Using the ebXML Registry Repository to Manage Information in an Internet Travel Support System

Jimmy Wright
Oklahoma State University
jimmyww@cs.okstate.edu

Minor Gordon
Oklahoma State University
minorg@cs.okstate.edu

Marcin Paprzycki
Oklahoma State University
marcin@cs.okstat.edu

Steve Williams
Oklahoma State University
skw@cs.okstate.edu

Patrick Harrington
Northeastern State University
harringp@nsuok.edu

Abstract

An Internet-based travel support system requires an efficient means of managing travel-related information both inside and outside the confines of the system, in order to present the most accurate and relevant travel choices to the end user. In this note, we describe a configuration for achieving this goal by employing an ebXML Registry/Repository system for cataloguing travel information from the Internet.

1. Introduction

In the past decade the travel services market has developed a hugely diverse presence on the Internet, in terms of both resources offered (hotel rooms, rental cars, dinner reservations, golf tee times, “general tourist information,” etc.) and approaches to offering them (e.g. aggregation, personalization, mobile delivery). Unfortunately, as in other domains, the potential travel services user must often deal with one of the crucial problems inherent in information diversity: the lack of an encompassing catalogue through which the content of interest may be located. Internet search engines usually provide only a non-categorized and mostly non-intuitive means of locating and representing data, and search results in the travel (as well as any other) domain are likely to include more unrelated hits than relevant travel choices. The recent addition of the Google and Yahoo directories are representative early attempts to organize the presentation of many types of data including travel data, however, they provide no organized booking interface for the data they offer. In addition, the Google directory consists of a mixture of travel resource types and geographical categories. On the other hand, some of the major travel sites such as Expedia, Travelzoo, etc. do apply a degree of organization to a limited subset of travel data (typically limited to airline, car, hotel reservation as well as cruise and vacation package arrangements), based on content stored in tailor-made databases within the system. Here, the mass of information stored on independent Internet sites is ignored. Thus, we believe that neither search engines nor the large travel sites are currently capable of providing a complete support to modern day traveler.

Ideally, a travel support system should act as a filtering and organizing intermediary between travel consumers and travel suppliers [15]. Its primary function [1] is to find the travel information that is most relevant to the customer and deliver it in a well-organized and intuitive way [2]. In order to support this (content delivery) role the system must explore the Internet and other sources to dynamically construct and manage a supply of travel content from known and previously unknown providers [1, 3, 17].

In exploring the potential of such a travel support system, we have followed a two-pronged approach. First, since travel support is a paradigmatic example of the application of agent technology [14, 16], we have decided to utilize software agents for the framework of our system [1, 6]. Second, as an information broker between travel content suppliers and end users (travelers) we must carefully consider the means by which we will structure the information within the system, in order to deliver the most relevant and accurate travel choices to the consumer [2, 5, 17]. We believe that one of the more promising approaches to structuring information from diverse sources is to apply index-based techniques similar to those described in [13] (and references available there). This approach allows us to effectively deal with data available from multiple sources across the Internet in such a way that pertinent information may be efficiently and accurately selected and delivered to consumers.

In the following sections we focus on the issues surrounding the design and implementation of our index based content management system. First, we briefly outline the most important features of the proposed e-travel system. Following are the proposed structures for manipulating and storing travel information. Finally, we describe the main features of the ebXML Registry/ Repository, our chosen implementation technology. An assessment of our initial experiences with ebXML R/R concludes the paper.

2. E-travel System

The initial design of the travel support system was presented in [1, 3, 4, 6]. On a high level of abstraction, the functionality of the system is divided into two coordinated subsystems, one responsible for content management and the other for delivery of content to the user (see also [15]). The most important function of the content management subsystem is to organize travel-related information in such a way to successfully support the content delivery mechanisms. Existing content delivery systems typically follow one of the two possible approaches to content management:

1) management *by aggregation*: retrieving all information that the system will possibly need beforehand, and organizing it in a predefined (by humans) format within databases for later access,

2) management *by selection*: maintaining a general idea as to what content is available on the Internet, indexing it, and retrieving it only as becomes necessary to satisfy the user's query.

Most online travel content gateways (e.g. Orbitz, Travelersadvantage, etc.) employ the first method, storing travel content locally and performing local searches in response to user queries. When a selection is made, (e.g. a particular hotel in Baltimore) primary source systems on the Internet are contacted (e.g. those run by travel providers such as hotel chains) for verification of locally cached information. The main advantage of this approach is the immediate availability of local content; this is also a disadvantage, in that it leads to problems of data coherency (e.g. the requested hotel in Baltimore which is displayed as available is actually sold-out, or a given discounted price is no longer available). In addition, the amount of data and continuous local processing necessary for aggregation systems to work makes them extremely resource intensive – this is also one of the reasons such systems trade in only a limited subset of travel choices (in addition to the fact that, at this stage of development of e-commerce, only income generating services, e.g. car rental reservations, bring income to e-travel agencies).

Search engines such as Google are hybrid systems, aggregating only a limited amount of data (such as page headers and few selected cached pages) necessary to support the search function. This approach attempts at striking a balance between the amount of content stored locally, frequency of local information updates and the precision of the search function. Rudimentary content organization and differentiation available in browsers combined with relative freshness of data, while relatively satisfactory for typical searches (of content that changes infrequently) is not enough to support travel-oriented services (where the freshness of content is paramount importance). In addition, the limitation of content to a, somewhat random, subset of pages representing a given site allows only a very rudimentary content organization and differentiation; for the most part this task is left to the user. A typical problem with this hybrid approach becomes evident when the user issues broad queries that result in an extremely large number of hits that reveal no simple way for the user to reduce them to desired content.

Our e-travel system fully embraces the second, index-oriented approach, as it was exemplified in [4, 13, 17, 18]. We have designed our system to index Internet content into a hierarchical local repository using non-objective terms [12]. To support the content delivery aspects of travel support the system dynamically utilizes remote content by referencing it from these local indices / pointers. Thus, the content management subsystem focuses on the classification of content instead of the content itself storing only enough information in indices to satisfy accurate user queries – like the "yellow pages". This technique eliminates the problems of data coherency (in aggregation systems), and is expected to ensure that the system will not waste resources on extraneous data. Furthermore, since only indices are stored locally, the local processing required to respond to user query can be substantially reduced. The downside of the indexing approach is that content must always be retrieved from a remote source. If a content provider "goes down" or becomes unreachable, the e-travel system is unable to retrieve the content pointed to by an index and thus cannot fulfill the user's request. More generally, any slowdown in reaching the content provider is reflected in the performance of the system (and potentially counterweights the advantages gained from local indexing). Nevertheless, in designing the e-travel system we felt that the advantages of accurate indexing and the avoidance of cache coherency issues compensate the disadvantages of remotely stored content.

2.1 Sources of Content Indices

Following the analysis presented in [1, 3], we assume that the travel content originates from verified and unverified sources on the Internet. Verified sources are referred to as Verified Content Providers (VCPs), which designation implies a degree of conformance to expected standards of accuracy, format and availability of supplied travel options. Content from VCPs can be either fed directly to the system or gathered by search agents, as described in [1, 4, 6]. In the first case, we assume that incoming indices (pointers to available information) are both in the required format and are complete, and thus can be stored in the system without further processing. In the second case, the acquired content indices may be incomplete and/or require further processing. When dealing with unverified sources the situation is similar to the latter case, with an added component of necessary verification and de-confliction of remote information (at this stage of system design we will omit these last two issues and assume that they have been successfully resolved). Regardless of source, the acquired indices are stored in the system for later access by the content delivery functions of the system. In the case when incomplete indices are acquired, an index completion subsystem is invoked to furnish the missing information [6]. When the user requests information, a relevant content pointer is either found in the system and the process of content extraction from the provider(s) is initiated (while additional Internet search(es) may be conducted to locate additional information), or a new index acquisition is forced, in order discover relevant content (from both VCPs and unverified sources).

2.2 Semantics

Ideally, the content management subsystem should shield the rest of the e-travel system from the mechanics of supply and retrieval of travel content. Additionally, it should allow the content delivery functions of the systems to operate on the assumption that travel information is accurately classified. In theory, this would require the content management subsystem to semantically "understand" the information it keeps track of [5, 17, 18]. Here we acknowledge that currently available technology does not support this assumption of semantic "understanding". In the absence of such technology, our system attempts the next best substitute. We apply a predefined categorical overlay to the travel information managed by the system, and allow the entire system to tune the accuracy of this overlay (e.g. with user, agent and supplier feedback, as described in [6]), with the ostensible goal of simulating real semantic classification. In addition, we pay close attention to the efforts initiated by the Open Travel Alliance that attempts at introducing a hierarchical description of the "world of travel" and most important processes taking place there [9].

2.3 Organization of Travel Content

Before travel resources can be categorized or indexed, a framework for the aforementioned overlay must first be defined. We have introduced the concept of the *site* as a basis for this classification scheme. A site is the real-world logical division upon which we model our treatment of travel resources. It consists of three basic components that we have observed to be common to all Internet-based travel content. These components are the *provider*, *type*, and *location*. *Type* and *location* refer to the real-world physical aspects of a travel resources; *provider*, on the other hand, is used to describe the electronic interface to the resource. The *provider*, *type* and *location* components of a logical site form a tuple [6]. Finally, the *?notes?* component is added to the tuple to support of various administrative functions necessary when dealing with data delivered by the search agents (for more details see Section 3). This tuple is the bridge between the logical site classification and implementation-level storage. A complete tuple has a form:

(<provider>, <type>, <location>, <?notes?>).

In selecting the three travel content defining components of the tuple, we seek a certain balance between verbosity, time and storage considerations. When too much data is incorporated into the tuple, there is an increased risk that it will change at the provider. This is called a data incoherency. To reduce this risk, the tuple contains only data that rarely changes. On the other hand, placing too little data in the tuple could lead to problems in categorizing the resource. Our objective was to avoid the pitfall of data incoherency while assuring that enough information was incorporated to properly categorize and interact with the resources.

Let us now look into the *provider, type* and *location* fields of the tuple in more detail.

2.3.1 The Travel *provider* component

The *provider* component describes the means of accessing travel resources on the Internet. It is stored in the form of a Uniform Resource Identifier (URI). This URI describes the access method for the resource, the location of the resource, and any marker data that may be unique to this resource within the provider. In addition to explicitly identifying the transport protocol, the protocol section also (directly or indirectly) identifies the access methods of the server. For example, http:// and ota:// each have their respective access methods (hypertext and Open Travel Alliance protocols). Other possible protocols include edi:// and soap://. The URI also contains the host name to communicate with using this protocol.

2.3.2 The Travel *type* component

The *type* component of a tuple describes the position of a travel resource in the taxonomic hierarchy of all resources (e.g. Accomodations -> Hotels -> Chains). The system will utilize this information to filter out content that is not pertinent to a users needs. Thus, it is the focal point for the proto-semantic division of travel information described. For example: if the user is interested in hotels, an agent will be able to retrieve only hotel indices from the repository. Current version of our taxonomy for the type component is derived from the modified Yahoo! directory of Travel and the Open Travel Alliance [9] XML Schemas. The content type is intended to define the relationships between travel resources. Illustrations of these relationships will be presented shortly.

2.3.3 The Travel *location* component

Geography and location are key factors for determining the relevance of indexed travel resources to a particular user's travel plans. The location component must be flexible enough to support the multiple ways it may be utilized. Location information must be specific enough to differentiate between different sites. It must be hierarchical so that organizational relationships between sites at different locations on different levels (continent, country, state, city, et al.) can be surmised (e.g. the destination is in a different country). Given these criteria, our initial design of the location component consists of: a taxonomic description based on the ISO-3166 standard, which defines the continent, country, state or province, and city; and latitude and longitude for exact locations and proximity searches.

The *type*/*location*/*provider* tuple as described above is the basis of the classification scheme utilized by all of the functions of the travel support system, from the retrieval of content from travel suppliers on the Internet to the delivery of travel choices to the end user. It is with these functions in mind that we proceed to manifest the tuple on the implementation level, and, we hope, provide an efficient means of communicating travel content. Let us also observe that the proposed schematic solves the, above indicated, problem of the Google directory. In our approach we are able to untangle the geospatial information from the travel resource information by providing two separate but complimentary "looks" at our data. In this way, we are also making an initial step toward developing ontology of travel. To this end, we turn to the ebXML Registry / Repository.

3. Building a travel index using the ebXML Registry / Repository

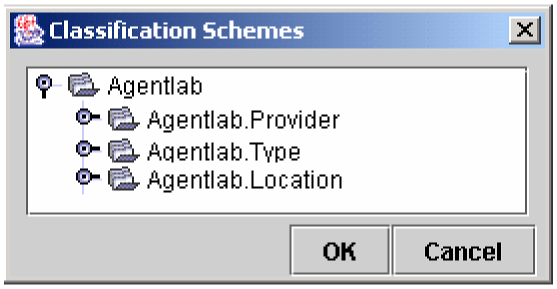
The ebXML Registry / Repository software, and the standard which define its implementation utilize XML to describe and enable information exchange between interested parties [11]. In the travel support system, the interested parties are the content management and delivery subsystems of the travel support system [1]. The former subsystem uses the facilities of the repository to classify and catalogue travel content from suppliers on the Internet. The delivery subsystem, on the other hand, looks to this catalogue as the ultimate reference to available travel choices. When constructing the registry (prior to the indexing of any content) the basic layout of the catalogue has to be defined. This layout is defined by a repository classification scheme that mirrors the indexing tuple. The root of the scheme is the *Agentlab* node. This node is a classification scheme object that has three children: the *Provider*, the *Type*, and the *Location* nodes. The bulk of the information in each of these nodes comes directly from the corresponding element of the tuple. For example, the External URI field of the *Provider* node is populated from the URI information of the tuple. The indexing agent [6] (and possibly the data completion subsystem) is (are) responsible for transforming the string data components of the tuple into these registry objects. When necessary, the indexing agent will construct new nodes to represent travel resources. It is also possible that nodes required for the construction of a travel index are already present in the repository. For example, if a new provider for a hotel that is already indexed by the repository is discovered, a new provider node must be created and associated with the existing type and location objects of the hotel. The inverse is also possible. For example, a single provider may house content on behalf of several hotels. In this case, associations between a single provider node and multiple name and location nodes will be created.

In addition to the data required for content identification and classification, the indexes require information to facilitate certain administrative functions. These administrative functions allow the indexing agent to mark newly created (but not yet validated) index items as inactive so that they are not inadvertently utilized in user requests until they are ready. The registry slot component may be used to implement this capability (this functionality was denoted by *?notes?* in the tuple specification presented above). This capability is also useful when dealing with index completeness. Search agents may discover sites that contain relevant content but not be able to acquire all of the data required by the tuple. In this event, the indexing agent will construct as complete index tuple as possible, and mark it as incomplete (preventing its inadvertent use). The completion of the index will be carried by the data completion subsystem. While the implementation details of this process are outside the scope of this document, we do recognize that it is a necessary component of the system and further details have been presented in [6].

Having established the basic requirements for constructing the appropriate classification nodes to create an index, we will now focus on the details of creating the classification scheme and its member nodes. The SubmitOjbectsRequest method is used for the creation of all new nodes. For the root node, (*Agentlab*) the ClassificationScheme member is used as demonstrated in the following XML description:

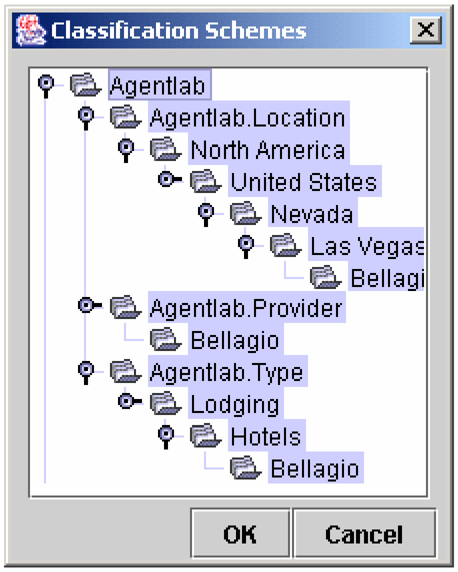
```
<SubmitObjectsRequest (parameters) >
  <LeafRegistryObjectsList>
      <ClassificationScheme id="<uuid>" (other parameters) >
  </LeafRegistryObjectsList>
</SubmitObjectsRequest>
```

As is the case with all repository objects, a globally unique identifier is required to create a node. The format for the Unique ID is that of a DCE 128 bit UUID [10]. If a Unique ID in the appropriate format is not supplied by the caller, the registry will create one. The Node representing the travel support system's classification scheme is called AgentLab. It appears as AgentLab in the Classification viewer component of the ebXML Registry Browser. As mentioned, the content for a particular site is uniquely defined by three nodes within the Classification Scheme. The *provider* node, the *type* node, and the *location* node are of type classification node. Nodes of each of these types are themselves members of a classification. The *Location* node is constructed from a scheme representing geography. The *Type* node represents types of travel content as described by our categorization scheme. A *Provider* node contains a URI defining the access method, and host address of the travel content. The following diagram outlines the high level classification:



```
Agentlab
    Agentlab.Provider
    Agentlab.Type
    Agentlab.Location
```

Once the basic structure has been defined, indexing can begin. An index is created by inserting three leaf nodes (as defined by the tuple and the diagram above) into the registry and then defining associations between them. In order to insert a node, it is necessary to locate (via repository query or other means) the parent of the node to be inserted, and reference it via the use of the "ObjectRef" class of the repository. As shown below:



```
Agentlab
    Agentlab.Location
        North America
            United States
                Nevada
                    Las Vegas
                        Bellagi
    Agentlab.Provider
        Bellagio
    Agentlab.Type
        Lodging
            Hotels
                Bellagio
```

Nodes representing the site "Bellagio" have been added to each of the three basic types. Also, notice that the Type and Location schemes are themselves hierarchies. In order to add the Bellagio "provider" to the provider hierarchy, the Unique Identifier for Agentlab.Provider had to be determined, and the addressed via the ObjectRef method.

Utilizing an explicit UUID for the base nodes in the hierarchies makes it possible to reference them without searching or querying. While the UUID's for the base nodes are explicitly specified, it is sufficient to allow the repository to generate a random UUID for the leaf nodes. This is because the Value attribute of the node object rather than the Unique Identifier is used to correlate the three nodes representing a site. The contents of the Value field are populated by obtaining a UUID using the getuuid registry method and appending to it the name of the site (i.e. "Bellagio"). By creating the nodes in this manner, registry browsers may construct queries that easily retrieve all of the nodes associated with a site. The site provider object is primarily a container for details about how to interact with the content. The registry attribute used to represent the server and access methods is the External URI. The URI is constructed then an association between the External URI and the provider node is created. There may be one or more URI's associated with a site, and a URI may be associated with many sites.

An interesting aspect of the construction of the Location "Site" nodes is the use of the registry "Slot" object to contain the Latitude/Longitude of the site. Future enhancements call for the slot to contain an OpenGIS object for the site. In addition, in the future, both the Name and Location Objects will have an external Link that directly references the Provider Site (reducing the necessity of retrieving additional objects when a site has been discovered). The XML that constructs the Bellagio index follows:

```
<rs:SubmitObjectsRequest (parameters)
 <LeafRegistryObjectList>
 <ObjectRef id="urn:uuid:00000001-0000-0000-0000-000000000000"/>
 <ClassificationNode id="urn:uuid:ffffffff-ffff-ffff-ffff-fffffffffff1"
 status="" objectType="ClassificationNode"
 parent="urn:uuid:00000001-0000-0000-0000-000000000000"
 code="utcr:00000003/Bellagio">
 <Name> <LocalizedString charset="UTF-8" value="Bellagio"/></Name>
 <Description> <LocalizedString charset="UTF-8"
 value="Content Provider Bellagio Hotel"/></Description>
 </ClassificationNode>

 <ObjectRef id="urn:uuid:00000002-0001-0001-0000-000000000000"/>
 <ClassificationNode id="urn:uuid:ffffffff-ffff-ffff-ffff-fffffffffff2"
 status="" objectType="ClassificationNode"
 parent="urn:uuid:00000002-0001-0001-0000-000000000000"
 code="utcr:00000003/Bellagio">
 <Name> <LocalizedString charset="UTF-8" value="Bellagio"/></Name>
 <Description> <LocalizedString charset="UTF-8"
 value="Bellagio Luxury Hotel Las Vegas"/></Description>
 </ClassificationNode>

 <ObjectRef id="urn:uuid:00000003-0001-0001-0001-000000000001"/>
 <ClassificationNode id="urn:uuid:ffffffff-ffff-ffff-ffff-fffffffffff3"
 status="" objectType="ClassificationNode"
 parent="urn:uuid:00000003-0001-0001-0001-000000000001"
 code="utcr:00000003/Bellagio">
 <Name> <LocalizedString charset="UTF-8" value="Bellagio"/></Name>
 <Description> <LocalizedString charset="UTF-8"
 value="Location of the Bellagio Luxury Hotel"/></Description>
 <Slot name="Coordinate"> <ValueList>
 <Value>Lat=36,10,42</Value>
 <Value>Long=-115,10,24</Value>
 </ValueList> </Slot>
 </ClassificationNode>
 <ExternalLink id = "urn:uuid:ffffffff-ffff-ffff-ffff-fffffffffff4"
 externalURI="http://www.Bellagio.com">
 <Name>
 <LocalizedString value = "Provider for the Bellagio Luxury Resort"/>
 </Name>
 <Description>
 <LocalizedString value = "Provider for the Bellagio Hotel"/>
 </Description>
 </ExternalLink>
 <Association id = "urn:uuid:ffffffff-ffff-ffff-ffff-fffffffffff5"
 associationType = "ExternallyLinks"
 sourceObject = "urn:uuid:ffffffff-ffff-ffff-ffff-fffffffffff4"
 targetObject = "urn:uuid:ffffffff-ffff-ffff-ffff-fffffffffff1" />
 </LeafRegistryObjectList>
</rs:SubmitObjectsRequest>
```

Once the data is in the repository, there are many ways of retrieving it. For example, the Adhoc Query may be used to search all of the URI's for a certain host name. Below is an example where the browser client was used to retrieve all of the nodes (of all three types) whose path attribute within the registry contains the string "Bellagio".

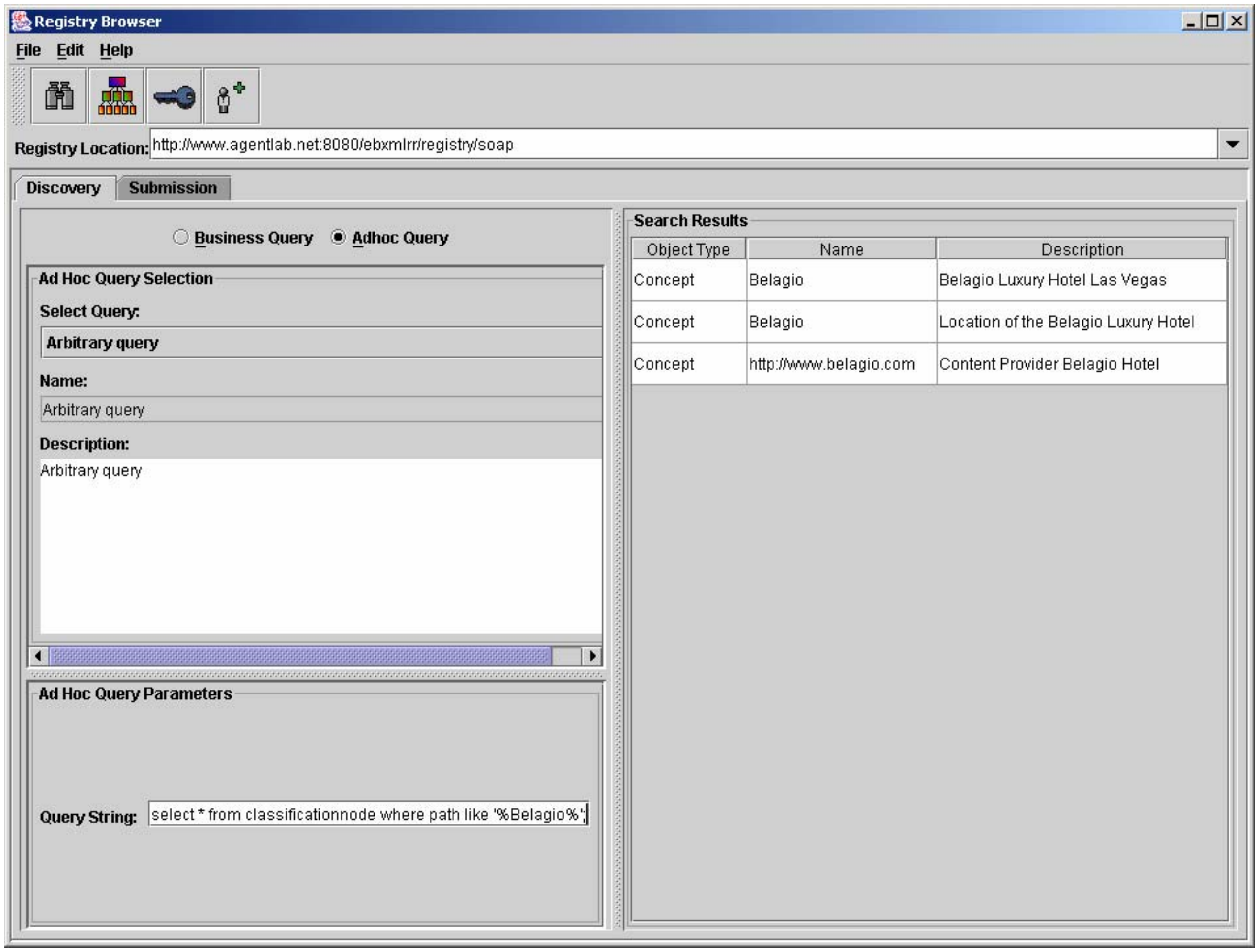
This example demonstrates a simple mechanism for obtaining all of the information pertaining to a specific site. (its *provider*, *type*, and *location*.) Notice that the adhoc query: "select * from classificationnode where path like '%Bellagio%';" did not return simple table rows (it returned RegistryItems). This is because "AdhocQuery" is a method of the respository (and not every Adhoc query string works). Also note that the 3 nodes (Provider, Type, and Location) for the string Bellagio were returned. If there had been more than one provider (or Site) for the Bellagio content, the UUID placed in the Value Attribute called the "uctr" Unique Travel Content Reference could have been used to Uniquely identify the desired Bellagio content provider (site).

Another interesting feature of the ebXML registry is the concept of Lifecycle. The addition of new objects to the repository is a dynamic process. When content of interest to the user has no index in the repository, an agent is initiated to locate some. The lifecycle of the index begins when the components of the tuple are validated and inserted into the registry. This does not mean that the provider is a VCP. A newly discovered provider does not immediately enjoy VCP status within the index. There is a progression of verification and feedback designed to determine the trustworthiness and reliability of the provider. The integrity of the information presented to the consumer is critical to the usability of the system and user satisfaction. Since the Travel System relies on external sources to deliver the content it cannot absolutely assure the availability of the data, the accuracy of the data, or the integrity of the provider of the data. As a site index progresses through its lifecycle, the system must help ensure its accuracy and integrity. The details of dealing with the trustworthiness of particular provider (site) are outside of the scope of this paper, but an approach similar to that applied across many e-communities (e.g. slashdot) can be applied here.

When dealing with travel content that expires it is useful to be able to retire a site or index. For example if a pub had, on a certain date, a special offer of two for one Corona beers, a slot with this information could be associated with the site's provider or location object, and would expire automatically when the lifecycle of the slot was completed (this functionality, again, belongs to the *?notes?* field of the tuple and has to be further investigated).

4. Early experiments

Our experience with the Registry Repository, while constructing our catalogue, has shown that it is well suited for this purpose. We found installation of the registry to be relatively straightforward. The system was quite flexible when it came to creating our classification schemes. It may in fact be too flexible. We found that there were limited controls governing the creation of new branches within our classification scheme. This places the responsibility for ensuring that the schemes are pruned correctly on the indexing agent. Another shortcoming we found with the registry was that it provided no interface to the OpenGIS objects native to the database (Oracle) we were using.



Registry Location: http://www.agentlab.net:8080/ebxmlrr/registry/soap

Select Query: Arbitrary query

Name: Arbitrary query

Description: Arbitrary query

Query String: select * from classificationnode where path like '%Belagio%';

Object Type	Name	Description
Concept	Belagio	Belagio Luxury Hotel Las Vegas
Concept	Belagio	Location of the Belagio Luxury Hotel
Concept	http://www.belagio.com	Content Provider Belagio Hotel

5. Concluding remarks

In this paper we have discussed the technology involved in indexing travel related content. Our main aim is to develop a complete e-travel support system. The ebXML Registry / Repository will be the central index repository of this system. Our initial experiments indicate that this is likely to be a good choice to support the necessary functionalities. In the near future we will combine the ebXML Registry / Repository with the search agents [6] and initiate the next phase of experiments. This time we will let the search agents autonomously search the web for travel related information and populate the repository. This experimental work (combined with the intelligent query capacities that are under development) will be used to reevaluate the results presented here and establish the final indexing schema that is going to be used in the system. We also expect that these experiment will provide us with additional insights that will help us is establishing the correct support for the administrative functions of the system (some of which have been already mentioned above). Finally, we have to return to the question of the classification of the travel resources. As indicated above, we are currently utilizing a slightly simplified Yahoo! catalog combined with the travel categories originating from the OTA project. We will study other existing classifications of travel as well as contact human travel experts in an attempt to develop a more complete classification. We will report on our progress in subsequent publications.

References

[1] Angryk, R., Galant, G, Gordon, M., Paprzycki M. (2002) “Travel Support System – an Agent-Based Framework,” Proceedings of the International Conference on Internet Computing (IC’02), CSREA Press, Las Vegas, pp. 719-725

[2] 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

[3] 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

[4] 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

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

[6] Paprzycki M., Gordon M, Harrington P., Nauli A., Williams S., Wright J., (2003) Using Software Agents to Index Data for an E-Travel System, Proceedings of the ABC Symposium, Orlando, Florida, July, 2003, to appear.

[7] Open GIS Consortium. http://www.opengis.org/techno/abstract.htm

[8] Semantic Web. http://www.semanticweb.org

[9] Open Travel Alliance http://www.opentravel.org

[10] DCE 128 bit Universal Unique Identifier
http://www.opengroup.org/onlinepubs/009629399/apdxa.htm#tagcjh_20
http://www.opengroup.org/publications/catalog/c706

[11] OASIS/ebXML Registry Services Specification v2.0, http://www.oasis-open.org/committees/regrep/documents/2.0/specs/ebrs.pdf

[12] Gudivada V, Raghavan V, Grosky W, Kasanagottu R, (1997) "Information Retrieval on the World Wide Web" IEEE Internet Computing 1997, Pages 57-68.

[13] Abramowicz, W., Kalczynski, P., Wecel, K. (2002) "Filtering the Web to Feed Data Warehouses." Springer Verlag Publishing, New York.

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

[15] V. Galant, J. Jakubczyc, M. Paprzycki (2002) Infrastructure for E-Commerce, in: M. Nycz, M. L. Owoc (eds.), Proceedings of the 10th Conference Extracting Knowledge from Databases, Wrocław University of Economics Press, Wrocław, Poland, pp. 32-47

[16] H. Nwana, D. Ndumu, A Perspective on Software Agents Research, The Knowledge Engineering Review, 14 (2), 1999, pp. 1-18

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

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