Designing Agent Based Travel Support System

Minor Gordon
Department of Computer Science
Technische Universität Berlin
Berlin, Germany
minorg@cs.okstate.edu

Marcin Paprzycki
Department of Computer Science
OSU, Tulsa, OK, USA
and
Computer Science, SWPS
Warsaw, Poland
marcin@cs.okstate.edu

Abstract

Online travel support systems have often been cited as an ideal proving ground for agent-based architectures, yet no working systems have materialized. Over the last few years we have been considering various aspects of the design of such travel support systems. The aim of this note is to present the current state of our comprehensive framework for delivering personalized travel services using agent infrastructure.

1. Introduction

Today the potential traveler has access to an unprecedented wealth of travel-related information available on the Internet yet planning actual trips using only Internet based resources requires a lot of work and sometimes turns out to be rather difficult. Potential traveler must devote a lot of effort sifting through many sites of varying quality of data simply to form a coherent picture of his intended destination, possible means of transportation, etc. The total amount of data is so large that it is impossible to find *all* pertinent and important information in a reasonable time.

In the last 15 years there have been numerous attempts to apply the software agent paradigm to travel services and the pervasive problem of information overload [26, 28, 30]. The analogy of software agents to travel agents makes the analogy between real world travel agencies and online agent platforms seem trivial, yet the majority of the proposed systems never left the drawing board. The few active experiments in travel-related agent architectures we have discovered have either been limited in scope [29, 34, 40] or abandoned. The CRUMPET project is a typical example of what happens with projects aiming at applying agents in travel support systems. CRUMPET was funded by the EU FP5 umbrella between 1999 and 2003, but as of April 2005 it is almost impossible to assess its achievements, because its WWW site no longer exists and information about the project itself is spread across a few auxiliary sites and conference papers that resulted from it [34]. Since 1999-2001 a large number of researchers that were originally interested in agent-based systems have moved on to more fashionable ventures such as the Semantic Web or Ambient Computing, leaving behind a trail of papers and unfinished, only partially implemented designs. We have also been trying to develop such a system since 2001, so we can easily understand why these groups have moved on rather then complete a working prototype. Working with existing agent systems is a struggle with cutting edge, constantly evolving and highly unreliable technologies. The design of a system that seems to be quite reasonable one day may become infeasible and/or obsolete sometime before the initial implementation is completed. For instance, we have run into this problem when experimenting with data indexing and the ebXML Registry Repository [8]. After many futile attempts we were forced to acknowledge that the existing implementation of the repository was unreliable and incapable of handling "real size" load and had to abandon it, after nine moths of work invested in making the repository a centerpiece of our design [16, 21, 41].

The aim of this paper is to present some of the lessons we have learned in designing our agent-based travel support system, and to outline an evolved version of the design. The current system stores semantically demarcated data in a central repository (data gathering rather than the indexing we originally envisioned [16]). While the majority of today's Internet-based travel services focus on transportation and lodging, with an emphasis on transactions, our system is designed to deliver an extended travel itinerary, including the standard transportation and accommodation choices as well as restaurants, movie theaters, national parks, historical sites and other points of interest, any of which may be selected by the user from an array of options composed specifically for him/her (content personalization). The system is accessible via Internet-enabled devices, ranging from standard PC-based browsers to palmtops and WAP-conversant phones etc., and even non-human entities (such as other agents) [11].

In this paper we devote our attention to the high level description of the agent system and omit (and assume to be "successfully addressed") a number of important questions:

(a) economic model – how such a system will generate revenue for the company that implements it (see e.g. [3, 4, 7, 19, 27, 39]),
(b) user profiling and clustering (in the context of RFM analysis and cluster analysis) to discover and modify customer segments (see e.g. [10, 36, 37, 38, 42]),
(c) methodologies for data mining and modeling in the context of content delivery personalization (see e.g. [36, 37]),
(d) personalized advertisement targeting (see e.g. [3, 4, 6, 10]),
(e) dealing with conflicting information and, more generally, validating information from unverified Internet sources (see e.g. [33]).

We proceed as follows: in Section 2 we present the general architecture of the system and briefly sketch its functions; Section 3-5 contains the descriptions of the content collection, content management and content delivery subsystems. In Section 6 we describe a scenario that illustrates how user will interact with the system. We conclude with a brief summary of the current state of the project.

2. System Architecture

The overall architecture of the proposed system is depicted in Figure 1. Before proceeding, let us make two comments: (1) The proposed system belongs to the class of infomediaries and therefore its development will proceed within the framework presented in [19]; (2) The general system structure is a modification of the skeleton of the general e-commerce system presented in [9]. Here, instead of dividing the complete system into two subsystem-spheres: supply and delivery, connected through a communication channel, we are introducing three sub-areas: content collection, content management and content delivery, with the content management subsystem together with the central data repository becoming the centerpiece of the system. Let us now briefly summarize each of the components presented in Figure 1.

Verified Content Providers (VCP)

Today, a very large number of web sites provide some form of travel-related information. However, as pointed out in [28], there exist a few serious potential problems arising from the dynamic nature of content on the Internet. Information relevant to travel is stored in a large number of small, independently operated Web sites (consider all restaurants, bars and independently owned hotels that have their own private websites). Since many of these sites are hosted by small local ISP's that may not be able to provide sufficient quality of service, these sites may migrate and, as a result, change their URL's. In this case information that was once available (its location was known) cannot be found easily. The second situation involves progress in Web site design. Due to technological change (e.g. switching from plain HTML to PHP) or overall change in what is perceived to be state of the art in website design, the layout of a given site may change completely, such that agents that were collecting information from that site are no longer able to access it without errors. To deal with this problem and the more general issues of accuracy and relevancy we utilize the concept of *Verified Content Providers* [1].

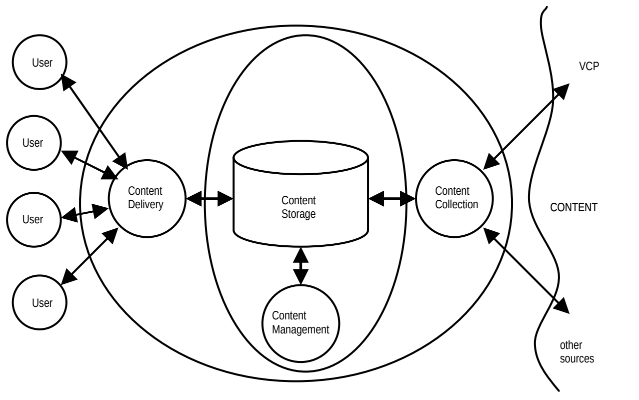


Figure 1. Infrastructure for the travel support system

Conceptually, a *Verified Content Provider* (*VCP*) is a site that is known to provide **reliable** and **consistently available** information. We assume that *VCP*s do not randomly appear and disappear from the Internet. Furthermore, *VCP*'s are expected to maintain the same site interface for extended periods of time. It can be even assumed that in the case of a commercial travel support system, *VCP*s will be providers with whom a contract is signed and thus if the interface changes "we" will be informed about it beforehand. Finally, *VCP*s are sites that constantly provide reliable information and thus can be trusted. While the category of *VCP*s is used to mark "the best of breed" information sources, the remaining sources available on the Internet can be assigned varying levels of trust and that level of trust can dynamically change over time. (e.g. a *Verified Content Provider* may cease being *verified* if it no longer meets the criteria of trustworthiness while a different source may become a *VCP* by systematically delivering high quality information). Delving into details of designing such an adaptive trust system is outside scope of this paper.

*VCP*s can be divided into two groups: (i) pushing sources – those that provide content for our system in a standard form (such as RSS feeds), and (ii) static – sites that require agents to collect pages periodically and extract the necessary information. It is important to stress (since we are interested in collecting, storing, and processing semantically demarcated data) that at this stage of development of the Internet, existence of any form of *VCP*s containing travel-related semantically demarcated content (except of academic demonstrator systems of minimal breadth and depth of available data; and the ChefMoz [5] site containing RDF demarcated data or reasonable, but not fully machine consumable quality [12]), while highly desirable, is just a wish that we hope one day will come true.

Other Sources
There exists a number of problems related to dealing with *unverified*, unstructured Internet-based information: (a) its amount, which makes an exhaustive search practically impossible, (b) the unreliability of data, and (c) contradicting sources that require application of sophisticated data deconfliction techniques. While we hope that the approach of relying primarily on trusted data providers (*VCP*s) will alleviate most of these problems (see [1] for more details), we still should not discard additional information available on the Internet and we will utilize it whenever possible. This approach will become even more important when the idea of the Semantic Web will start to take hold and semantically demarcated information available in small-independent sites will be much easier to process automatically.

Content Collection Subsystem (CCS)
In the proposed approach, the information provided by or collected from the *Verified Content Providers* and from other Internet sources is stored in a semantically demarcated form. We have selected RDF as the ontology tagging "language." For storing RDF triples we have decided to use the Hewlett-Packard Jena system [24]. Our *Content Collection System (CCS)* stores sets of RDF triples that in aggregation represent travel objects (hereafter referred to as *tokens*). Travel object tokens may originate either from a direct feed from *VCP*s or from an agent subsystem that collects data from the Internet and are stored in the central repository. While the question of the size and centralized nature of the repository will have important consequences to the scalability of the system, these questions are outside of the scope of this paper and will have to be addressed only in the future (when the system reaches such size that the performance of the repository will become the true performance bottleneck for the whole system). In Section 3 we present details of the functions and structures that comprise the *CCS* subsystem.

Content Management Subsystem (CMS)
This subsystem includes all functions related to the data stored in the central repository (Jena database). Observe that at least two cases of possibility of data inadequacy have to be taken into account. First, we have to deal with incomplete data items. For instance, an object to be stored in the repository may contain triples for the name, address and web site of a restaurant, but not the phone number (or some other information defined by our restaurant ontology [13]). In this case such an object is incomplete and we must attempt to find the missing information. The second case involves data that is stored in the system and that is in some way time sensitive, e.g. cinema programs change on Fridays, and thus they have to be updated. Obviously, even information that is not explicitly time-sensitive (e.g. Museum opening times) has to be re-checked in some unspecified time intervals. In Section 4 we discuss this subsystem in more detail.

Content Delivery Subsystem (CDS)
Here the data stored in the central repository is manipulated for delivery to end users. The agents in this subsystem obtain a query from the user and work with the CMS to find information matching the user's personal preferences. Data presented to the user may be acquired from the repository, or result form additional Internet searches. We devote Sections 5 and 6 to describing this subsystem in detail.

Users
The system will be accessed via Internet-enabled devices, ranging from standard PC-based browsers to palmtops and WAP-conversant phones etc., and even non-human entities (such as other agents) [12, 13]. Some details describing how we are implementing user-system communication can be found in Section 6.

3. Content Collection Subsystem

The structure of the content management subsystem is presented in Figure 2.

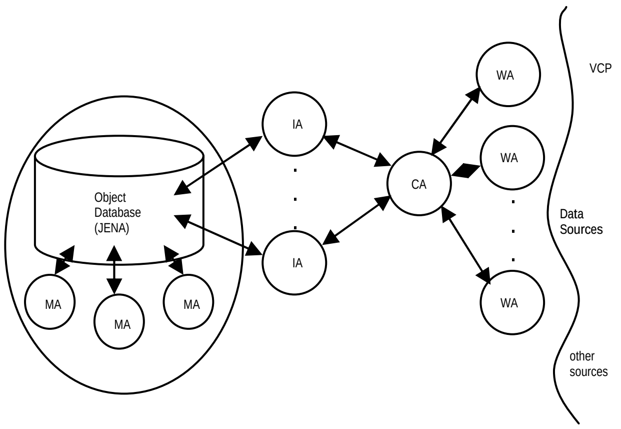


Figure 2. Content management subsystem: WA – wrapper agents, CA – coordinator agent, IA – indexing agents, MA – management agents

Recall that the central repository of our system stores RDF triples – semantically demarcated data tokens. Because currently there is very little RDF-demarcated data natively available on the Internet, we must extract HTML content from existing web sites into RDF triples "manually." Our Content Collection subsystem uses *Wrapper Agents* (*WA*) that interface with various WWW sites, mapping XML- or HTML-demarcated data into RDF triples describing travel objects (according to an appropriate ontology used in our system [12]). Sets of triples gathered by Wrapper Agents, and assembled into travel tokens, are sent to a *Coordinator Agent* (*CA*), which schedules Wrapper Agents and relays the resulting triples to the Jena database. Earlier we suggested that some of the *VCP*s may, sooner or later, contain and/or start delivering RDF demarcated content directly to the system. In this situation all we have to do is to adjust some of the *WA*s to be able to handle these particular inputs, while the remaining parts of the *CCS* remain the same (this illustrates the power of agent-based system design, where changes need to be incorporated only into specific groups of agents and remain localized to them). An interesting question arises when ontologies (e.g. our hotel ontology [14]) available on the Internet do not match our custom ontologies. In this case, the ontology matching and resolving techniques have to be applied, but this process remains outside of the scope of current paper. Here we assume that in this case selected *Wrapper Agents* will have to take role of the ontology resolvers, working on achieving this goal on a case by case basis. In general we assume a collection of *WA*s capable of interfacing with an assortment of data sources and delivering to the *CA* RDF demarcated tokens describing various travel resources. Communication between *WA*s and the *CA* occurs through exchange of ACL messages (RDF triples are serialized and send as a content of ACL Inform messages).

All incoming information is received by the *CA*. Its primary role is to act as a large priority queue, where all data objects will be temporarily stored (since in an agent system one cannot have a "free floating queue," but any such data object has to be encapsulated in an agent). Obviously, having only one *CA* may become a bottleneck and in this case having additional *CA*s may alleviate the problem (however, also this purely technical problem is outside of the scope of this paper). The *CA* prioritizes the data in the queue. This is done based on the answer to the questions: why is given data taken brought to the system and when it will be needed? If the data token is a result of web crawling, it will be assigned "basic" priority. If the token is a result of time-oriented trigger event (for instance that the theater program is changing) then it will be assigned an "elevated" priority (it is quite possible that someone will soon request this information and thus it has to be inserted into the system as soon as possible). Finally, tokens resulting from the "user queries" will be assigned "highest" priority. Here the assumption is that user is still online and necessary information has to be delivered as soon as possible.

Information is inserted into the repository by a pool of *Indexing Agents* (*IA*). These agents request from the *CA* (via an ACL message) the next data token to work on and obtain it wrapped in an ACL message. *IA*s check the completeness of data. There are many situations when the data tokens may be deficient (e.g. hotel info misses information about available amenities). In this case, before insertion into the repository, data tokens are marked as incomplete. However, they may still be used by the content delivery subsystem to deliver response to the user (especially when these tokens are widely queried, or they are the only available content pertaining to a given query). Finally, in the case of a token replacing another token and the two tokens containing conflicting information, both tokens are left in the system and marked accordingly for deconfliction to take place.

4. Content Management Subsystem

In our earlier work [2, 9], the content management subsystem included both its current functions and the content collection functions described above. After our implementation experiences we decided to change this and separate the collection and the management functions. Content management involves all the agents that operate on data stored within the repository. In this way we separate these agents and functions form the rest of the system, with which the communication occurs only via ACL messages.

Thus far we have indicated three possible functions to be performed by the *CMS* agents. The first is related to the completeness of tokens. Incomplete tokens will be marked as such by the *IA*. *CMS* agents will traverse the repository to find incomplete tokens. They will then formulate queries to be answered and request (via an ACL message) from the *CA* that appropriate *WA* be released to search for the missing information.

The second situation involves tokens containing conflicting information. They are marked as such by the *IA* and left for the deconflicting agents to deal with. Deconfliction may involve additional queries to the Internet as well as consideration of factors such as, the freshness of the older data, reliability of the sources etc.

The third situation deals with time-sensitive data. There is a large amount of travel-related information on the Internet that changes in regular intervals (e.g. programs of operas, theaters, cinemas, etc.). It is possible, for each of these situations, to establish proper time to find updated information. We assume here that the database will generate triggers that will result in agents involved in management of time-sensitive data to communicate with the *CA* to request an update of a given token. However, we also must recognize that all of data available in the system is time sensitive. Even content that seems to be relatively "stable," like the restaurant menu or ZOO opening times change from time to time. Therefore each data item will have a "time stamp" describing when it was created. After a specified time (different for different travel objects), the database will generate a trigger that will start the update function. Here one of the adaptation mechanisms available in the system will take place. When the update request does not result in any changes, the time length before the next update will be increased, while the request resulting in change will cause the update time to be shortened.

Obviously, there may exist other data management functions involving data in the central repository. The clear advantage of the agent system is that in such a case the only required function is to add new agent type that will perform the required functions [25].

5. Content Delivery Subsystem

The content delivery subsystem is responsible for answering user queries. Here the primary challenge is communication between clients and agents in our system. While in theory this should be relatively straightforward, in our earlier work [112] we discovered that this is not the case. Specifically, the absence of agent platforms to host agents on client devices limits the reach of the agent platform to protocols supported by the client, and those only to the extent that the agent system may be adapted to them. Thus we make only a minimal set of assumptions about expected capabilities of the device (like that it will be able to communicate with the Internet using the HTTP protocol). Furthermore, all the computational power and support for communication has to be on the server side. Our initial attempt to solve the problem [12] employed the Mozilla XUL and XUP languages for platform-independent user interfaces. Recently we changed our approach to this problem [13], and our current design will be described detail in the next Section. Here we will proceed with the assumption that the user has submitted a query through an Internet enabled device to his *Personal Agent* (*PA*) (see Figure 3). This message has been translated from its original form by the *Proxy Agent* (*PrA*) residing on the "gateway HTTP server." Regardless of the form of the original query the "query content" is extracted and wrapped into an ACL message and in this way send to the *PA*. The *PA* forwards the message to two locations. First, to the group of agents responsible for content personalization. This message is stored in the user behavior database, where information about all interactions between the user and the system is logged. More precisely, all user queries sent to the system and all system responses are logged for future mining [10]. Based on the queries and responses and user responses to the queries it is possible to establish a profile that can be used by the *PA* to filter and personalize content delivered to the user [15]. Second, the query is send to the *DB Agent* (*DBA*). The *DBA* is the interface of the system to the Jena database. The *DBA* translates the user query into the *RDQL* language (the database language used by Jena to query the entire set of RDF triples). The *DBA* executes the query and as a result obtains a set of tokens describing one or more travel objects. These RDF demarcated tokens are then sent to the personalization infrastructure.

The personalization infrastructure consists of a number of "RDF Agents." These are simplistic agents. Each represents one of more of simple rules of the type "Szechuan food is also Chinese food" or "Romantic Comedy is also a Comedy." These rules are applied to the set of RDF triples returned by an RDQL query. Rule applications may involve querying the repository and expand the result set. The personalization infrastructure agents operate as a team passing the result set, wrapped in an ACL message from one to the next and their role is to maximize the set of responses to be delivered to the user (no potential response is removed form the set).

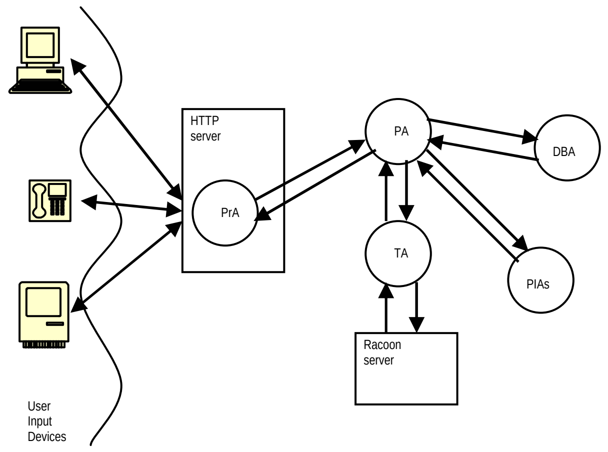
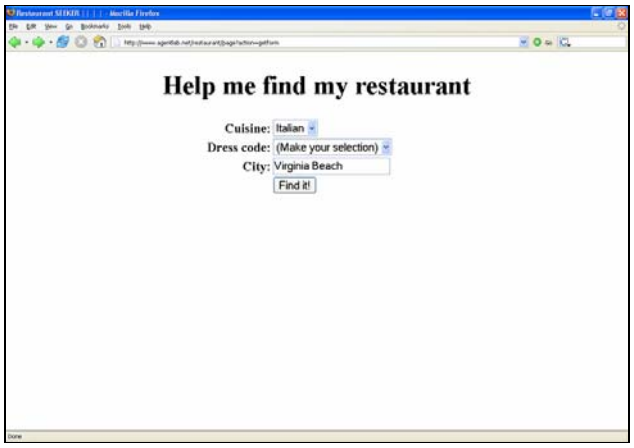


Figure 3. Content delivery subsystem: PA – personal agent, DBA – database agent, PIA – personalization infrastructure agents, PrA – proxy agent, TA – transformation agent

The maximal set of responses is sent back to the *PA*. The *PA* utilizes the user profile filter the answer set. For instance, the *DBA* and the *PIA* agents may not know that the user never stays in Howard Johnson hotels and never visits Braum's restaurants. Thus RDF triples representing these two chains will be included in the expanded answer set. However, the *PA* will remove them form the set. The answer set is send to the *Transformation agent* (*TA*) that utilizes a Racoon [35] server to render the response to be displayed on user device (as well as back to the personalization infrastructure to be logged as a response to the given query).

6. Usage scenario

Let us now consider a usage scenario that will illustrate in detail how the data flow in the system is handled (this description should be matched with subsystem depiction presented in Figure 3). We assume here that initial data is already stored in the system. Let us assume now that user fills a form (an extremely simplified test-form) to find a restaurant in Virginia Beach (Figure 4).



Help me find my restaurant

Cuisine: Italian
Dress code: (Make your selection)
City: Virginia Beach
Find it!

Figure 4. User form to find a restaurant.

The target of this query is the following restaurant, which is described in our Jena database with a set of RDF triples (here shown in the N3 syntax – note that some lines have been wrapped because of the narrowness of the two-column format):

```
:United_States_VA_Virginia_Beach_Bella_Mont
e954313245
    a res:Restaurant;
    res:title "Bella Monte";
    res:id
    "United_States_VA_Virginia_Beach_Bella_M
    onte954313245";
    res:locationPath
    "United_States_VA_Virginia_Beach";
    res:link
    "http://digitalcity.com/hamptonroads/din
    ing/venue.dci?vid=81334".
    loc:streetAddress "1201 Laskin Rd.";
    loc:city "Virginia Beach";
    loc:country "United States";
    loc:phone "757.425.6290";
    loc:state "VA";
    loc:zip "23451";
        res:description "Lunch and dinner
    served Monday through Saturday.";
        res:alcohol alc:FullBar;
        res:dress drs:Casual;
        res:reservations rsv:Recommended;
        res:cuisine cui:Italian;
        res:cuisine cui:Regional;
```

When the user clicks the “Find it!” button of the form an HTTP request is sent to our *Proxy Agent* (*PrA*), which translates the CGI query string

```
http://www.agentlab.net/restuarant/page?action=ge
tdata&cuisine=Italian&dress=&city=Virginia+Beach
```

into a temporary form:

```
     [[cuisine{{Italian[[city{{Virginia Beach
```

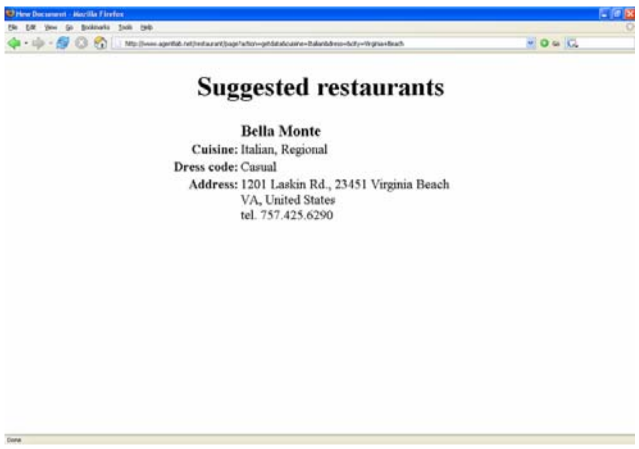
which is then sent in an ACL message to the *Personal Agent* (*PA*). The *PA* forwards this ACL message in turn to the personalization infrastructure for logging and to the *Database Agent* (*DBA*). The *DBA* transforms the keywords of the query into the following RDQL string:

```
SELECT
 ?res
WHERE
    (?res, <res:cuisine>, <cui:Italian>),
    (?res, <loc:city>, "Virginia Beach")
USES
    res for
<http://www.agentlab.net/schemas/Restaurant#>,
    cui for
<http://www.agentlab.net/schemas/CuisineCode#>,
     loc for
<http://www.wam.umd.edu/~krakatoa/cs828y/project/
travel.daml#>
```

This RDQL query is executed and matching RDF triples are serialized to an RDF/XML document, fragment of which has the following form:

```
<res:Restaurant
rdf:about="&res;United_States_VA_Virginia_Beach_B
ella_Monte954313245">
    <res:title>Bella Monte</res:title>
    <res:description>Lunch and dinner served
    Monday through Saturday.</res:description>
    <res:dress rdf:resource="&drs;Casual"/>
    <res:reservations
    rdf:resource="&rsv;Recommended"/>
    <res:alcohol rdf:resource="&alc;FullBar"/>
    <res:cuisine rdf:resource="&cui;Italian"/>
    <res:cuisine rdf:resource="&cui;Regional"/>
    <loc:city>Virginia Beach</loc:city>
    <loc:country>United States</loc:country>
    <loc:phone>757.425.6290</loc:phone>
    <loc:state>VA</loc:state>
    <loc:streetAddress>1201 Laskin
    Rd.</loc:streetAddress>
    <loc:zip>23451</loc:zip>
</res:Restaurant>
```

In a full fledged system this initial response would then be sent to the personalization infrastructure that would analyze it and try to expand the number of response-tokens. For instance, if there was a movie theater near by the location of the restaurant then a token for the theater could be then added to the answer set. The complete set of answer-tokens is then sent back to the *PA*. The *PA* could add additional tokens (like utilize knowledge that user smokes cigars and that there is a Cigar Shop close to the restaurant) or remove some tokens (movie theater plays movies that are not likely to attract this particular user' attention). Finally, the *PA* sends the filtered answer-set to the personalization infrastructure agents for logging (to obtain a query-response pair) and via the *TA* to the Racoon server for it to be rendered (in our case to HTML). The result is then forwarded back to the *PA* that sends it to the *PrA* and then it is finally forwarded to the browser, where it could look like a response presented in Figure 5.


Suggested restaurants

Bella Monte
Cuisine: Italian, Regional
Dress code: Casual
Address: 1201 Laskin Rd., 23451 Virginia Beach
VA, United States
tel. 757.425.6290

Figure 5. Response form the system.

While the depiction presented there is extremely simplistic it was obtained in the process of actual interaction with the system. In other words, the above described data-flow has been implemented and actually works (see also [13]).

7. Concluding Remarks

In this note we have outlined the high-level architecture of an agent-based travel support system. In pursuing the agent framework we have described the most important classes of agents in our system, their respective functions and the relationships between them. We have also presented a usage scenario illustrating the dataflow of a user query and illustrated it on the basis of an existing implementation of this functionality of the system under development.

We are currently in the process of implementing most of the key parts of the system. We are using JADE agent environment [6], Jena repository for the RDF demarcated data [24], and Racoon for rendering responses for variety of devices [35]. We will be using JESS for providing expert system capabilities in the system. In support of our system we have developed ontology of a hotel and re-engineered restaurant ontology based on the ontology implicitly underlying the ChefMoz project [5]. We continue locating and designing further ontologies needed in our system. We have also developed the first collection of Wrapper Agents that will be used to populate our data repository.

We will report on progress of implementation in the near future.

References

[1] Abramowicz W., Kalczyński P. J. (2002) "Building and Taking Advantages of the Digital Library for the Organizational Data Warehouse," in: Cobb M. et. al. (eds.), Proceedings of the Second Southern Conference on Computing, Hattiesburg, Mississippi, USA, October 26-28, 2000, CD, 8 pages.

[2] Angryk R., Galant V., Gordon M., Paprzycki M., "Travel Support System – an Agent Based Framework," in: H. R. Arabnia, Y. Mun (ed.), Proceedings of the International Conference on Internet Computing (IC'02), CSREA Press, Las Vegas, NV, 2002, 719-725

[3] Brady R., Forrest E., Mizerski R. (2002) "Marketing w Internecie," PWE, Warszawa, Poland

[4] Chaudhury A., Malik D. N., Rao H. R. (2001) "Web Channels in E-commerce," CACM, 44 (1), pp. 99-104

[5] Chefmoz Dininig Guide, http://chefmoz.org/

[6] Chmiel K., Tomiak D., Gawinecki M., Kaczmarek P., Paprzycki M., Szymczak M., "Testing the Efficiency of JADE Agent Platform," with in: Proceedings of the ISPDC 2004 Conference, IEEE Computer Society Press, Los Alamitos, CA, 2004, 49-57

[7] de Kare-Silver, M. (2001) "E-shock: the New Rules. E-Strategies for Retailers and Manufacturers," Palgrave, Houndmills, UK

[8] ebXML R/R: http://ebxmlrr.sourceforge.net/

[9] Galant V., Jakubczyc J., Paprzycki M. (2002) "Infrastructure for E-Commerce," Proceedings of the 10th Conference on Extraction of Knowledge from Databases, Karpacz, Poland, May, 2002 (to appear)

[10] Galant V. and Paprzycki M. (2002) "Information Personalization in an Internet Based Travel Support System," Proceedings of the BIS'2002 Conference, Poznań, Poland, April, 2002, pp. 191-202

[11] Galant V., Gordon M., Paprzycki M., "Knowledge Management in an Internet Travel Support System," in: B. Wiszniewski (ed.), Proceedings of ECON2002, ACTEN, Wejcherowo, 2002, 97-104

[12] Galant V., Gordon M., Paprzycki M., "Agent-Client Interaction in a Web-based E-commerce System," in: D. Grigoras (ed.), Proceedings of the International Symposium on Parallel and Distributed Computing, University of Iaşi Press, Iaşi, Romania, 2002, 1-10

[13] Gawinecki M., Gordon M., Kaczmarek P., Paprzycki M., "The Problem of Agent-Client Communication on the Internet," Parallel and Distributed Computing Practices (to appear)

[14] Gawinecki M., Gordon M., Paprzycki M., Szymczak M., Vetulani Z., Wright J., "Enabling Semantic Referencing of Selected Travel Related Resources," Proceedings of the BIS'2005 Conference, Poznań University of Economics Press, Poznań, Poland, 271-288

[15] Gawinecki M., Vetulani Z., Gordon M., Paprzycki M., "Representing Users in a Travel Support System," in: Proceedings of the Intelligent Systems Design and Applications, Wrocław, Poland, September, 2005, (to appear)

[16] Gilbert A., Gordon M., Nauli A., Paprzycki M., Williams S., Wright J., "Indexing Agent for Data Gathering in an e-Travel System," Informatica, Vol. 28, No. 1, 2004, 69-78

[17] Gilbert A., Gordon M., Paprzycki M., Wright J., "The World of Travel: a Comparative Analysis of Classification Methods," Annales UMCS Informatica, A1, 2003, 259-270

[18] Gordon M., Gilbert A., Paprzycki M., "Knowledge Representation in the Agent-Based Travel Support System," in: T. Yakhno (ed.) Advances in Information Systems, Springer-Verlag, Berlin, 2002, 232-241

[19] Grover V., Teng J. C. T. (2001) "E-commerce and the Information Market," CACM, 44(4), pp.79-86

[20] Grunninger M., Lee J. (eds.) (2001) "Ontology, Applications and Design," Special Section of the CACM, 45(2), pp. 39-65

[21] Harrington P., Gordon M., Nauli A., Paprzycki M., Williams S., Wright J., "Using Software Agents to Index Data in an E-Travel System," in: N. Callaos (ed.), Electronic Proceedings of the 7th SCI Conference, Orlando, 2003, CD, file: 001428.pdf, 6 pages

[22] Heilmann K., Kihanya D., Light A., Mousembwa P. (1995) "Intelligent Agents: A Technology and Business Application Analysis," Technical Report, University of Vienna, BA248D

[23] Jakubczyc J., Gordon M., Galant V., Paprzycki M., "Knowledge Management in an E-commerce System," Proceedings of the Fifth International Conference on Electronic Commerce Research, Montreal, Canada, October, 2002, CD, 15 pages

[24] Jena 2 – A Semantic Web Framework, Hewlett Packard, http://www.hpl.hp.com/semweb/jena2.htm

[25] Jennings N. R. (2001) "An agent-based approach for building complex software systems," CACM, 44 (4), pp. 35-41

[26] Maes P., "Agents that Reduce Work and Information Overload," Communications of the ACM, 37(7), 1994, 31-40

[27] Mohr J. (2001) "Marketing of High-Technology Products and Innovations," Prentice-Hall, Upper Saddle River, NY

[28] Nwana H., Ndumu D. (1999) "A perspective on software agents research," The Knowledge Engineering Review, 14 (2), pp. 1-18

[29] Ndumu, D., Collins, J., Nwana, H. (1998) "Towards Desktop Personal Travel Agents," BT Technological Journal, 16 (3), pp. 69-78

[30] Paprzycki M., Abraham A., "Agent Systems Today; Methodological Considerations," in: Proceedings of 2003 International Conference on Management of e-Commerce and e-Government, Jangxi Science and Technology Press, Nanchang, China, 2003, 416-421

[31] Paprzycki M., Angryk R., Kołodziej K., Fiedorowicz I., Cobb M., Ali D. and Rahimi S. (2001) "Development of a Travel Support System Based on Intelligent Agent Technology," in: S. Niwiński (ed.), Proceedings of the PIONIER 2001 Conference, Technical University of Poznań Press, Poznań, Poland, pp. 243-255

[32] Paprzycki M., Kalczyński P. J., Fiedorowicz I., Abramowicz W. and Cobb M. (2001) "Personalized Traveler Information System," in: Kubiak B. F. and Korowicki A. (eds.), Proceedings of the 5th International Conference Human-Computer Interaction, Akwila Press, Gdańsk, Poland, pp. 445-456

[33] Petry F., Cobb M., Ali D., Angryk R., Paprzycki M., Rahimi S., Wen L., Yang H. (2002 to appear) "Fuzzy Spatial Relationships and Mobile Agent Technology in Geospatial Information Systems," in: Sztandera L., Matsakis P. (eds.) Soft Computing in Defining Spatial Relations, volume in series: Soft Computing, Physica-Verlag, to appear

[34] Poslad S., Laamanen H., Malaka R., Nick A., Buckle P., Zipf A. (2001) CRUMPET: Creation of User-friendly Mobile Services Personalised for Tourism. Proceedings of: 3G 2001 - Second International Conference on 3G Mobile Communication Technologies. 26-29 March 2001. London, UK. http://conferences.iee.org.uk/3G2001/

[35] Racoon: http://rx4rdf.liminalzone.org/Racoon

[36] Rud O. P. (2001) "Data Mining Cookbook. Modeling Data for Marketing, Risk, and Customer Relationship Management," Wiley, New York, NY

[37] Simon A. R., Shaffer S. L. (2001) "Data Warehousing and Business Intelligence for E-commerce," Morgan Kaufman, New York, NY

[38] Sołtysiak S., Crabtree B. (1998) "Automatic learning of user profiles – towards the personalization of agent service," BT Technological Journal, 16 (3), pp. 110-117

[39] Strauss J., Frost R. (2001) "E-Marketing," Prentice-Hall, Upper Saddle River, NY

[40] Suarez J. N., O'Sullivan D., Brouchoud H., Cros P. (1999) "Personal Travel Market: Real-Life Application of the FIPA Standards." Technical Report, BT, Project AC317

[41] Wright J., Gordon M., Paprzycki, M., Williams S., Harrington P., "Using the ebXML Registry Repository to Manage Information in an Internet Travel Support System," in: W. Abramowicz and G. Klein (eds.), Proceedings of the BIS'2003 Conference, Poznań University of Economics Press, Poznań, Poland, 2003, 81-89

[42] Zhang B., Li W., Xu Z. (2002) "Personalized Tour Planning System Based on User Interest Analysis," Proceedings of the BIS'2002 Conference, Poznań, Poland, April, 2002, pp. 184-190