
http://www.wikicfp.com/cfp/servlet/event.showcfp?eventid=25031
```

Figure 5 shows a screenshot of the Web page that shows the *CORE 2012* CFP with `id=25031`. This is retrieved from the *WikiCFP* Web site.

Fig. 5. CFP for CORE 2012, retrieved from the *WikiCFP* Web site.

Once the *Web-Searcher* has finished retrieving the CFP from the Web site, its next goal is to generate an HTML file and store it in the MAS data storage system, as described previously. The file name for this specific CFP is the following: `event.showcfp?eventid=25031.html` (the *Web-Searcher* agent takes the last part of the URL to name the HTML file). The following code fragment, written in Jason [9], shows the plan followed by the *Web-Searcher* agent to accomplish its goal of downloading a Web page, before starting to process it.

```
// Web-Searcher Agent in project WebCFPAgent.mas2j
/* Initial beliefs and rules */
...
/* Initial goals */
...
/* Plans */
...
+!searchCFP(CFPID): true <-
  cfpActions.searchCFP(CFPID).
...
```

The code starts by defining the *Web-Searcher* beliefs and rules, its goals, and plans; then it can be seen a small fragment of the plan to satisfy the `searchCFP` goal, which takes the `CFPID` as parameter representing the Web page to be downloaded. The plan executes an internal action, implemented in the Java class `cfpActions.searchCFP`,

and it is the responsible to execute the `wget` command as previously described. It should be noticed that this action is executed for the three CFP Web sites currently considered for this system. All actions carried out by the agents are also implemented through Java classes. The next step in the process is to clean up the resulting file in order to get it ready to be read, and extract its conference information.

4.2 Cleaning Up a CFP

In this task, the *Storage-Formatter* agent takes every downloaded HTML file (CFP), and cleans it from any unmatched start and end tags, converting the stored HTML file into a well-formed XHTML file. Having this file is essential in the MAS, in order to use the *Storage-Formatter* agent's ability of XML parsing and extracting its conference information. The following code fragment shows part of the structure of the XHTML file generated by this agent. It can be observed information about the conference, such as its full name, URL, conference dates, place of the conference, abstract registration deadline, submission deadline, notification date, among other information. We have removed some markup from the generated XHTML file for display purposes, and to show only the important information that the agent needs to retrieve, which is highlighted in bold and red colour.

```
<span property="v:description">CORE 2012 : 12th Congress on
Computing</span>

<tr><td align="center">Link: <a href=
"http://www.core2012.cic.ipn.mx/" target="_newtab">
http://www.core2012.cic.ipn.mx/</a></td></tr>

<table cellpadding="3" cellspacing="1" align="center">
   <tr bgcolor="#e6e6e6">
      <th>When</th>
      <td align="center">Nov 28, 2012 - Nov 30, 2012</td>
   </tr>
   <tr bgcolor="#f6f6f6">
      <th>Where</th>
      <td align="center">Distrito Federal, México</td>
   </tr>
   <tr bgcolor="#f6f6f6">
      <th>Submission Deadline</th>
      <td align="center">
         <span property="v:startDate" content="2012-08-
19T00:00:00">Aug 19, 2012</span>
      </td>
   </tr>
   <tr bgcolor="#e6e6e6">
      <th>Notification Due</th>
```

```
      <td align="center">
         <span property="v:startDate" content="2012-10-
21T00:00:00">Oct 21, 2012</span>
      </td>
   </tr>
</table>
```

This XHTML file can now be taken by this agent to be read and extract its information for storing it in the MAS storage system. Next subsection explains this process, which is also carried out by the *Storage-Formatter* agent.

4.3 Parsing and Storing a CFP

The final step in the process is carried out by the *Storage-Formatter* agent, which stores the CFP information permanently in the system. The agent takes the XHTML file; reads it using its XML parsing ability; extracts all the CFP information according to the structure of the XHTML file; and stores it in the desired storage medium. It should be noticed that the agent carries out an analysis of the internal structure of the CFP, in order to extract its information. With this analysis the *Storage-Formatter* agent determines exactly where every piece of information is located, such as the name of the conference, submission deadline, notification date, conference dates, etc.

Concerning the storage medium for the CFP, the MAS has implemented five different storage mechanisms, which can be selected through the *Web-Interface* agent. These mechanisms are the following: *MySQL* as a relational database [10]; *DB4O* as an object-oriented database [11]; *eXist* as an XML database [12]; XML files using the *Web Objects in XML* serializer [13], [14]; and XML files using custom XML serialization. The MAS is under development, testing and evaluation of the performance of the five different storage mechanisms, in order to determine the most efficient implementation for its final version.

This final process aims to leave all the important information about CFP in only one homogeneous place, from which it can be accessed later through different interfaces. The basic structure we use to store a CFP depends on the specific implementation, but the information is always the same. The following code fragment illustrates all the fields used to store the CFP information internally, represented as Java attributes of a particular class, which is part of the internal actions for this agent.

```
private String abreviatura;
private String nombre;
private String fechaInicio;
private String fechaFin;
private String fechaDeadline;
private String ciudad;
private String pais;
private String url;
private ArrayList listaCategorias;
```

The outcome of this step is to have the information of every CFP stored in our system permanently. An example of the information that the agent extracts from the CFP for the *CORE 2012* is shown in Figure 6. It should be noticed that this process is executed by the agents for every CFP in the Web sites mentioned previously. Additionally, the following code fragment illustrates part of the XML document generated for the *CORE 2012* CFP with all its information.

```
<conferencia>
   <abreviatura>CORE 2012</abreviatura>
   <nombre>12th Congress on Computing</nombre>
   <fechaInicio>Nov 28, 2012</fechaInicio>
   <fechaFin>Nov 30, 2012</fechaFin>
   <fechaDeadline>Aug 19, 2012</fechaDeadline>
   <ciudad>Distrito Federal</ciudad>
   <pais>Mexico</pais>
   <url>http://www.core2012.cic.ipn.mx</url>
   <categoria>computer science</categoria>
</conferencia>
```

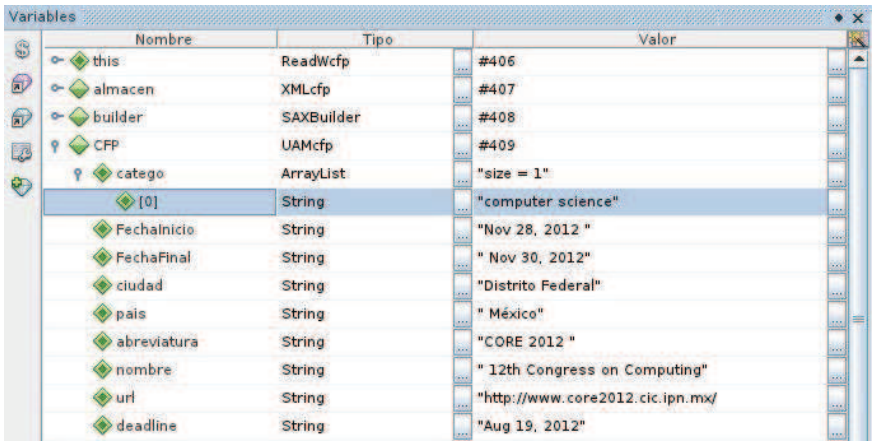

Fig. 6. Information extracted from the CORE 2012 CFP.

5 Interfacing with the Multi-Agent System

This section briefly describes the two types of interfaces of the MAS. The first interface is using the *Web-Interface* agent, which is the responsible for the interchange of information between a normal user and the MAS. The other interface with the MAS is through the *WS-Listener* agent, which executes a Web service that allows the interaction of the user with the system by retrieving specific information using CFP queries.

5.1 Web-Interface Agent

The *Web-Interface* agent exhibits as part of its abilities a Web interface, which is divided in categories and countries, but it is also possible to search CFP with specific criteria, such as their name, dates, deadline for submitting a paper, place where the conference will take place, or a combination of them. The *Web-Interface* agent can communicate the parameters established by the user to execute searching of conference CFP, so that the *Web-Searcher* and the *Web-Interface* agents can start working together. The *Web-Interface* agent receives from the Web interface a set of parameters representing the search options specified by the user, and builds a set of queries used by the *Web-Searcher* agent to search for the CFP that comply with the desired criteria; then the CFP selected are shown in an ordered list.

Additionally, the *Web-Interface* agent has the ability to allow searching and downloading individual CFP. The following Jason code fragment shows the plan followed by the *Web-Interface* agent to accomplish its goal of searching for an individual CFP and downloading the corresponding Web page. After downloading the CFP, it is passed to the *Web-Searcher* agent to be processed as described in section 4.

```
// Web-Interface Agent in project WebCFPAgent.mas2j
/* Initial beliefs and rules */
...
/* Initial goals */
...
!startSearch.
/* Plans */
...
/* Searching Plan*/
+!startSearch :  cfpid(CFPID) <-
  .send(web_Searcher,achieve,searchCFP(CFPID)).
...
```

The code starts by defining the *Web-Interface* beliefs and rules, its goals, and plans as in the rest of the agents. It can be seen a short fragment of the plan, where the agent tries to satisfy a requirement from the user by starting the retrieval of a CFP. The user's action fires the *startSearch* goal of this agent; and sends a message to the *Web-Searcher* agent in order to begin the task of retrieving the CFP information as the user requested; just as described in section 4, where the *Web-Searcher* agent takes the control, downloads the CFP, and process it.

5.2 WS-Listener Agent

The *WS-Listener* is the agent that provides the communication of the MAS with external systems through a Web service interface. This agent is the responsible for listening requests from systems, executing the Web services indicated by the parameters of each request, and sending to the requesters the information about CFP that is stored

in the MAS. It should be noticed that the *WS-Listener* agent has a set of Web services, depending on the type of request. At the moment this agent is under development, but it is designed to return the same information as the *Web-Interface* agent; the only difference is that the *WS-Listener* returns the result in XML standard messages, and the *Web-Interface* agent gives the result through a Web interface.

6 Conclusions and Future Work

This paper presented an intentional agent-based system that automates and facilitates the retrieval of CFP information from heterogeneous sources, specifically from three different specialized Web sites dedicated to concentrate information about scientific conferences and CFP. This task is performed by a group of four agents designed under the BDI approach: *Web-Interface* agent, *Web-Searcher* agent, *Storage-Formatter* agent and *WS-Listener* agent. These agents retrieve the relevant information of CFP for conferences, store it in the storage system, and show it to users by means of a flexible Web site, which is part of the *Web-Interface* agent duties. Some interoperability with other external software systems is incorporated in our MAS through the *WS-Listener* agent, which listens for Web service requests, and returns the results in the form of standard XML messages.

The MAS presented has five different implementations for storing the heterogeneous information retrieved: MySQL as a relational database, DB4o as an object-oriented database, eXist as an XML database, Web Objects in XML as an XML serializer, and XML files with custom serialization. A sample Jason code of some of the agents is shown, in order to demonstrate the types of plans that they follow. It is being carried out an investigation and evaluation of the suitability and performance of every storage approach, in order to determine the best way for storing the information retrieved by the three specialized Web Sites. As part of the future work, we also consider the inclusion of two more intentional agents: the *CFP-Recommender*, which would be in charge of the generation of recommendations about conference CFP, based on the user profile and similar user profiles; and the *CFP-Ranking* agent which would be responsible of ranking the CFP according to specialized ranking Web sites for conference CFP.

References

1. Luna-Ramírez, W.A., Jaimez-González, C.R., Ramírez-Colín, L.A.: An Agent-Based System Architecture for Retrieving Call for Papers. In: IADIS International Conference on Applied Computing 2012, pp. 434-436, ISBN: 978-989-8533-14-2, Madrid, Spain (2012)
2. Wooldridge, M., Jennings, N.R.: Intelligent Agents: Theory and Practice. Knowledge Engineering Review 10(2), (1995)
3. Rao, A.S.: AgentSpeak(L): BDI Agents Speak out in a Logical Computable Language. In: Van de Velde, W. and Perram, J. (eds), Proceedings of the Seventh Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96), 22-25 January. Eindhoven, Netherlands. LNAI 1038:42–55, Springer-Verlag, Heidelberg, Berlin, (1996)

4. A Wiki for Calls For Papers. Available at: http://www.wikicfp.com/. Last accessed in October 2012
5. Conference Alerts: Academic Conferences Worldwide. Available at: http://www.conferencealerts.com/. Last accessed in October 2012
6. DBWorld. Available at: http://research.cs.wisc.edu/dbworld/. Last accessed in October 2012
7. Rao, A.S., George, M.P.: BDI-Agents: from Theory to Practice. In: The First International Conference on Multiagent Systems, San Francisco, (1995)
8. Padgham, L., Winikoff, M.: Developing Intelligent Agent Systems: A Practical Guide. Available at: http://www.cs.rmit.edu.au/agents/prometheus/, John Wiley and Sons, (2004). Last accessed in October 2012
9. Bordini, R.H., Hubfner, F.J.: Jason. A Java-based AgentSpeak Interpreter Used with Saci for Multi-agent Distribution over the Net, (2006). Available at: http://jason.sourceforge.net/wp/. Last accessed in October 2012
10. MySQL DB: The world's most popular open source database. Available at: http://www.mysql.com/. Last accessed in October 2012
11. DB4Objects DB: by Versant. Available at: http://www.db4o.com/. Last accessed in October 2012
12. eXist: An open source database management system built using XML technology. Available at: http://exist-db.org/exist/index.xml. Last accessed in October 2012
13. Web Objects in XML: an XML serializer for Java and C# objects. Available at: http://woxserializer.sourceforge.net/. Last accessed in October 2012
14. Jaimez-González, C.R., Lucas, S.M.: Easy XML Serialization of C# and Java Objects. In: Balisage: The Markup Conference 2011, vol. 7, Balisage Series on Markup Technologies, ISSN: 1947-2609, doi:10.4242/BalisageVol7.Jaimez01, Montréal, Canada (2011)