Automatic Ontology Construction for a Multiagent-based Software Gathering Service*

E. Mena[1], A. Illarramendi[2], and A. Goñi[2]

[1] IIS depart., Univ. de Zaragoza. Spain. http://www.cps.unizar.es/~mena/
[2] LSI depart., UPV. San Sebastián. Spain. http://siul02.si.ehu.es/~jirgbdat/

Abstract. Ontologies and agents are two topics that raise a particular attention those days from the theoretical as well as from the application point of view. In this paper we present a software gathering service that is mainly supported by an ontology, SoftOnt, and several agents. The main goal of the paper is to show how the SoftOnt ontology is built from distributed and heterogeneous software repositories. In the particular domain considered, software repositories, we advocate for an automatic creation of a global unique ontology versus a manual creation and the use of multiple ontologies.
Keywords: ontologies as metadata, agent technology, distributed software retrieval

1 Introduction

Currently, there is a great deal of interest in the development of ontologies to facilitate knowledge sharing in general, and data repositories integration in particular. In this paper we advocate using an ontology to describe *semantically* the content of a set of data sources containing pieces of software. So, instead of users have to deal directly with different software repositories, the goal is that a system uses an ontology to help users to retrieve software. Furthermore, user requests will be mapped automatically to queries on that ontology and the proposed system takes the responsibility of retrieving and installing the appropriate software in an efficient way using for that specialized agents.

One of the most frequent tasks of computer users is to obtain new software, in order to improve the capabilities of their computers. For that, a common procedure is to visit some of the several websites that contain freeware, shareware and demos (such as Tucows [24] and CNET Download.com [4]), games (such as Games Domain [2] and CNET Gamescenter.com[5]), java-related software (like Gamelan [6]) or many others. Different kinds of users need different kinds of software. For example, naive users could be interested in entertainment programs like computer games and CD players. On the contrary, some other users could be interested in DBMS's, word processors, spreadsheets, or backup utilities, among

* This work has been supported by *CICYT (Comisión Interministerial de Ciencia y Tecnología*, Spain [TIC97-0962]), MoviStar (a spanish cellular phone company) and the University of the Basque Country.

many others. Other kind of software could be interesting for most of users, like antivirus, Web browsers, etc.

We summarize in the following the three main problems that users must face when they want to obtain a new piece of software:

1. To know the different programs that fulfil their needs. Not only their names but also where to find them in the vast Web space. This task is complex enough to discourage naive users from installing new software by themselves.
2. To know the features of their computers, in order to select the most appropriate version. This task implies having technical knowledge about her/his system (CPU, OS and version, free disk space available, RAM memory, etc.) and the software installed previously (for requesting a full version or just an update).
3. To be aware of new software and/or new releases of software of interest. Although few commercial programs currently alert about new releases, users need to keep an eye on the Web, or on other repositories, if they want to be informed about new software that could be of their interest.

With the goal of alleviating the previous problems we have developed a Software Retrieval Service to provide users with a transparent access to local or remote software sites [16]. We present in this paper the technique used to encapsulate software sites which is based on the use of an ontology and the agent technology.

Ontologies and agents are two topics that raise a particular attention nowadays [10, 1, 11]. Several works can be found in the literature that consider each topic separately and even jointly as in our case [12]. However, we have not found, so far, any significant one that also treats the main focus of this paper: how to build an ontology automatically from distributed and heterogeneous websites that contain software repositories. Among those that can be related somehow, we mention some of them. In [9] they use Yahoo! [25] categories to describe documents. In [13] they propose to extend HTML in order to annotate pages with terms from a concrete ontology. Finally, in [14] they use an ontology managed by an agent to resolve knowledge disparities that may occur when heterogeneous information sources must interact.

In the rest of the paper, we first explain the reasons for which we propose to build one ontology in an automatic way (Section 2). In Section 3 we provide a brief description of the agents used in the Software Retrieval Service. The translation step that permits obtaining ontologies from software repositories and the integration step that generates a global ontology by integrating the previously obtained ones are explained in Sections 4 and 5, respectively. Finally, some conclusions appear in Section 6.

2 SoftOnt: a Software Ontology Built in an Automatic Way

Ontologies are very interesting specification tools for the task of describing a set of terms of interest in the particular domain of software repositories. The terms

of the ontology are linked with the corresponding software repositories through the mapping information, which is managed by the proposed system. So, the users only have to express their software needs by using terms in the ontology, and do not have to care about the distribution and heterogeneity of the (web or local) repositories that contain software.

2.1 One Ontology vs. Multiple Ontologies

When ontologies are selected to describe data sources contents of a particular domain, we must decide on dealing with one global integrated ontology or dealing with multiple ontologies linked by interontology relationships. In general, the relevant features that must be taken into consideration before choosing between a global ontology or several ontologies linked by semantic relationships are the following:

- The number of data sources involved. A huge number will lead to very complex mappings in the case of a global ontology.
- The number of categories that we will need to extract from the data sources. A huge number will lead to many terms and relationships between them, in the case of a global ontology.
- The vocabulary problem existing across data sources. The integration of data sources designed under different points of view will lead to a global ontology with many terms due to the existence of many specializations, generalizations but not synonyms between those data sources.

Taking the above features into account, we propose to deal with only one ontology in the considered software domain for the reasons that we enumerate in the following:

1. The number of data sources is low. Just by integrating a few software repositories we would offer access to most of the (freeware/shareware) software available on the Web. Notice that the most popular software websites have a good set of categories and pieces of software. In fact, we usually find what we look for just visiting one website.
2. The number of categories is not very high. Although different organizations could develop different categorizations of kinds of software, the biggest websites keep that number below one thousand (around 270 in Download.com and around 900[1] in Tucows). Only in the case of several hundreds or thousands of kinds of software (a huge ontology) we would need a different approach.
3. The vocabulary heterogeneity problem is limited. The restricted domain of kinds of software does not allow a big heterogeneity with respect to names used to describe different categories.

[1] After removing inner synonyms among different OS.

2.2 Automatic vs. Manual Ontology Construction

Another decision that must be taken when dealing with ontologies is the automatic versus manual ontology building process. It is widely accepted that the automation of the process of federating different data sources is difficult due to the need of managing semantic information that cannot be extracted automatically [22]. However, we explain in the following the features of our context, mostly concerning to the kind of data sources involved, that allow an automatic federation:

– Low syntactic heterogeneity. For the public websites that contain software (like Tucows, Download.com, etc.), we need to develop specialized wrappers that extract information from HTML or XML pages. It is important to stress that most of the public websites are already categorized (games, networking, entertainment, by OS, etc.) and we can take profit of it: the hierarchical categorization of a website can be used to create an ontology where subcategories in the website are transformed into subterms (specializations) in the ontology (see Figure 1). Concerning to the access to local software repositories, we would have control on them so the development of a wrapper is not a difficult task.

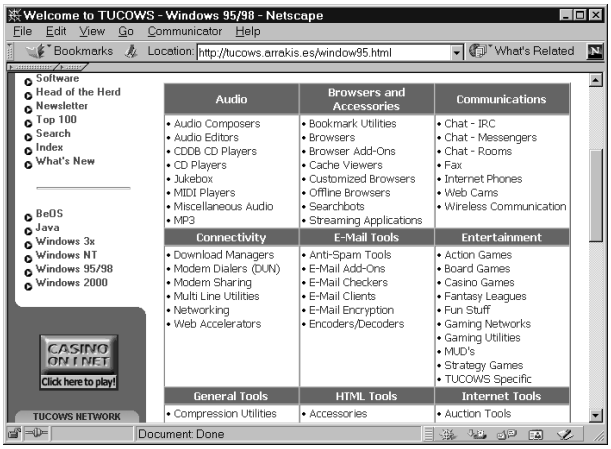

Fig. 1. Windows95 categories in Tucows

– Low semantic heterogeneity. In similar contexts, very different categorizations can be done by different organizations, for instance, bibliographic data [15]. That leads to the establishment of complex mapping relationships between the data elements in the data sources and the terms in the semantic descriptions (ontologies). These semantic relationships must be asserted by persons, although semi-automatic approaches can be taken in order to suggest some probable properties by considering the syntactic relationships [23]. However, in the scenery of software repositories the problem is minor. This can be seen

by visiting different software websites: most of them have many categories in common, or use synonyms to denote the same category. For example, kinds of games in Download.com (Figure 2) are quite similar to categories of Entertainment in Tucows (Figure 1).

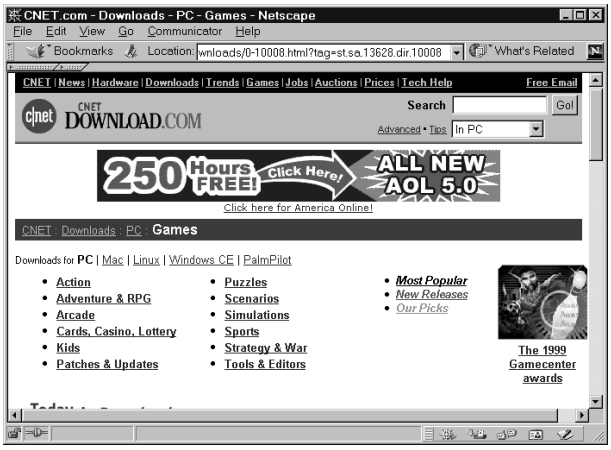

Fig. 2. Games categories in Download.com

Therefore, we believe that an automatic integration is possible and is a good solution for the construction of a software catalog. By avoiding human intervention our system can update, with a certain time granularity, the software offered, which definitely increases the quality of the proposed service. For the integration process, the two classical steps defined in the specialized literature of federating data sources, translation and integration, will be followed.

3 Agents involved in the system

In this section we present briefly all the agents that take part of the Software Retrieval Service. This service, situated in a concrete server that we call GSN[2], offers to the users the possibility to select, retrieve and install software in an easy and efficient way. The Software Retrieval Service uses the SoftOnt ontology mentioned in the previous section.

Agents are executed in contexts denominated *places* [20]. Mobile agents can travel from one place to another. The proposed service incorporates four places (see Figure 3):

1. **The User place**, located on the user computer. It includes an agent that belongs to the user, *Alfred*. Alfred is an efficient majordomo that serves the

[2] The *Gateway Support Node (GSN)* is the proxy that provides services to computer users.

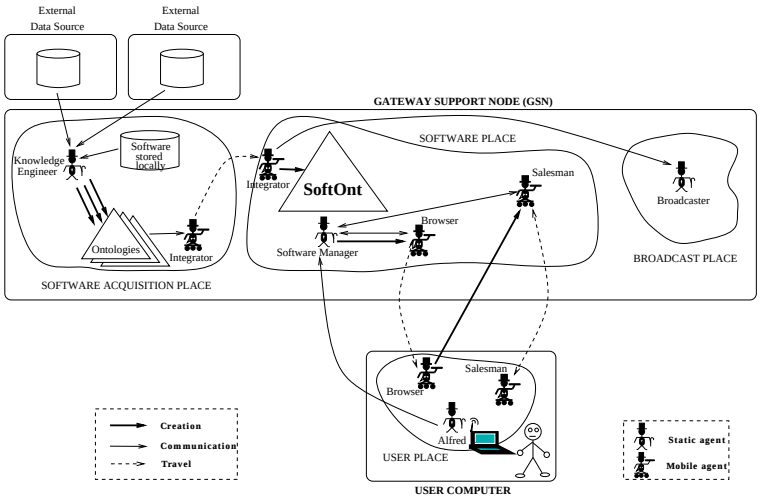

Fig. 3. Main architecture for the Software Retrieval Service

user and is in charge of storing as much information about the user computer, and the user himself, as possible. Alfred is the mediator between the user and the rest of the system.

2. **The Software Acquisition place**, located on the GSN. It groups those agents related to the *SoftOnt* ontology creation process (the main goal of this paper): 1) *The Knowledge Engineer* whose goal is to mine (local or remote) software repositories, with the help of specialized wrappers, in order to obtain a semantic description (an ontology) for each repository; and 2) *The Integrator*, which performs the integration of all the ontologies obtained by the Knowledge Engineer, with the goal of obtaining the ontology SoftOnt. During the integration process it uses a thesaurus for the automatic vocabulary problem resolution.
3. **The Software place**, located on the GSN. It groups those agents related to the SoftOnt ontology exploitation process: 1) *The Software Manager* whose main goal is to provide the Browser agent with a catalog of the available software, after consulting and pruning the SoftOnt ontology according to the needs expressed by Alfred (on behalf of the user); 2) *The Browser*, whose goal is to interact with the user in order to refine a catalog of software until the user finally chooses a concrete piece of software; and 3) *The Salesman* which is in charge of finally carrying the program selected by the user to her/his computer, and performing any e-commerce interaction needed (which depends on the concrete piece of software).
4. **The Broadcast place**, located on the GSN. It includes *The Broadcaster*, whose goal is to inform users about new software releases.

A more detailed description of the role and interaction of these agents in the Software Retrieval Service can be found in [16].

4 The Translation Step: Obtaining a Description of Data Sources

In our context, most of the underlying repositories are remote websites containing software, which are classified as semistructured repositories, i.e., there does not exist a data schema of the information stored. Fortunately, HTML pages in websites containing software classify the different pieces of software in several categories, and we can take advantage of this. The solution for the translation step is the construction of specialized *wrappers* [7, 15, 21, 8] that access the HTML pages that compound the website and extract from them the different categories of software that can be found in such a website.

The design of such wrappers must consider that HTML pages can change. Some works have developed techniques to easily construct or adapt wrappers to semistructured data repositories [7, 15]. These works suggest the use of free-context grammars to define the structure of the data sources (HTML pages, in our case). Thus, wrappers extract certain information from the pages (software categories, in our context), taking a grammar as basis. If a change in the structure of some HTML page happens, the grammar can be adapted to the new syntactic structure of the page, but that would not have a great impact from the point of view of implementation. Following the suggestion, we have built wrappers based on grammars in order to extract the software categories from remote websites of public software.

```
<TD VALIGN="TOP" BGCOLOR="#ffffff">
<FONT FACE="Tahoma" SIZE=2><P>
 <a href="action95.html">Action Games</a><br>
 <a href="board95.html">Board Games</a><br>
 <a href="casino95.html">Casino Games</a><br>
 <a href="fanleague95.html">Fantasy Leagues</a><br>
 <a href="fun95.html">Fun Stuff</a><br>
 <a href="gnetwork95.html">Gaming Networks</a><br>
 <a href="gameutils95.html">Gaming Utilities</a><br>
 <a href="mud95.html">MUD's</a><br>
 <a href="strat95.html">Strategy Games</a><br>
 <a href="tucows95.html">TUCOWS Specific</a><br>
</FONT></TD>
```

(1)

```
<start> ::= <begin> <list-of-categories> <end>

<begin> ::= '<TD VALIGN="TOP" BGCOLOR="#ffffff">'
            '<FONT FACE="Tahoma" SIZE=2><P>'

<list-of-categories> ::=  <category>
                        | <category> <list-of-categories>

<category> ::= '<a href="'
               link  {store URL with subcategories or software}
               '">'
               category-name {store name of category analyzed}

<end> ::= '</FONT></TD>'
```

(2)

Fig. 4. Tucows Entertainment categories in HTML (1) and grammar-based category extractor (2)

We can observe in Figure 4, on the left, that the information related to the categories and links to other pages containing sub-categories or software are included in the HTML description, but mixed with HTML tags. In Figure 4, on the right, we show a grammar that, taking the previous HTML page as entry, extracts the different Tucows Entertainment categories. We would like to stress that the category extraction mechanism also extracts all the information needed to download the pieces of software under each category.

In this manner, a wrapper constructed with this technique can 1) access a remote website, 2) mine its web pages and 3) obtain as result of that process a set of categories and the features of the programs belonging to them. With

that information, an ontology can be easily built using some KBMS. In Figure 5 we show the concept hierarchy corresponding to an ontology obtained using this technique: it is the semantic description of Tucows that we looked for. Due to space limitations we have only detailed the branch Entertainment.

Fig. 5. Extract of ontology for Tucows

When an ontology describes a set of data sources it is necessary to define some kind of mapping information between terms in the ontology and data elements in the data sources, where the information is stored. In our context, during the process of mining software repositories to build an ontology, all the mapping information needed is also recollected.

Thus, the above process of translation is repeated for other websites, using different wrappers as they have different HTML pages and structure. See Figure 6 where the ontology corresponding to Download.com is shown. Due to space limitations we have only detailed the branch Games.We can observe that the obtained ontologies present a certain similarity.

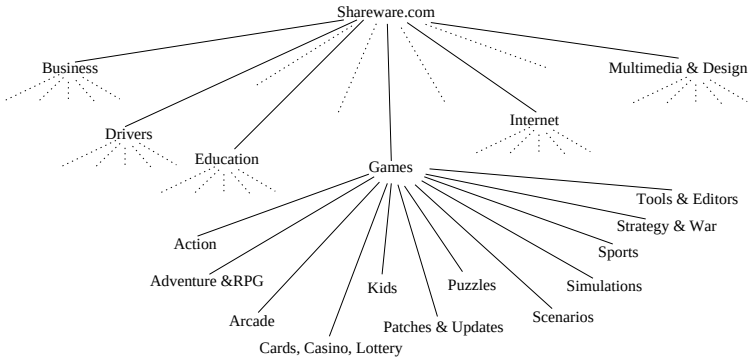

Fig. 6. Extract of ontology for Download.com

Finally, remember that the task corresponding to the translation step is developed by the Knowledge Engineer agent.

5 The Integration Step: Creation of SoftOnt

The ontologies obtained from the different software repositories must be integrated into only one ontology[3], SoftOnt. As different repositories can have classified their software in different ways, the main problem to perform an automatic integration is the *vocabulary problem*, i.e., the existence of synonyms, hyponyms or hypernyms among terms in different ontologies. In an open and dynamic environment, this problem is very difficult to solve although some approaches have been suggested, like the OBSERVER system [15].

5.1 The Vocabulary Problem

In our context, the vocabulary problem is not so serious due to the restricted nature of the type of information stored in the software repositories. There are many kinds of software but in most of the repositories that we have studied, the categories used are very similar: Internet tools, games, Audio, Video, etc. In a first approach we decided to use only a specialized thesaurus (well-known web tools as WordNet [19] could be used) in order to deal with synonyms, hyponyms and hypernyms. More expressive but complex mechanisms, that will require some kind of user intervention, are out of the scope of this paper (a proposal appears in [17, 18]).

As example, we present in Figure 7 the result of integrating the Tucows and Download.com ontologies, after detecting the following semantic properties using WordNet as thesaurus (terms from Tucows are in uppercase and terms from Download.com are in lowercase): 'ENTERTAINMENT' subsumes 'Games', 'ACTION GAMES' is a synonym of 'Action', 'CASINO GAMES' is subsumed by 'Cards, Casino, Lottery', and 'STRATEGY GAMES' is subsumed by 'Strategy & Wars' (only semantic properties between 'ENTERTAINMENT' and 'Games' are included). Many other properties could be found with the help of a thesaurus specialized on Computer Science.

For the task of integrating, with the help of a thesaurus for automatic vocabulary problem resolution, the different ontologies obtained by the Knowledge Engineer agent, we have designed the *Integrator* agent, which is in charge of creating SoftOnt. The Integrator is a mobile agent: after the integration is performed it moves from the Software Acquisition place to the Software place in order to update the previous version of SoftOnt. In this way, the Software Acquisition place could be on a different computer than the GSN. The Integrator is also capable to detect the changes occurred with respect to the previous version of SoftOnt. Such changes are communicated to the Broadcaster which will communicate to interested users that SoftOnt has been updated.

[3] The mapping information of the integrated ontology can be generated automatically by combining the mappings of the ontologies that are being integrated [3].

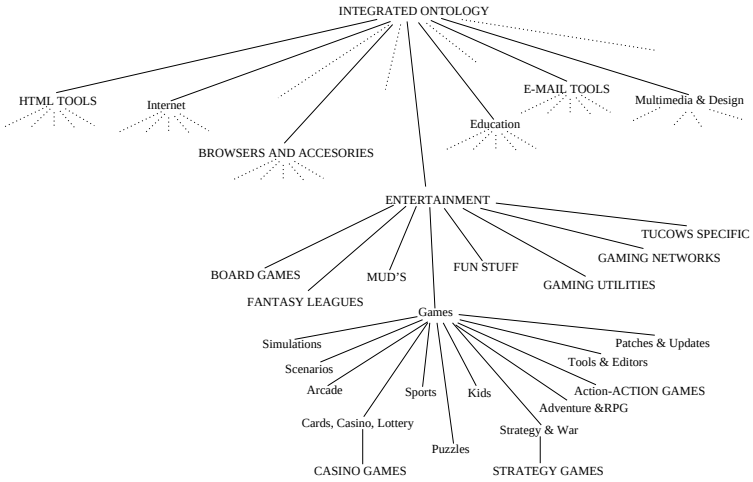

Fig. 7. Integration of Tucows and Download.com ontologies

5.2 Justification for the design of the Knowledge Engineer and Integrator agents

In order to achieve the same goal, someone could think of alternative architectures, including those that do not make use of agent technology, as well as those that advocate for mobile agents to access remote repositories. The arguments for our proposal are the following:

Concerning to the use of agents, in order to 1) translate software repositories into ontologies; and 2) integrate them into a global ontology, the system needs to manage specialized knowledge in an autonomous way. Moreover, the update strategy of the SoftOnt ontology must be independent of the rest of the system, in order to completely isolate the generation of SoftOnt from its exploitation. The previous tasks are too complex for a unique module. Therefore, we advocate for the design of two autonomous agents, the Knowledge Engineer for translation, and the Integrator for integration, which generate some information, an ontology called SoftOnt, that is used by the other agents in the system (which actually ignore when or why SoftOnt is created and updated).

The access to software repositories is one of the most appropriate scenery to use agents. The Knowledge Engineer could create different specialized *mobile agents* that would travel to remote software repositories in order to extract the information right there. This implies a very much faster execution as the HTML pages are accessed locally. This approach is also better with respect to network disconnections. Then, mobile agents could travel back to the Software Acquisition Place to return the extracted information to the Knowledge Engineer. They would be *mobile wrappers*. Unfortunately, the mechanisms needed to receive mobile agents are not very popular yet (for example, Tucows and Download.com sites do not accept incoming agents). Therefore, taking into account the current situation, we thought of local wrappers that access the remote information.

6 Conclusions

In this paper we have presented a multiagent-based application for the creation and exploitation of an ontology in the particular context of software gathering. We can conclude that:

- An ontology is an interesting specification tool to describe software repository contents. Thus, users do not need to deal with several distributed and heterogeneous software repositories.
- The ontology can be created automatically due to context features. Moreover, it can be updated easily to maintain it continuously up to date.
- Agents provide interesting features to create and exploit the ontology, such as autonomy and mobility. They also allow the management of knowledge used to solve problems and take appropriate decisions.

Finally, we can say that our proposed architecture for the Software Retrieval Service, based on agents and one ontology, offers flexibility and adaptability with a low overhead, as shown by our preliminary performance results. We believe that our proposal establishes an interesting trade-off in the use of classical distributed access techniques and new agent technology.

Acknowledgements

We would like to thank Ignacio García for his valuable help in the implementation of the prototype.

References

1. Y. Arens, C.Y. Chee, C. Hsu, and C.A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal on Intelligent and Cooperative Information Systems*, 2(2):127–158, 1993.
2. Attitude Network Ltd., 1999. http://www.gamesdomain.com.
3. J.M. Blanco, A. Goñi, and A. Illarramendi. Mapping among knowledge bases and data repositories: Precise definition of its syntax and semantics. *Information Systems*, 24(4):275–301, 1999.
4. CNET Inc., 1999. http://www.download.com.
5. CNET Inc., 1999. http://www.gamecenter.com.
6. Earthweb & Sun Microsystems, 1999. http://www.gamelan.com.
7. J. Hammer, M. Breunig, H. Garcia-Molina, S. Nestorov, V. Vassalos, and R. Yerneni. Template-based wrappers in the tsimmis system. In *Proceedings of the Twenty-Sixth SIGMOD International Conference on Management of Data, Tucson, Arizona*, May 1997.
8. T. Kirk, A.Y. Levy, Y. Sagiv, and D. Srivastava. The information manifold. In *Proceedings of the AAAI Spring Symposium on Information Gathering in Distributed Heterogeneous Environments, Stanford, CA*, March 1995.
9. Y. Labrou and T. Finin. Yahoo! as an ontology – using yahoo! categories to describe documents. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM99)*, 1999.

10. J.L. Lee, S.E. Madnick, and M.D. Siegel. Conceptualizing semantic interoperability: A perspective from the knowledge level. *International Journal on Cooperative Information Systems (IJCIS)*, 4(4), 1996.
11. A.Y. Levy, A. Rajamaran, and J.J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of the VLDB 96.*, 1996.
12. A.Y. Levy, A. Rajaraman, and J.J. Ordille. Query-answering algorithms for information agents. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI96), Portland, Oregon*, 1996.
13. S. Luke, L. Spector, and D. Rager. Ontology-based Web agents. In *Proceedings of the First International Conference on Autonomous Agents*, 1997.
14. Z. Maamar, B. Moulin, G. Babin, and Y. Bedard. Software Agent-Oriented Frameworks for Global Query Processing. *Journal of Intelligent Information Systems (JIIS)*, 13:235–259, 1999.
15. E. Mena. *OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies.* PhD thesis, University of Zaragoza, November 1998. http://siul02.si.ehu.es/PUBLICATIONS/thesis98.ps.gz.
16. E. Mena, A. Illarramendi, and A. Goñi. Customizable Software Retrieval Facility for Mobile Computers using Agents. In *proceedings of the Seventh International Conference on Parallel and Distributed Systems (ICPADS'2000), workshop International Flexible Networking and Cooperative Distributed Agents (FNCDA'2000), IEEE Computer Society, Iwate (Japan)*, July 2000.
17. E. Mena, A. Illarramendi, V. Kashyap, and A. Sheth. Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *International journal on Distributed And Parallel Databases (DAPD)*, 8(2), April 2000.
18. E. Mena, V. Kashyap, A. Illarramendi, and A. Sheth. Imprecise answers on highly open and distributed environments: An approach based on information loss for multi-ontology based query processing. Accepted for publication in a special issue of the International Journal of Cooperative Information Systems (IJCIS), 2000.
19. G. Miller. World Wide Web interface to WordNet 1.5, June 1995. http://www.cogsci.princeton.edu/~wn/w3wn.html.
20. D. Milojicic, M. Breugst, I. Busse, J. Campbell, S. Covaci, B. Friedman, K. Kosaka, D. Lange, K. Ono, M. Oshima, C. Tham, S. Virdhagriswaran, and J. White. MASIF, the OMG mobile agent system interoperability facility. In *Proceedings of Mobile Agents '98*, September 1998.
21. Y. Papakonstantinou, A. Gupta, H. Garcia-Molina, and J. Ullman. A query translation scheme for rapid implementation of wrappers. In *Proceedings of the International Conference on Deductive and Object-Oriented Databases*, 1995.
22. A.P. Sheth, S.K. Gala, and S.B. Navathe. On automatic reasoning for schema integration. *International Journal on Intelligent and Cooperative Information Systems*, 2(1):23–50, 1993.
23. S. Spaccapietra, C. Parent, and Y. Dupont. Model independent assertions for integration of heterogeneous schemas. *VLDB*, 1:81–126, 1992.
24. Tucows.Com Inc., 1999. http://www.tucows.com.
25. Yahoo! Inc. http://www.yahoo.com/.