**D7 – Dynamic Ontology Management System (Design)**

**WorkPackage:** WP3 - Dynamic Ontology Management System  
**Version:** Final 1.0  
**Author(s):** Editor: A. Kokossis, P. Sujoy  
**Company(s):** UniS, URV, VUA, ATC  
**Submission Date:** 1/10/2002  
**Confidentiality:** Consortium & EC  
**Deliverable Type:** External

© Copyright by the h-TechSight Consortium.

The h-TechSight Consortium consists of:

| University of Surrey (UniS) | Project Coordinator | UK |
| University Rovira i Virgili (URV) | Partner | Spain |
| Vrije Universiteit Amsterdam (VUA) | Partner | Netherlands |
| SemanticEdge | Partner | Germany |
| Athens Technology Center SA (ATC) | Partner | Greece |
| IchemE | Partner | UK |
| BAYER AG | Partner | Germany |
| P.N. GEROLYMATOS SA (PNG) | Partner | Greece |
| Jet Oil | Partner | Greece |
| Institut Français du Pétrole (IFP) | Partner | France |
SUMMARY

One of the objectives of the h-TechSight project is the development of tools that help to build and dynamically maintain knowledge maps related to a specific knowledge-intensive domain. A first design of a dynamic ontology management system to be used for this purpose is proposed in this document.

In Section 1 the rationale concerning the development of a Dynamic Ontology Management system is described. In Section 2, we present the basic capabilities of a dynamic ontology management system; the system should be able to build, manage and dynamically update ontologies. Sections 3 and 4 are focused on the two main components of the system: an ontology personalisation interface (built on top of OntoEdit and Sesame) and a multi-agent information search engine. This multi-agent search engine is composed of a collection of intelligent information agents which search the web to find and rate concepts related to the ontology provided by the user.

In summary, the aim of this first version of this document is to present an initial design for a dynamic ontology management system that will be able to semi-automatically manage the dynamic evolution of ontologies in a complex, real-world domain. A second version of the documents will be released when the dynamic ontology management system is fully designed. A prototype of this system will be presented at month 18 of the project.
Table of Contents

SUMMARY.................................................................................................................................................. 2

1. INTRODUCTION ..................................................................................................................................... 5
   1.1 RATIONALE ....................................................................................................................................... 5
   1.2 ACRONYMS AND ABBREVIATIONS ................................................................................................. 5
   1.3 LIST OF RELATED DOCUMENTS ..................................................................................................... 6

2. A DYNAMIC ONTOLOGY MANAGEMENT SYSTEM............................................................................ 7
   2.1 REQUIREMENTS .............................................................................................................................. 7
   2.2 ARCHITECTURE OF THE DYNAMIC ONTOLOGY MANAGEMENT SYSTEM ...................................... 8

3. ONTOLOGY PERSONALISATION INTERFACE ..................................................................................... 10
   3.1 ONTOEDIT ...................................................................................................................................... 10
   3.2 SESAME......................................................................................................................................... 11
       3.2.1 Overview................................................................................................................................. 11
       3.2.2 The Repository Abstraction Layer .......................................................................................... 11
       3.2.3 Repository............................................................................................................................. 11
       3.2.4 The RDF Administration Module .......................................................................................... 11
       3.2.5 The RQL Query Module ........................................................................................................ 11
       3.2.6 The RDF Export Module ....................................................................................................... 11
   3.3 THE REPOSITORY CONSTRUCTION: GINY ......................................................................................... 11
   3.4 JENA .............................................................................................................................................. 11
   3.5 XBASE.......................................................................................................................................... 11

4. A PROTOTYPE MULTI-AGENT INFORMATION SEARCH SYSTEM ................................................... 11
   4.1 INTRODUCTION ............................................................................................................................. 11
   4.2 MULTI-AGENT SYSTEM ARCHITECTURE ..................................................................................... 11
   4.3 CASE STUDY: SEARCH FOR BIOSENSOR COMPANIES .................................................................... 11
   4.4 FUTURE WORK ............................................................................................................................... 11

REFERENCES ............................................................................................................................................... 11

APPENDIX A: WEB SITE MATCHES ............................................................................................................. 11
Index of Figures

Figure 1: Dynamic Ontology Management System ................................................................. 8
Figure 2: OntoEdit interface .................................................................................................. 10
Figure 3: Options available in OntoEdit after connecting to Sesame ..................................... 11
Figure 4: Sesame architecture ([Broekstra et al., 2002]) ......................................................... 11
Figure 5: Stacking of RALs to add functionality ([Broekstra et al., 2002]) ............................ 11
Figure 6: The RQL query module ([Broekstra et al., 2002]) .................................................... 11
Figure 7: Evaluating an RQL query using Sesame ................................................................. 11
Figure 8: GINY’s Framework Model ....................................................................................... 11
Figure 9: Xbase System Schema .......................................................................................... 11
Figure 10: Multi-Agent Search System Architecture ............................................................ 11
Figure 11: Ontologies handled by the Coordinator Agent ....................................................... 11
Figure 12: Product class and its properties ............................................................................ 11
Figure 13: Documents found for each class in the "Company" ontology .................................. 11
1. INTRODUCTION

1.1 RATIONALE

Since in a complex technical domain not every user may be interested in all aspects of the domain, the ontologies developed in h-TechSight need to be organised according to different user roles. Such personalised ontologies must be flexibly configurable in order to cope with a complex and dynamically changing environment. For example, it must be possible to define personal organisation schemes that allow classifying ontologies with respect to a personal view (i.e., topics related to the personal competencies). To provide this ability, we need to develop a tool that will support this type of functionality. In order to ease the use of the tool, it should be able to automatically classify the stored knowledge, at least according to some standard categories.

In a chemical engineering example, for instance, there is a “Product Specifications” model with standard categories “Product_chemistry”, “Production_technology”, “Materials_suppliers”, “Product_markets”, “Environmental_burden” (or “Global_warming_effects”), and “Product_maturity”. Thus a user could specify a “Product Specifications” ontology from a “Production_Technology” perspective. This should give the user only parts of the ontology, which are relevant to production technology.

Besides providing the ability for a user to personally configure an ontology, it would be extremely important and beneficial to incorporate self-correcting capabilities to the ontology. This will provide the ability to grasp the dynamic profile of a knowledge domain, (e.g. trends and expertise in demand, new dynamics and markets, declining fields and professions), which may translate to impressive gains in increasing business intelligence.

In view of the above, the h-TechSight project aims to provide a Dynamic Ontology Management system that will incorporate these capabilities.

1.2 ACRONYMS AND ABBREVIATIONS

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAML</td>
<td>DARPA Agent Markup Language.</td>
</tr>
<tr>
<td>DB</td>
<td>DataBase.</td>
</tr>
<tr>
<td>DBMS</td>
<td>DataBase Management System</td>
</tr>
<tr>
<td>FIPA</td>
<td>Foundation for Intelligent Physical Agents.</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HTML</td>
<td>HyperText Markup Language.</td>
</tr>
<tr>
<td>JADE</td>
<td>Java Agent Development Environment.</td>
</tr>
<tr>
<td>MAS</td>
<td>Multi-Agent System.</td>
</tr>
<tr>
<td>OIL</td>
<td>Ontology Interchange Layer.</td>
</tr>
<tr>
<td>OODBMS</td>
<td>Object Oriented DataBase Management System</td>
</tr>
<tr>
<td>RAL</td>
<td>Repository Abstraction Layer.</td>
</tr>
<tr>
<td>RDB</td>
<td>Relational Database</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework.</td>
</tr>
<tr>
<td>RQL</td>
<td>Object Oriented Query Language</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language.</td>
</tr>
</tbody>
</table>
SOAP  Simple Object Access Protocol
WWW  World Wide Web.
XML  eXtensible Markup Language.
XOL  XML Ontology Exchange Language

1.3  **List of Related Documents**

1. IST-2001-33174: H-TechSight Annex 1, Description of Work
2. D2 – State of the Art KM technologies and practices
2. **A DYNAMIC ONTOLOGY MANAGEMENT SYSTEM**

2.1 **REQUIREMENTS**

The Dynamic ontology management system should be able to satisfy the following requirements:

- **Building dynamically interrelating ontologies:** Provide the ability to replace conventional ontology schemas with advanced ontologies that can analyse and present highly relevant information. In particular, to be able to validate and sustain components from the ontology using the available information from the web (or any other source). Conceptually similar documents will be clustered and grouped into one or more categories and the information can become available by crawling text sources on particular subjects, with a purpose to generate statistical information such as term frequency and proximity.

- **Automatically updating ontologies:** The ontologies should be automatically updated as knowledge assets are added to and removed from repositories. The ability to analyse documents and assign them to an appropriate category is one way to overcome one of the largest obstacles to effective knowledge management: the timely and accurate knowledge classification.

- **To periodically and dynamically assess the relative position of the ontology and its components:** Provide the means to monitor and dynamically update knowledge structure (knowledge maps). Be able to periodically and dynamically analyse changes in the knowledge domain, point new fields and terms, and automatically report findings to the user.

Technically speaking the defined requirements can be translated in the following:

1. Design and implementation of an interface to personalise ontologies and customise the knowledge map of the technology areas that the user needs to explore. This should be able to allow:
   - Importing and reusing existing ontologies.
   - Ontology pruning for removing irrelevant parts of an imported ontology
   - Ontology refinement to complete the ontology at a fine granularity
   - Ontology consistency management to ensure consistent evolution of the ontology.

2. Design and implementation of Search Agents that are capable to discover new concepts and support the automatic update of the ontology terms.

Both the framework for defining ontologies and the ontologies themselves need to be defined using extensions of XML such as XOL (XML Ontology Exchange Language) and of RDF (Resource Description Framework) such as RDFS. The utilisation of standard representation frameworks will ensure interoperability of the *Ontology Personalisation Interface* and the *Multi-Agent Information Search System*. The design of these sub-systems is described in the following sections.
2.2 ARCHITECTURE OF THE DYNAMIC ONTOLOGY MANAGEMENT SYSTEM

The design of the proposed system is depicted in Figure 1. The system consists of two differentiated but collaborative parts:

- The intelligent search engine (shaded top block) and
- The knowledge repository and ontology personalisation interface (shaded bottom right block).

Figure 1: Dynamic Ontology Management System

The expert communicates with the proposed system by generating a domain ontology or by asking the personalisation interface system to provide him/her with an ontology customised to his/her own requirements. Once a domain ontology is created, the user agent (A\textsubscript{user}) requires information related to this ontology to the knowledge repository or to the search engine. The agent decides whether the information stored in the repository is good enough for the user expectations or the search engine is required to update this information with new knowledge extracted from the web.

These two main subsystems are described in more detail in the following sections of this document, but we want to first show the exchange of information between them with an example. Let's assume that a user who is not an expert requires information about cells applied in biosensors. The user could import an already built biotechnology domain ontology (domain ontologies are stored in the RDFs database, repository, and are supported by SESAME). After importing a domain ontology, the user could perform two actions: he/she can consult the information stored in the repository (the repository stores information which was previously discovered from the web or manually introduced); alternatively, the user can invoke the intelligent search engine and obtain new data from the web. The first option is the simplest and it only requires one select operation from the database. The second option requires the use of search agents that seeks information about cells applied in biosensors. The agents return a sorted collection of links which will be analysed by GINY [Riaño and Gramajo, 1998] in order to compose the instances of the domain ontology. Following the example, let's suppose that the http://www.fst.rdg.ac.uk/courses/fs560/topic3/topic3.htm link was found by a search agent, GINY...
analyses the link and discovers that there are microbial cells, amplifiers, antibodies, etc. with different attribute values. All the discovered information will be sent to the user for further consultation and analysis. The information will also be sent to the repository to get stored.

The ontology personalisation interface and the repository constructor tool, GINY, are introduced in the next section. The intelligent search engine is described in Section 4.
3. **Ontology Personalisation Interface**

The advent of the semantic web ([Berners-Lee, 1999]) has made it imperative to develop ontologies as the mechanism that enables machines to communicate and garner knowledge. In light of this context, we propose a design that will enable us to seamlessly interact the front-end ontology editing tool (OntoEdit) with a repository (MySQL, http://www.mysql.com). This is facilitated by the use of a middleware tool (Sesame).

3.1 **OntoEdit**

OntoEdit is a graphical based Ontology Engineering tool (its GUI is shown in Figure 2) that supports the development and maintenance of ontologies and is available either in the stand-alone or client-server format [Staab et al., 2000][Sure et al., 2002]. The internal ontology model on which it is based allows the domain to be developed through the use of classes, relations, axioms, facets and attributes.

The interoperability of the tool is facilitated by the development of plugins which permits it to interact with other tools like Ontobroker and Sesame. There are plugins that allow the user to import and export ontologies, portal assembly kits, integrated inferencing and ontomapping ([Sure et al., 2002]). The connection between OntoEdit and Sesame is shown in Figure 3.

The tool permits the usage of DAML+OIL and RDFS. There are two versions available (a free edition with some limitations and the professional edition with all the features). The Free version allows the usage of only 50 instances whereas the Professional version, in addition to the facilities available for the Free version, has an extended functionality that allows consistency checking of the ontology and an ontology server for the administration and storage of ontologies.

![Figure 2: OntoEdit interface.](image-url)
3.2 SESAME

Sesame, developed by AIdministrator (see http://www.aidministrator.nl), is an architecture that allows persistent storage and effective querying of metadata in RDF and RDFS. This tool is implemented using Java and it can be download for free. Sesame can easily be within other complex systems, such as the one being developed in the h-TechSight project.

3.2.1 Overview

Sesame has been designed and implemented to be independent of any storage devices. As such, it can be implemented on top of RDBMS, OODBMS, triple stores without having to change the query engine.

Sesame interacts with the DBMS through the Repository Abstraction Layer (RAL) interface. RAL communicates with 3 functional modules viz. the RDF administration module, the RQL query module and the RDF export module. These components then communicate with their clients either by HTTP or SOAP by their respective protocol handlers as shown in Figure 4.
3.2.2 The Repository Abstraction Layer

Repository Abstraction Layer (RAL) is a stable interface that offers RDF-specific methods to the clients and translates these calls to API methods for specific DBMS. As such, it allows the user to insert, delete and retrieve metadata from the repository. Currently (for version Sesame 0.6), only 2 DBMS are supported by RAL: MySQL (http://www.mysql.com) and PostgreSQL (http://www.postgresql.org).

RAL has been designed to transfer data in streams thereby facilitating the return of result sets as soon as they are available without having to store large amounts of data in memory ([Broekstra et al., 2002]). This gives the flexibility to access large volumes of data without the need for memory configuration or expensive hardware. As such, the repository is used for the persistent storage of the data. However, without caching the data in memory it impedes performance and minimizes one of the essential functionalities of RDBMS.

In order to ameliorate this problem, layers of RAL can be stacked on top of another, allowing the functional modules to communicate with only the top layer of the abstraction layer. The RAL at the top then forwards these call to the layer beneath and the process goes on till the result is returned, as shown in Figure 5.

Figure 4: Sesame architecture ([Broekstra et al., 2002])
3.2.3 Repository

The different types of repositories that Sesame allows are as follows:

- **DBs**: Although RAL has been designed to permit the interaction with different DBMS, Sesame (as of version 0.6) only communicates with MySQL and PostgreSQL databases.
- **Files**: Flat files can be used as storage devices which when combined with a schema-caching RAL can be effective in dealing with small amounts of data.
- **Repositories**: Sesame can access data across any RDF network services and hence permits the usage of remote repositories. It would be interesting to see whether we can make usage of added database functionalities like replication.

3.2.4 The RDF Administration Module

This module currently performs 2 functions, namely the insertion of RDF data and schema incrementally and deletion of the data from the repository. It is, however, not possible to update RDF data and schema. The options available are (1) uploading data, (2) removing statements and (3) clearing the entire database. Therefore, the solution for 'updating' an ontology is to extract all data (e.g. using OntoEdit), modify it locally, clear the entire database, and upload the modified data (4).

This module retrieves information from the RDF(S) source, parses it using the SiRPAC RDF parser [Barstow, 2000]. The parser then delivers the information to the module according to the RDF...
statement (Subject(S), Predicate(P), Object(O)) and checks the consistency of each statement with the information already present in the repository and draws information, if necessary, as:

- if P equals type, then it infers that O must be a class
- if P equals subClassOf, then it infers that both S and O are classes
- if P equals subPropertyOf, then it infers that both S and O are properties
- if P equals domain or range, then it infers that S is a property and O is a class

3.2.5 The RQL Query Module

Sesame uses a variant of RQL ([Broekstra et al., 2002], [Alexaki et al., 2000]) as compared to the declarative language originally developed within the European IST project C-Web. This version of RQL supports both optional and multiple domain and range restrictions.

The module parses the query, builds a query tree model and is then fed to the query optimiser. The optimiser then transforms the query model into an efficient, equivalent model. This is illustrated in Figure 6.

![Figure 6: The RQL query module ([Broekstra et al., 2002])](image)

The query model is subsequently evaluated in streams, thereby facilitating the retrieval of data without caching the data. This model translates RQL queries into a set of calls to the RAL, thereby allowing the evaluation of the query to be done in the query engine itself unlike the RDF Suite where the query is performed in the engine of the DBMS (see Figure 7).

3.2.6 The RDF Export Module

This module supports the usage of both the schema and non-schema part of the data. It exports the contents of a repository formatted in XML-serialised RDF.

Ontology editors like OntoEdit can then use either the schema or non-schema or both from the extracted information.
3.3 THE REPOSITORY CONSTRUCTION: GINY

GINY [Riaño and Gramajo, 1998] is a framework that allows the automatic structuring of information. An unsupervised Clustering process is employed to obtain the structure that should be used to store the information about a specific domain which is represented by a domain ontology [Jain and Dubes, 1988].

As shown in Figure 8, GINY is organised as a front-end where users (human or agents) can make queries and obtain responses from a centralised database.

GINY receives the domain ontology and communicates with the multi-agent search engine to obtain the unstructured retrieved information related to the domain. This information is used by the Data Structure Extractor to obtain a conceptual ER model to store this information. This model can produce both a DDL script that defines a relational database or a RDF description of a knowledge-base.

GINY’ s framework is depicted in Figure 8. GINY requires an ontology that provides the structure of the required information (classes and attributes which will get a value when an instance of this class is discovered) and links to web pages that contain information about the classes. Every link is considered as an instance of a class and thus the values for its attributes can be obtained from the analysis of such web pages. The web pages will be analysed by a Natural Language Analyser module which builds a data table with the extracted responses. Finally, that table is given to the Data Structure Extractor Module that generates the data and knowledge structures and fills them with the web information. These structures will be then stored in the repository where the user can access them through SQL queries.
3.4 JENA

JENA (http://www.hpl.hp.com/semweb/index.html) is a toolkit for developing semantic web applications. This toolkit is a Java API that manipulates RDF models. Its features include:

- statement centric methods to manipulate a RDF model as a set of RDF triples
- resource centric methods to manipulate a RDF model as a set of resources with properties
- cascade method calls for more convenient programming
- built in support for RDF containers - bag, alt and seq
- enhanced resources - the application can extend the behaviour of resources
- integrated parsers (ARP and David Megginson’s RDFFilter)

This latest release of the Jena toolkit integrates a number of new components, some of which are also available separately:

- ARP parser compliant with latest working group recommendations
- integrated query language (RDQL)
- support for storing DAML ontologies in a model
- persistent storage module based on Berkeley DB
- support for persistent Jena models in relational databases
- open architecture supporting other storage implementations
3.5 XBASE

XBASE is a system that allows the translation of information between XML and a Relational Data Base (RDB), it implements query mechanisms as a subset of the SQL language over a set of XML data files and RDB. The access to these two kinds of data sources is transparent to the user who receives an integrated answer of the data. The translation process is shown in Figure 9.

![Xbase System Schema](image)

**Figure 9: Xbase System Schema**
4. A PROTOTYPE MULTI-AGENT INFORMATION SEARCH SYSTEM

4.1 INTRODUCTION

The system to be described helps the user in the search for information on a specific domain. Its objective is to avoid as much as possible the two main problems of keyword based web search, namely that some of the retrieved documents are irrelevant, and that some of the relevant documents may not be retrieved. These problems derive from the nature of keyword based search and from the lack of structure in the searched documents.

It has been identified previously that ontologies can improve the accuracy of web searches because the search program can look for web pages that refer to a precise concept (in context and with detailed information) [Kokossis&Bañares-Alcantara, 2003]. The prototype system involves several ontologies:

- a domain ontology (an input from the user), providing the initial information about the domain of search,
- a set of query ontologies (intermediates), resulting from partitioning the domain ontology into smaller (more specific) ontologies,
- a set of response ontologies (intermediates), consisting of the information found during the search (one response ontology is created per each query ontology), and
- an information ontology (an output from the system), which is the result of merging the response ontologies.

All of them are implemented in RDF (Resource Description Framework, http://www.w3.org/RDF) which is becoming the de facto standard for the representation of ontologies [Fensel, 2001]. The domain ontologies were generated with two well known graphical ontology editors, OntoEdit (http://www.ontoprise.de/ontoedit.htm) and Protégé-2000 (http://protege.stanford.edu/index.html).

4.2 MULTI-AGENT SYSTEM ARCHITECTURE

Figure 10 depicts the Multi-Agent System architecture.

![Figure 10: Multi-Agent Search System Architecture](image-url)
The agents that perform the search for information and its further classification are:

- **The Internet Agents.**
  Given a concept with some associated properties, i.e. a query ontology, this type of agent searches the web with the help of one or more internet search engines and returns the web pages related to this concept in the form of a response ontology. The retrieved web pages are sorted according to a weight function. The system can deploy as many internet agents as requested.

- **The Coordinator Agent.**
  This agent receives as an input the domain ontology created by the user, and is responsible for partitioning it into several query ontologies which are sent to the Internet Agents. The Coordinator Agent is in charge of generating the knowledge base that stores all the responses given by the Internet Agents. The Coordinator Agent must therefore merge and classify all the response ontologies into the information ontology. The response ontology generation process is not an easy task. The information is distributed and sometimes duplicated, and the agent must be able to place it in the appropriate location within the structure created by the user (see Figure 11).

- **The Broker and User Interface Agents.**
  The Broker Agent acts as a facilitator, indicating to the Coordination Agent, for example, the state of each of the Internet Agents so that the idle ones can be assigned a task. The User Interface Agent allows the user to specify a query and to configure the initial ontology, in other words control the target and focus of the search.

![Figure 11: Ontologies handled by the Coordinator Agent](image-url)
The first prototype of this system has been implemented in JADE (Java Agent Development Environment [Bellifine et al., 1999]), which is a Multi-Agent platform composed by a collection of Java libraries and that follows the protocol specifications of FIPA (Foundation for Intelligent Physical Agents (FIPA), www.fipa.org) [Fipa, 2002]. For the case study the Internet Agents are connected to Google, the search engine (www.google.com), although the system is open to other search engines.

4.3 CASE STUDY: SEARCH FOR BIOSENSOR COMPANIES

The Multi-Agent search prototype was initially tested with a biosensor domain ontology. This ontology consists of 71 classes with more than 80 properties encapsulated within the classes. The "Company" class and its subclasses are depicted in the left-hand side of the OntoEdit window shown in Figure 12. The "Product" class is highlighted and shows all of its properties in the right-hand side of the window. For this specific class the user decided to include 7 properties or attributes which are related to this concept, and they are added to the inherited properties from the parent classes of "Product" (the inherited properties are shown with a grey shade).

If the concept to be searched were "Biosensor", the Coordinator Agent would split the domain ontology into 71 query ontologies. In this case study we have analysed a subset of this ontology: the "Company" class together with all its subclasses as a result the system works with 9 query ontologies.

In our first prototype, every agent analyses all the links associated with the first 10 results retrieved from the Google search engine, this amount is configurable. The names of the class and its parent classes are supplied as keywords to the search engine. Several searches were done with different numbers of Internet agents and different deadlines. Figure 13 shows the average number of pages (documents) found which are related to each of the 9 query ontologies using 2 and 5 agents with different deadlines (0.5, 2.5 and 5.0 minutes). We can see that the system behaves similarly when operating with deadlines of 2.5 and 5.0 minutes, and that an increase in the number of Internet agents improves the search. Bear in mind that the agents in the case study must perform searches for several ontologies as there are less agents than query ontologies.
At present the Internet agents analyse every web page and calculate a relevance weight according to the number of properties of the concepts that can be matched on the document text, and to their position in the document, i.e. whether they are in the title, headings, plain text, etc.

Once all the pages are processed, they are sorted and sent to the Coordinator Agent. As of the time of writing this document, the Internet agents generate the response ontologies but the Coordinator Agent does not yet perform the merge procedure to create the information ontology as planned in the final architecture, see Figure 10.

To validate the results, we have manually visited the web pages related to the "Environment_device" class (see Appendix A), and found that the more relevant web pages were indeed those with larger assigned weights. However, the Internet agents also retrieved irrelevant web pages, these pages were at the bottom of the relevance list though. As it stands, the user must decide on a “cut value” for the weights below which results are most likely irrelevant. This value cannot be generalised to other searches, thus a true solution rather than a patch requires that the web page analyser within the Internet agents be refined to eliminate as many irrelevant pages as possible.

From the results of the case study we can conclude that the prototype system finds relevant information (although we cannot prove that it finds all of it!) and successfully prioritises it.
4.4 Future Work

The work has concentrated on the implementation of the Internet agent and this report describes the preliminary results obtained with our first prototype. The system is not fully implemented yet and requires further work. We are currently modifying the Internet agents to:

- Improve their weight function calculation.
- At the moment the weight given to each web page is calculated by looking at which properties are found in the page, the number of such properties, and their position on the document. We want to improve this function by adding techniques used by other approaches [Glover, 2002].
- Improve the response ontology.
- In this first prototype the response ontologies consist of a sorted list of web pages. The response ontologies can be enhanced by providing values for each property found in every web page.

Once the Internet agents are fully implemented, the Coordinator Agent must be completed to generate the information ontology that merges and organises all the response ontologies, see Figure 11. This process is not an easy task, as the retrieved information may be duplicated. As different Internet agents may find and analyse related web pages, the Coordinator agent should use its knowledge of the hierarchical relation between the ontology classes and decide if the information is redundant (and thus remove the repeated information) or not (and keep the two related but complementing pieces of information). The resulting information ontology will enhance the domain ontology by adding knowledge extracted from the web pages by the MAS information search system.

The discovery of new concepts will be the next stage of the project. All the web pages can be processed to discover new concepts that were omitted or unknown to the user. This concepts consist of new classes, or properties of classes that are found in the retrieved pages but that the user did not include in the domain ontology. Web mining tools can be used to build ontologies from HTML documents. The resulting ontology can be compared with the domain ontology (the one created by the user) to detect new classes or properties. Applicable techniques are employed to generate databases with information retrieved from internet [Gramajo and Riaño, 2002].
REFERENCES


APPENDIX A: WEB SITE MATCHES

Environment_device subClassOf: Product matches: 14

THE LARGEST SINGLE AREA OF APPLICATION FOR THE ENVIRONMENT
Title: Biosensor Technology
Images: 3
Ratio: 1

This is a web of the University of Georgia about biosensor technology. Some topics are:
- What is a biosensor?
- Biological recognition elements
- Transduction technology
- Biosensor applications

THE DEVICE
Title: Use a carbon materials as biosensor controls
Images: 3
Ratio: 1

Russian web about physical, chemical and biological sensors. This web talks about a device that represents measuring systems of electrical activity of microbiological test in liquid environment.

ALPHA WEB SMARTS - GLOSSARY - AUSTRALIA'S EXPERT WEB
Title: ATS: EIDN Pty Ltd -- Luciferase-Based Biosensor: Contaminant Detection -- Energy, Environment, Waste Management - Australian Technology Showcase
Images: 5
Ratio: 1

Australian web about contaminant detection. It states that:
Detection of environmental and biological contaminants has become much easier with the development of a new Luciferase-based biosensor by the Technology Brokering Division of EIDN Pty Ltd.

TRANSDUCER - A DEVICE
http://www.rpi.edu/dept/chem-eng/Biotech-Environ/BIOSEN/basicsensor.html
Title:
An Introductory Tutorial on Biosensor Technology. This web talks about issues related to biosensors such as:

- Use for biosensors
- Molecular recognition
- Signal transduction
- What are the advantages of biosensors over other measurement schemes?

A BIOSENSOR CAN BE DEFINED AS A COMPACT ANALYTICAL DEVICE

http://www.nbbnet.gov.my/research%20project/exe%20summary/zam.htm

Malaysian web about Development of Biosensor Technology For Detecting Specific Contaminants in Environment and Agriculture. Issues:

- Objective of the project
- Status of the research
- Methodology and strategies of the research

ALPHA WEB SMARTS - THE WEB RANKING COMPANY

http://www.alphawebsmarts.net/pages/glossary.html

This is another link of the Australian page found before. The web page presents a glossary about computer science.

HUMAN RIGHTS AND ENVIRONMENT RESOURCES

http://shr.aaas.org/hrenv/glossary.php?g_id=5268

A web page of the Science and Human Rights Program. There is a definition of biosensor.
Document from a Spanish web page, titled: Technology offer: Real time monitoring of BOD (Biological Oxygen Demand) in effluents

The Spanish company BIOSENSORES has developed a very innovative biosensor for measuring the BOD (Biological Demand of Oxygen) in water.

Another link of the Australian page found before, with a quotation form.

A web page from the University of Georgia, about the 2003 annual meeting of the Institute of Biological Engineering.

Article about Biosensors from The World & I On-Line

New direct analysis devices made by linking biological and electrical components may soon replace time-consuming, laboratory methods for detecting small quantities of specific substances.
Web page of the University of Virginia about a Biosensor toolkit. Some issues described in this web page:

- Molecular Reception.
- Biomaterial Science.
- Biocomputing.

WEB


Title: The Waldorf Apartment Hotel - Table No. 3

Images: 13

Ratio: 0.51

This is an incorrect retrieval, because it is from an apartment hotel in Sydney. A possible reason for the inclusion of this web page could be that this page has a link to the previous Australian web page. The system found the word “price” which is an attribute of the class “Environment_device” and assigns an erroneous ratio.