 UNIVERSITY OF CALGARY	Course Number: SENG 609.22	Course Name: Agent-based Software Engineering
	Session: Fall, 2004	Department: Electrical and Computer Engineering
		Document Type: Project Report

Search Interest System

SENG 609.22 Agent Based Software Engineering

Garrett Camp & Kobe Davis

<http://inorder.org>

Table of Contents

1. System Specification

1.1 Business Case

1.2 System Description

1.3 Assumptions

1.4 Requirements

1.5 Wish list

2. System Architecture

2.1 Organization

2.2 Goals

2.3 Agent Architecture

3. System Design

3.1 Agents

3.1.1 Database/KB extraction agent

3.1.2 Search Agent

3.1.3 Caching agent

3.1.4 Peer Clustering, News & Ad agents

3.1.5 Interface Agent

3.1.6 Extraction Agent

3.2 Agent Use Cases

3.2.1 KB selection agent

3.2.1.1 Connect to KB agent

3.2.1.2 Select DB agent and send query

3.2.1.3 Respond to search from search agent

3.2.2 Search Agent

3.2.2.1 Contact KB selection agent, combine results from each selected KB

3.2.3 Caching Agent

3.2.3.1 Send List of URL's

3.2.3.2 Browse Cached results

3.2.4 Clustering/News/Ad agent

3.2.4.1 Get related objects

3.2.5 Interface Agent

3.2.5.1 Make/Select Topic

3.2.5.2 View and/or adjust term ranks

3.2.5.3 Keyword recommendation

3.2.5.4 Reformulate query

3.2.5.5 Browse related pages

3.2.6 Extraction agent

3.2.6.1 Create Group

3.2.6.2 Create Topic

3.2.6.3 Create Related terms

1.1

3.3 Agent Internal architecture

3.3.1 Example agent architecture

3.4 Data model

4. Message Specification

4.1 Message protocol

4.2 Message formats (inter-agent)

5. Technology Overview

5.1 TCP/IP/HTTP

5.2 LAMP system

5.3 XML

5.4 SOAP

6 Design Decisions

6.1 Agent Roles

6.2 Usability Considerations

6.3 Agent Architecture/Messaging

1 System Specification

1.2 Business Case

In today's 'online' world, one of the most important and common activities is searching for information. For example, take Google, a company with an idea to make searching for information more relevant and rewarding has grown from a small startup to a large and ever more influential company in the online (and business) world. Users are often frustrated when searching for information if they cannot find what they are looking for. This can be due to a number of reasons such as too many results (in other words a lot of unrelated or low value information returned). Google took steps to remedy that by filtering irrelevant information through their PageRank techniques. Another problem, which the search interest system tries to address, is not knowing exactly what to look for. In other words, not knowing what related terms to use when searching to produce the best results.

A search engine does not intuitively know what a user is looking for when searching when a user enters their search terms. It can only produce what it determines to be the best matches for the given query. The Search Interest System attempts to address this by helping the user find related terms for a given query which can be used to both refine and/or broaden a search. The interactive, guided IR system has the potential to elicit search requirements more effectively.

Potential uses for the Search Interest System include the following:

- Research Application: Can be used for collaborative searching between researchers to avoid duplication of exploratory effort as well as producing a record of past searches.
- Learning Application: Can be used by students (or anyone) to get an overview of a domain and learn what concepts are related.
- Marketing Application: Allows for search interest modeling so that marketing for products/services can be effectively targeted.
- Pure Search Application: Help the user to decide what to search for and provide decision support when forming queries.

This is an extension of the prototype system inorder.org [INORDER]

1.3 System Description

The Search Interest System allows a user to find (and define) related terms for a given query. These results become part of a growing ontology, which is in turn available to subsequent users/searchers. This functionality is described at a high-level in this section and will be further broken down into components (agents) and described in detail in later sections of this document.

At the highest-level search interest system is used to collect the opinions of people interested in a given domain. This is accomplished by first creating or defining the domain, which is automatically populated with a number of possible related terms/topics. Users can then further refine the domain by selecting terms of higher interest or those terms which are deemed to be more related to the given topic. In this way the system aggregates the semantic selections of unacquainted researchers.

The end result of this refinement of the original domain should be a hierarchy of related terms/topics, which as a result of the process should be of high quality and show a good relation to the original domain. This resulting list can be used as a keyword recommendation service for later searches on the subject/domain. A user performing a general search and unsure of what terms to enter to provide better search results could then make use of the Search Interest System. The system would help the user to find additional terms to search on or perhaps create a new query altogether. Results of queries would automatically be downloaded and cached so the user can easily browse the results of their search.

The Search Interest System can also provide information on other users performing related searches or refining similar domains. In this way users can explore and contact a community of others users with related interests and further their research.

The previous description gives rise to the following components of the Search Interest System. The relationship between these components is described in *Figure 1*.

- Search query expansion/reformulation
- Keyword Recommendation
- Caching Browsing Assistance
- Automated document classification
- Social networking based on the similarity of search interests

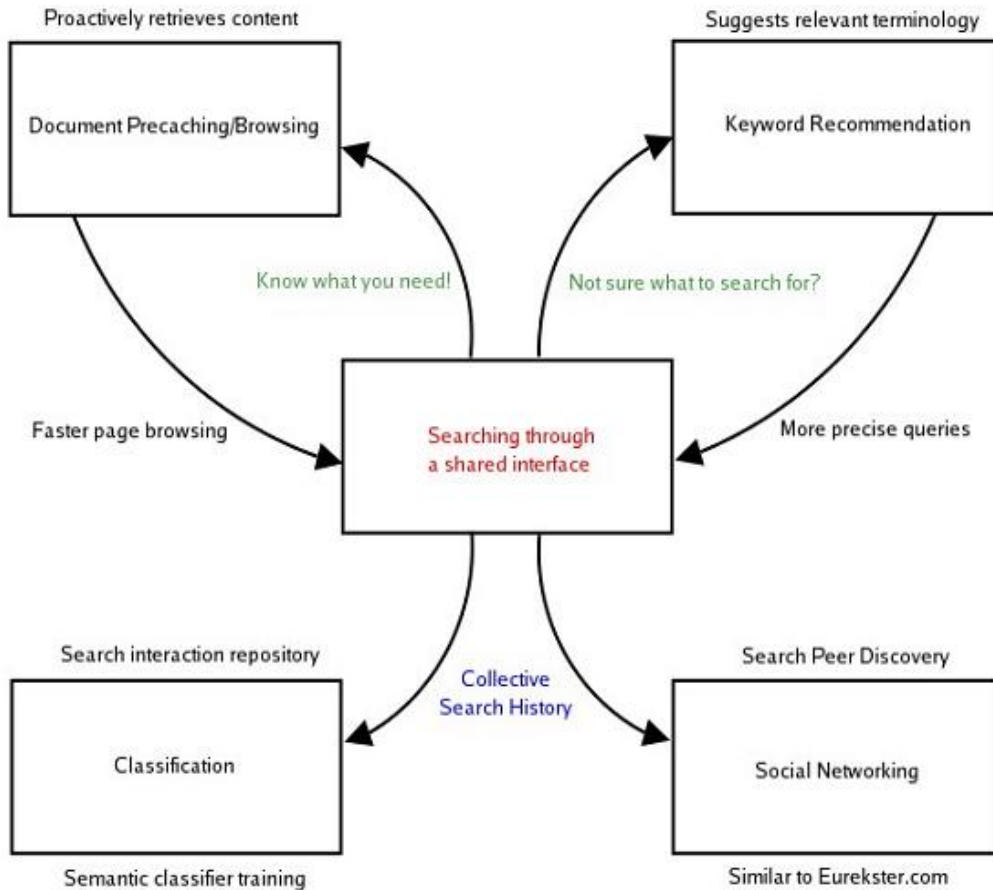


Figure 1 - Relationships between Search Interest System Major Components

1.4 Assumptions

The following assumptions have been made in the design of the Search Interest System.

- 1.4.1 All user input will be through a standard web browser/web interface
- 1.4.2 Uses select and vote on terms of interest according to their search requirements.
- 1.4.3 System has access to database(s) with document collection(s) of sufficient size
- 1.4.4 All services use the same protocol, HTTP over TCP/IP.
- 1.4.5 Some level of trust is maintained to avoid spamming/low-quality data contribution.
- 1.4.6 Users understand the purpose/workings of the system, ie a click is a vote for domain relevance.

1.5 Requirements

The following requirements have been developed for the Search Interest System.

- 1.5.1 The system will delivery keyword suggestions for a given query.
- 1.5.2 The system will work across and within any knowledge domain (given a common interface). For example, the system will work across a general knowledge base as well as more specific ones such as a medical or professional database of some sort.
- 1.5.3 The system will allow fast summarization of searched document collections.
- 1.5.4 The system will deliver a list of online (who may possess useful related knowledge).

1.6 Wish List

The following items have been identified as a wish list of items for future development to enhance or extend the functionality of the system specified here.

- 1.6.1 Real-time semantic clustering and recommendation.
- 1.6.2 Client side implementation to reduce server load and potentially also include local information in search results.
- 1.6.3 Distributed crawling (Grub Project)
- 1.6.4 Advanced preferences for privacy and knowledge sharing

2 System Architecture

The following models give an overview of the organization and goals of the Search Interest System. The concepts are borrowed from MESSAGE [MSG], which defines extensions to the UML for modeling of multi-agent systems. These UML diagrams give a high-level view of our system help to organize the concepts.

2.1 Organization

The following diagram in *Figure 2* is an organization view for the SIS that focuses on the structural relationships between entities. It shows how the knowledge management system and any administrators/moderators are part of the SIS and by extension the various search engines/systems.

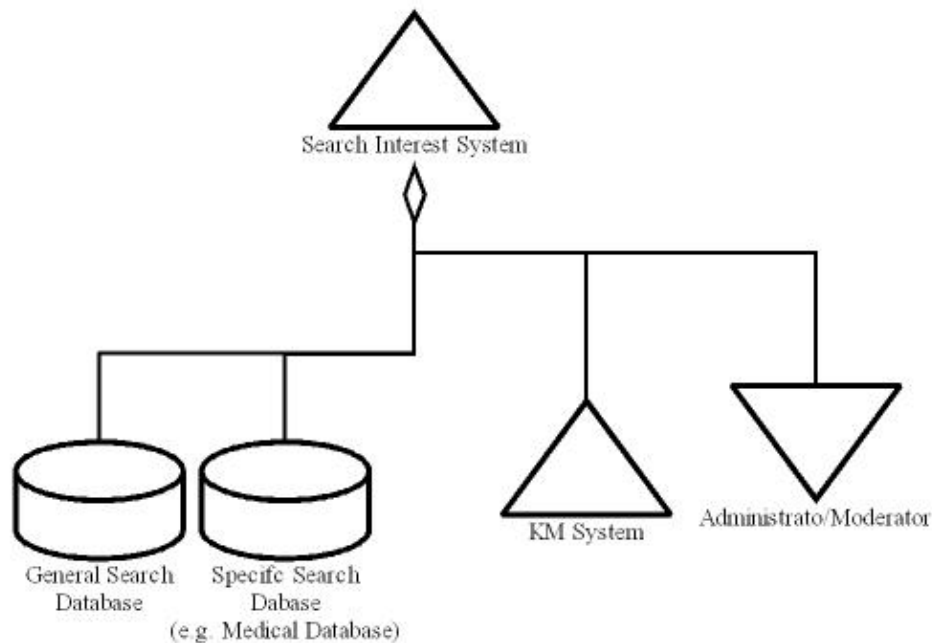


Figure 2: Structural Organization Diagram

2.2 Goals

The following *Figure 3* describes the goals of the system and how they are in some ways both sequential and complementary to the end goal of assisting the user with their searching. We have parallel goals of providing better search terms as well as informing the user of others online with similar interest and possibly providing related news/ad content. To achieve the goal of better search terms involves the parallel and sequential goals of finding and searching various databases. The end result is a hierarchical mapping of the goals of the SIS.

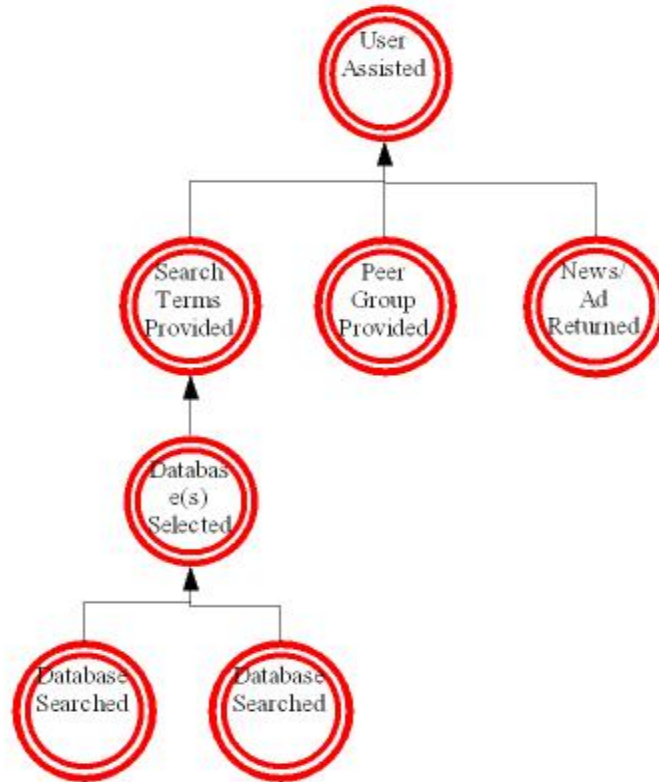


Figure 3 Goal/Task Implication Diagram

2.3 System Architecture

The way the system is envisioned to work is as a layer of service(s) above and beyond the current available internet search system (we will assume Google is *the* internet search system for the purposes of this document). A user who is attempting to search on Google with limited success can turn to the Search Interest System to help guide their search. As Google offers [Google Appliances](#) [GA], which can be attached to intranet's and other networks we will assume that all databases the search interest system will query/connect to will be accessed through a Google API [GAPI].

All agents will be available as web services available over the http protocol using simple http GET requests for communication. *The DB Agent will communicate with Google and Google Appliances through Google's proprietary API.* The end user will be presented with a standard web interface through a browser and will also be implicitly send their requests to the various services over http. *Figure 4* below give an overview of the architecture of the system.

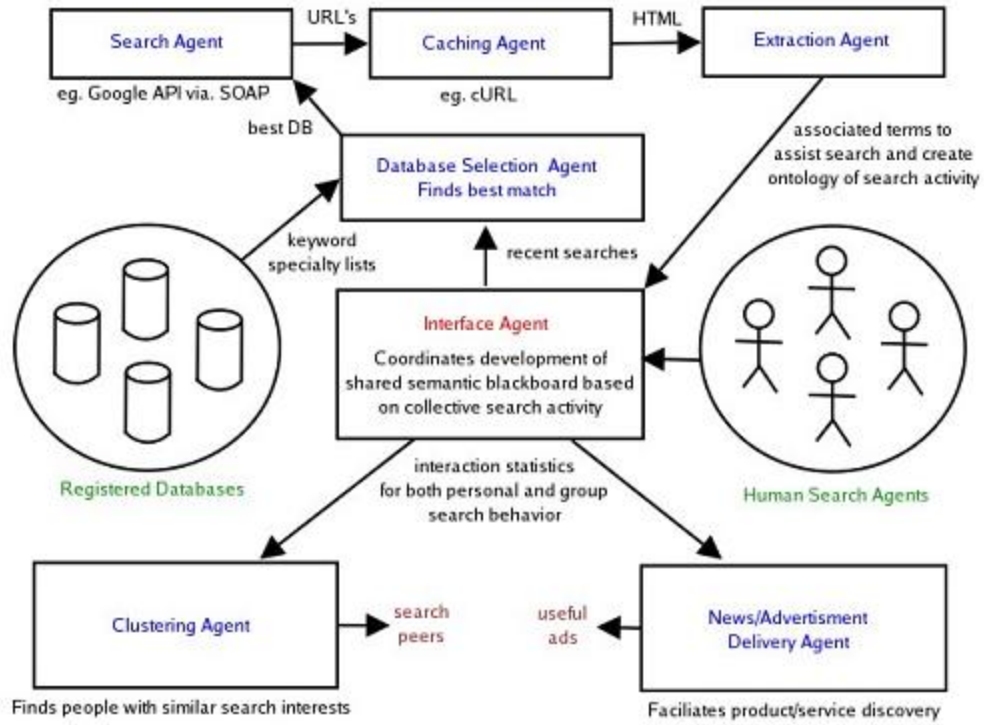


Figure 4. System Architecture

3 System Design

3.1 Agents

The Search Interest System is made up of the following agents or agent types:

- **DB Selection Agent**
- **Search Agent**
- **Caching Agent**
- **Extraction Agent**
- **Interface Agent**
- **Clustering Agent**
- **Browsing Agent**
- **News/Ad Delivery Agent**

Each of the agents has different purposes and provides a different service within the Search Interest System, which are described in the following sections.

3.1.1 Search Agent

The Search agent is responsible for coordinating interaction between the Interface, Clustering and Extraction agents and the KB selection agent. It contacts search engine returned by the KB selection agent to acquire sets of URL's which are most relevant the supplied query. This agent is able to contact a variable number of search engines, gathering URL's from a single engine search or performing a meta search which combines results from several search engines. The Search agent may request results from the top 1,3 or 10 most appropriate registered search engines, and then rerank the combined results according to the reputation and/or expertise of the knowledge source.

The Search Agent uses a SOAP-based API to communicate with each search engine the KB selection agent returns. [SOAP] The search agent invokes the appropriate API for each engine the KB agent instructs it to use, collecting sets of 10-50 URL's from each engine for subsequent aggregation and relevance ranking. These combined results are passed on to the Caching agent or Interface agent for to be downloaded and/or displayed to the user.

3.1.2 Database/Knowledge Base (KB) Selection Agent

This agent is a true agent in that is expected to 'put some thought into' selection of which databases to search given a user's query. In other words there is an autonomous nature to the agent in how it achieves its purpose and delivers its service. The KB Selection agent will be responsible for knowing about all available databases (each database will have a corresponding Search Agent). It will also maintain a set of statistics for each database such as available content, size, speed, and reliability. Based on these statistics the agent

can make a knowledgeable decision as to which database to query and return search terms from.

The KB Selection Agent will be the single point of contact for the Search/Caching Agents (described in this section) for forwarding queries and returning search results. The agent will receive a query request, select the appropriate database(s) and forward the query. The agent will then wait for all query results to return (or possibly not given a time limit in which case it would update the statistics for the tardy database) and then forward the query results back to the caller.

3.1.3 Caching Agent

This agent has the simple responsibility of caching the pages returned by the various queries received from the Search Agent. This is done for two reasons. The first and most important is so that the Search Agent has a place to go to for pages when searching for common terms. This also allows the user to view the actual results (vs. just the terms) of a given search. This will allow the user to browse or flip through results much quicker than individually selecting and downloading returned URLs.

When the DB Agent has returned a list of results the Caching Agent will begin to download in parallel the pages identified by the returned query. It will manage timing out those results that are too long in returning as well as periodically refreshing or flushing various items in the cache in order to keep it both relevant and up to date.

3.1.4 Clustering Agent (Peers) & News/Ad Agent

The next two components are very similar in that they are concerned with returning related objects. Their resulting purpose and functions as Agents will be very similar so will be described that way. Although the low level implementation will require separate agents

Clustering Agent (Peers)

The Clustering Agent is concerned with identifying the peers of a given user based on search criteria. It will access the group knowledge base to identify those peers based in individual semantic trails. The Agent will compare those semantic trails based on a cosine similarity algorithm and return as list of closest matches within a given context. The agent will be smart in that it may initially choose to search based on a narrow given context and then broaden that context to find a suitably sized (while relevant) group of peers. It should learn overtime what the parameters and types matching return the best resulting group of peers.

News/Ad Agent

This Agent is very similar to the Clustering Agent in that it will analyze a user's queries and in parallel attempt to identify related items from advertising and news content. This Agent will attempt to identify this content based on a number of factors and over time

should also learn which parameters return the best results and therefore the highest number of click-throughs on the returned content.

The Agent will scan news feeds (RSS, Atom etc.) to gather, evaluate and display relevant news. Links to products/services (Ads) will also be displayed based on target keywords.

To determine selections made by both of the previous agents (Clustering, News/Ad) a common algorithm will be used to analyze first and foremost are the terms being searched on but items of interest may also include;

- *Personal search history*
- *Contextual – last few clicks within a current context*
- *Behavioral - click frequency within all contexts*
- *Common attributes of items that were returned AND selected from previously*

3.1.5 Interface Agent

The interface agent coordinates the collection, aggregation and sharing of terminology among users of the search interest system. Its purpose is to provide a user-friendly means for web users to share semantic knowledge and focus their search efforts. The interface agent allows large groups of anonymous and unacquainted individuals to share their search experiences and pass on useful information to those with similar informational needs. Through its hybrid conceptual-document knowledge sharing approach the interface helps users explore online resources more efficiently by incrementally refining search criteria and improving the relevance of retrieved results.

The interface agent is responsible for not only coordinating the exchange of concepts and documents but also the moderation of the shared knowledge base which is constructed through collective search activity. Maintenance of this semantic blackboard is performed through term selection and voting actions respectively. When a suggested term (one extracted from web documents) is clicked for the first time within a group it is added to the currently selected topic. The interface also provides + and – buttons for each clicked term within a group to adjust its group-wide ranking. This 7-level ranking system adjusts the color of the suggested search term to make it more or less noticeable, letting users share their opinion on how relevant the keyword within the current search context.

The Interface agent primarily interacts with the Extraction agent to generate keyword recommendations. It displays terminology which is extracted from web search results and displays hypertext links to screen, which enable one-click query refinement and search ontology modification. This allows human interaction with the interface to create a model of terminological interest with a given domain (the context of the group), which may assist search activity and track individual semantic preferences during the search process. This data may then be used to locate online peers via the Clustering agent, and receive personalized news and advertisements via the News/Ad agent.

3.1.6 Extraction Agent

The extraction agent is responsible for suggesting associated or related terminology to users for the purpose of refining search requirements. The extraction agent uses a web mining approach to extract useful keywords from search results and/or its internal cache, creating lists of associated semantic knowledge that helps a user recognize and select new knowledge, which is better suited to the search task at hand. Such suggested terminology is delivered to the Interface agent where it is displayed to the user to facilitate query refinement and ontology construction.

The extraction agent collaborates primarily with the Interface, KB selection and Caching agents to coordinate this web mining process. The data mining technique used to extract useful terminology is based on a term co-occurrence frequency approach that analyzes a corpus of HTML results. Terms that occur a few times in several of the results returned become highly ranked, identifying keywords within the search results which are representative of “themes” or “concepts” associated with the supplied query. These associated concepts are potentially useful semantic knowledge within the current search context (defined by the group and/or topic labels) whose relevance may then be validated through human interaction with the interface agent.

3.2 Use Cases For Agents

3.2.1 Search Agents

3.2.1.1 Contact KB selection agent, combine results from each selected KB

Use Case Name:	Contact KB selection agent, combine results from selected KB's
Brief Description	The search agent contains the logic which combines URL's returned from various registered databases selected by the KB selection agent and delivers them to the caching or interface agents.
Pre Condition	Waits for query and list of DB's from the KB selection agent
Post Condition	Set of URL's successfully delivered to the Caching agent
Process Steps	
1	Receives list of most appropriate search API's for a given query from the KB selection agent.
2	Contacts each supplied search engine via a SOAP based API with the current query to receives multiple sets of URL's in XML format.
3	Combines sets of URL's according to reputation/expertise of knowledge source and delivers them to Caching agent.
Relationships	
Initiating	KB selection agent, Interface Agent
Collaborating	Caching Agent, KB selection agent, Interface Agent
Other Diagrams	

Data Requirements	
	Query, DB list, URL sets, Merged URL set

3.3

3.3.1 KB Selection Agent

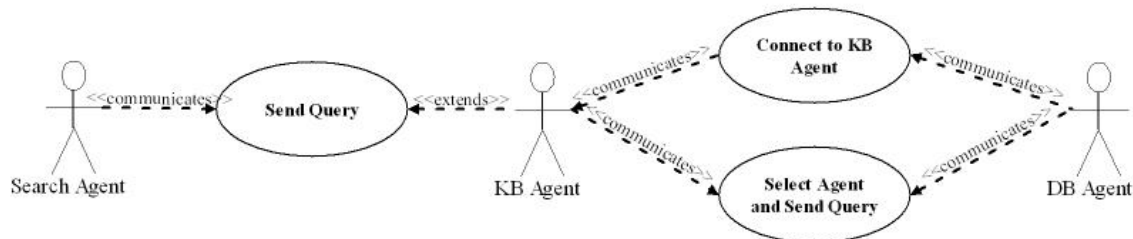


Figure 5. KB Selection Agent Use Case

3.3.1.1 Connect to KB Agent

Use Case Name:	Connect to KB Agent
Brief Description	How the DB Agent establishes a connection with the KB Agent
Pre Condition	DB Agent is not connected to the KB Agent, to be done once when DB Agent first initializes
Post Condition	The KB Agent has a connection or location for the DB Agent as well as high-level info.
Process Steps	
1	DB Agent sends a Connect command to the KB Agent
2	KB Agent responds with a Connect OK
Exceptions	
1a	Inadequate connection speed, resources etc. Connection is denied
Relationships	
Initiating	DB Agent
Collaborating	KB Agent
Other Diagrams	
Data Requirements	
	URL, Database size, Topics, Speed

3.3.1.2 Select DB Agent and Send Query

Use Case Name:	Select DB Agent and Send Query
Brief Description	Based on pre-established criteria KB Agent selects a DB agent and passes the query from the Search Agent. DB Agent processes query and responds with results.
Pre Condition	DB Agent is available and matches criteria for search.
Post Condition	DB Agent has responded in a timely fashion with search results.
Process Steps	

1	KB Agent applies selection query to pick DB Agent
2	KB Agent forwards query to DB Agent
3	DB Agent responds with results.
Exceptions	
1a	DB Agent does match criteria, not s
2a	DB Agent does responds in a timely manner
Relationships	
Initiating	KB Agent
Collaborating	DB Agent
Other Diagrams	
Data Requirements	
	Search Terms, List of URLs

3.3.1.3 Respond to Search from Search Agent

Use Case Name:	Respond to Search from Search Agent
Brief Description	This use case overlaps somewhat with 3.2.1.2, in that 3.2.1.2 is a result of this use case. Search Agent sends a query to the KB agent which responds with results from DB Agents.
Pre Condition	KB Agent is available
Post Condition	KB Agent has responded in a timely fashion with search results.
Process Steps	
1	KB Agent forwards query to DB Agent
2	DB Agent responds with results.
Exceptions	
2a	DB Agent does not respond in a timely manner
Relationships	
Initiating	Search Agent
Collaborating	KB Agent
Other Diagrams	
Data Requirements	
	Search Terms, List of URLs

3.3.2 Caching Agent

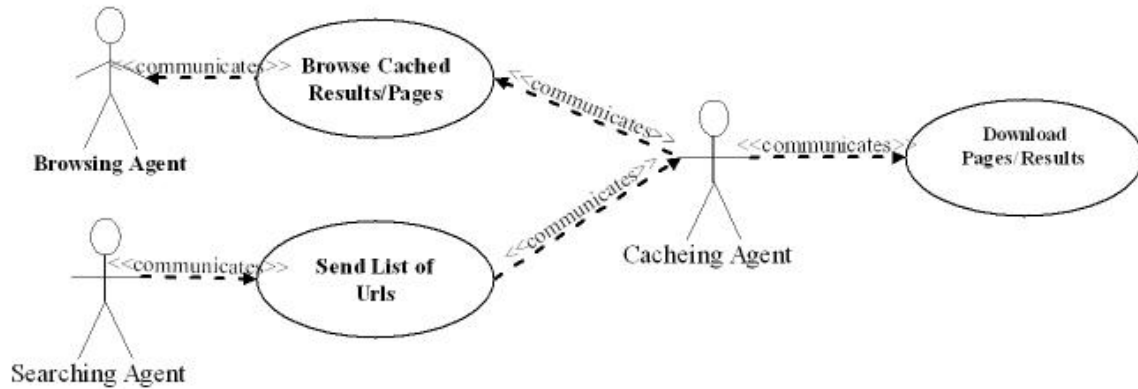


Figure 6. Caching Agent Use Case

3.3.2.1 Send List of URLs

Use Case Name:	Send List of URLs
Brief Description	Search Agent sends a list of URLs and a query id to the Caching agent which downloads/caches the pages.
Pre Condition	Caching Agent is available
Post Condition	Caching Agent responds with a set of downloaded pages.
Process Steps	
1	Search Agent sends request (URLs) to Caching Agent
2	Caching Agent searches attempts to either retrieve page from cache or download page from server
3	Caching Agent responds to Searching Agent with list of pages.
Exceptions	
2a	Page is not in cache (fall back to downloading)
2b	Server is not available; page cannot be downloaded. URL is flagged as not available.
Relationships	
Initiating	Search Agent
Collaborating	Caching Agent
Other Diagrams	
Data Requirements	
	Search Terms, List of URLs, Downloaded Pages

3.3.2.2 Browse Cached Results

Use Case Name:	Browse Cached Results
Brief Description	Browsing agent request the cached results (pages) from a previous query.
Pre Condition	Caching Agent is available and still has data in cache.
Post Condition	Caching Agent responds with a set of downloaded pages.
Process Steps	
1	Browsing Agent forwards original query to Caching Agent

2	Caching Agent responds with results (pages) of original query
Exceptions	
2a	Page is not in cache (fall back to downloading)
2b	Server is not available; page cannot be downloaded. URL is flagged as not available.
Relationships	
Initiating	Browsing Agent
Collaborating	Caching Agent
Other Diagrams	
Data Requirements	
	Search Terms, Cached Pages

3.3.3 Clustering/News/Ad Agent Use Case

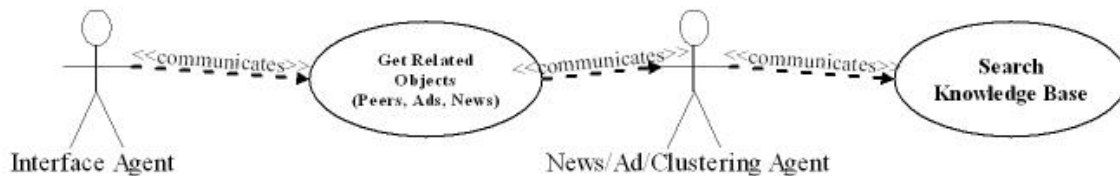


Figure 7. Clustering Agent Use Case

3.3.3.1 Browse Cached Results

Use Case Name:	Get Related Objects
Brief Description	Based on various attributes of the current user return related objects such as online peers, news and targeted ads
Pre Condition	Clustering Agent is available
Post Condition	Caching Agent responds with related objects
Process Steps	
1	Interface agent sends request to clustering agent with user id
2	Clustering agent looks up various attributes of user in knowledge base including past queries. Past peers and clicked on ads, news etc.
3	Clustering agent uses matching algorithm to determine best matches in various categories including; peers, news and targeted ads.
4	Clustering agent responds with content
Exceptions	
2a	No history for user, nothing returned

Relationships	
Initiating	Interface Agent
Collaborating	Clustering Agent
Other Diagrams	
Data Requirements	
	User ID, Related Content

3.3.4 Interface User cases:

3.3.4.1 Make or Select Topic

Use Case Name:	Make or Select Topic
Brief Description	Generates and/or displays a set of weighted terms associated with the topic label to establish the context of interest within the search group.
Pre Condition	Selection of a group in which the topic should be created.
Post Condition	Presentation of extracted and clicked terms within context of the topic label. This suggests terms related to the topic label which may assist the search process.
Process Steps	
1	Check for existence of topic in internal semantic KB.
2	If topic exists read set of weighted terms from KB and display to the screen via dynamically generated HTML.
3	If topic does not exist, combine topic label with group label (combine two term sets) and send to the extractor agent. The extractor agent then generates the topic, writes the topic to disk, and notifies interface agent to display it.
Exceptions	
3a	If topic does not exist, and less than 20 HTML pages cannot be found by the Search agent which is called by the extraction agents, then notify user the topic cannot be created.
Relationships	
Initiating	Interface Agent
Collaborating	Extraction Agent if Topic does not already exist.
Other Diagrams	
Data Requirements	
	Topic label, weighted terms in MySQL db.

3.3.4.2 View and/or adjust term ranks

Use Case Name:	View and/or adjust term ranks
-----------------------	-------------------------------

Brief Description	Displays all terms within a group of a similar rank, sorted by time of recent modification. Allows voting on terms to adjust their ranks to higher or lower level (of a different color), which draws more or less attention of human participants. Rank is intended as a measure of the current consensus of semantic relevance within the given context.
Pre Condition	Existence of a term within a topic. Extracted terms which have not been clicked ever may not be voted on.
Post Condition	Modification and/or display of term rank status.
Process Steps	
1	User selects desired term rank level to view if desired. If no level is specified the default middle level (0) is shown which displays terms of all levels together.
2	User votes + or – on a term which exists within at least of the groups topics, incrementing or decrementing its rank on between +/- click rank.
3	The interface alters the color of the voted term to reflect the rank level. This adjusts the attention it will be given by future search participants.
Exceptions	
2a	+ votes to terms at the top level (as well as – votes to terms at the bottom level) do not alter the rank... the term remains at its initial rank.
Relationships	
Initiating	Human user via the Interface Agent
Collaborating	none
Other Diagrams	
Data Requirements	
	Term id, it's associated term rank.

3.3.4.3 Keyword Recommendation

Use Case Name:	Keyword Recommendation
Brief Description	Suggests keywords related to the current query using a web mining approach. This provides users with related ideas to help them realize connections between ideas they wish to explore, and facilitate the query reformulation process.
Pre Condition	An initial query to send to the extract agent
Post Condition	A set of weighted terms returned by the extraction agent which may alter the current query and/or be added to the internal semantic blackboard with a single mouse click.
Process Steps	
1	Send current query to the extraction agent.

2	Extraction agent invokes search, cache and KB selection agents to create a corpus of HTML related the query (top ranked documents from the most appropriate search engine) which may be mined for co-occurring terms.
3	Extraction agent writes these terms to the MySQL database and notifies interface agent to display them to screen for selection by the user.
Exceptions	
2a	If the search agent (called by extraction agent) cannot return a corpus of at least 20 HTML pages notify the user that keywords cannot be recommended for the current query (because few web documents exist containing the current combination of terms)
Relationships	
Initiating	Human user via the Interface Agent
Collaborating	Extraction agent.
Other Diagrams	
Data Requirements	
	Current query, lists of terms returned from extraction agent. HTML corpus from search agent needed by the extraction agent.

3.3.4.4 Reformulate Query

Use Case Name:	Reformulate Query
Brief Description	Alters the users given query as terms are clicked within the group. As more appropriate terms are selected the query is incrementally refined to better reflect user search needs.
Pre Condition	Initial query, consisting of 0-4 search terms
Post Condition	Modified query consisting of 1-4 search terms, and a modified term list for a the topic from which the refinement was selected
Process Steps	
1	User selects a term which more accurately reflects
2	Term is added to a queue of most recent term selections. Once the queue reaches a length of 4 (4 selections) oldest terms are replaced according to a LIFO replacement policy.
3	Selected terms are also added to currently selected topic if they do not already exist within it.
Exceptions	
Relationships	
Initiating	Human user via Interface Agent
Collaborating	Extraction Agent
Other Diagrams	
Data Requirements	
	Query, Topic term list returned by extraction agent

3.3.4.5 Browse Related Pages

Use Case Name:	Browse Related Pages
Brief Description	Present user with sequential set of top matching pages based on their last few clicks. These pages are either validated results which have been bookmarked by past group participants, or top search results from the most appropriate search engine. As users flip through these recommended pages, quality results may be bookmarked for the benefit of future searchers.
Pre Condition	Current Query, Initial set of bookmarks for that term set.
Post Condition	Final set of bookmarks, which differs from initial set if any new browsed URL's are bookmarked during the browsing session.
Process Steps	
1	Check if any bookmarked pages exist for the given query. Create an array of up to 10 of the most popular (bookmarked most frequently) URL's which match the given query (word order not important, bag-of-words approach)
2	If bookmarked results do not exist, send the query to the Search Agent
3	Let the user flip through top 10 recommended pages and then return them to the semantic interface to encourage further modification of search requirements which will locate more relevant documents.
Exceptions	
2a	If the Search agent cannot locate 10 results, notify the user that no recommended pages can be found, and that the query should be simplified.
Relationships	
Initiating	Human User vs. Interface Agent
Collaborating	Search Agent
Other Diagrams	
Data Requirements	
	Query, bookmarks, URL's returned by search agent.

3.3.5 Extraction agent use cases

3.3.5.1 Create Group

Use Case Name:	Create Group
Brief Description	Create a new group that defines the overall context of search activity which will be performed by participants. This involves extracting a set of default keyword recommendation, which characterizes the domain and guides initial query reformulation activity.
Pre Condition	A requested group label. eg. "mobile agents"
Post Condition	A group with 1 default topic ready for human alteration.

Process Steps	
1	Interface agent send query to KB selection agent with new_group flag and instruct extraction agent to wait until caching agent has finished HTML downloads.
2	Extraction agent analyses a large HTML corpus (larger than for create topic use case) and produces ranked set of keywords associated with the group label.
3	Extraction agent writes new group and default topic to MySQL database and registers it so the interface agent may access it.
Exceptions	
2a	Corpus isn't of sufficient size to produce decent group because group label consists of terms which occur in very few web documents (less than ~ 50)
2b	
Relationships	
Initiating	Interface Agent
Collaborating	KB selection agent, Search agent, Caching Agent
Other Diagrams	
Data Requirements	
	Group label, URLs, HTML cache, Default topic for group with same label consisting of extracted domain terminology.

3.3.5.2 Create Topic

Use Case Name:	Create Topic
Brief Description	Create a topic within a group. This topic will consist of extracted terms for a HTML cache generated based on the group label combined with the topic label.
Pre Condition	A defined group and a requested topic label.
Post Condition	A new topic consisting of terms for query “topic label + group label”
Process Steps	
1	Interface agent sends requested “topic label + group label” to KB selection agent with new_topic flag and instructs Extraction agent to wait for the corresponding HTML cache to be created.
2	Extraction agent processes the cache to extract most relevant terminology.
3	Extraction agent writes weighted keywords to MySQL db and notifies interface agent it is ready for display.
Exceptions	
2a	Corpus isn't of sufficient size to produce decent group
2b	
Relationships	
Initiating	Interface Agent
Collaborating	Interface Agent, KB Selection Agent, Search Agent, Caching Agent
Other Diagrams	

Data Requirements	
	Group label, Topic Label, URLs, HTML cache, New topic

3.3.5.3 Create Related Terms

Use Case Name:	Create Related Terms
Brief Description	Produce a set of terms related to the given query which may assist query reformulation, but will not be a topic which other users can view.
Pre Condition	Selected group, requested topic label
Post Condition	New topic consisting of extracted terms for the current query. No group or topic terms are added to the query so that conceptual exploration may occur outside of the pre-existing context definitions.
Process Steps	
1	Interface agent sends query to KB selection agent, instructs extraction agent to wait for cache to be created.
2	Extraction agent processes cache to produce set of weighted keywords based on association to the supplied query and writes them to the MySQL db.
3	Extraction agent notifies interface agent to display the terms.
Exceptions	
1a	If a cache already exists for the given query skip to step 2.
2a	If the terms have been extracted previously skip to step 3.
Relationships	
Initiating	Interface Agent
Collaborating	Interface Agent, KB Selection Agent, Search Agent, Caching Agent
Other Diagrams	
Data Requirements	
	Query, URLs, HTML Cache, Weighted terms

3.4 Agent Internal Architecture

3.4.1 Example Agent Architecture

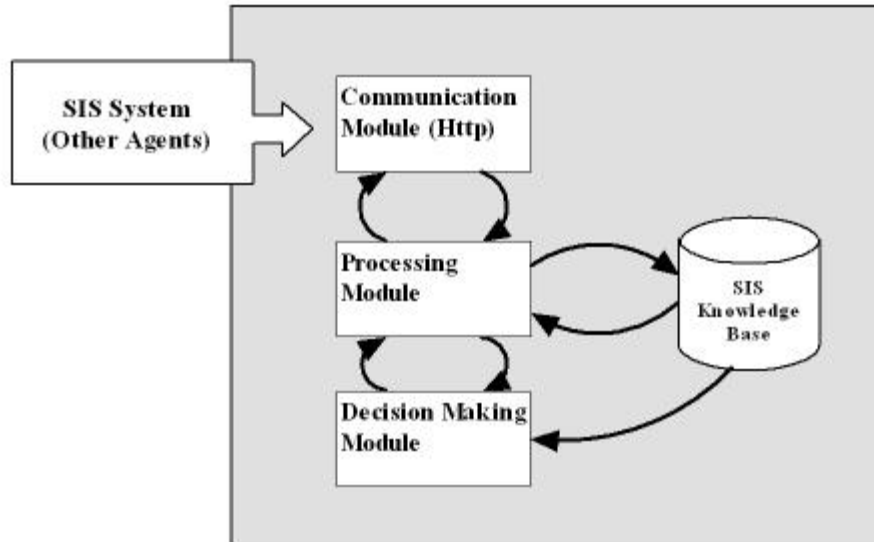


Figure 8. Example Agent Architecture

The above *Figure 8* shows an example architecture for all agents in the SIS System. Even though each Agent fulfills a different role their internal structure/architecture is very similar. Each agent will send and receive requests over http through its communication module. The communication will parse incoming messages as well as create outgoing messages that are handed to it by the processing module. The processing module will record incoming information about the request such as user info, query info and general logging to the knowledge base. It will also retrieve any related information needed for further processing. The final module, the decision making module will, the actual brain of the agent will then decide how best to fulfill the request. It does this by receiving the request info from the processing module and then querying the SIS knowledge base for any related info with which it needs to make a decision, then passing back the result to the processing and subsequently the communication modules.

For example, the KB Selection Agent will process requests (queries) from the Search Agent. The request will be parsed/translated by the communications module and then logged by the processing module. The request will then be passed to the decision making module which will query the knowledge base and make a selection as to which DB Agents to pass the request to based on various parameters. It will then formulate a new request to be passed on to the selected DB Agents.

3.5 Data Model

The following diagram, *Figure 4*, shows an E-R Diagram for the System Interest System's Knowledge Base. It shows how the resulting domain, topic, list and query data will be stored. It also shows the mapping of the resulting base of user's and databases queried.

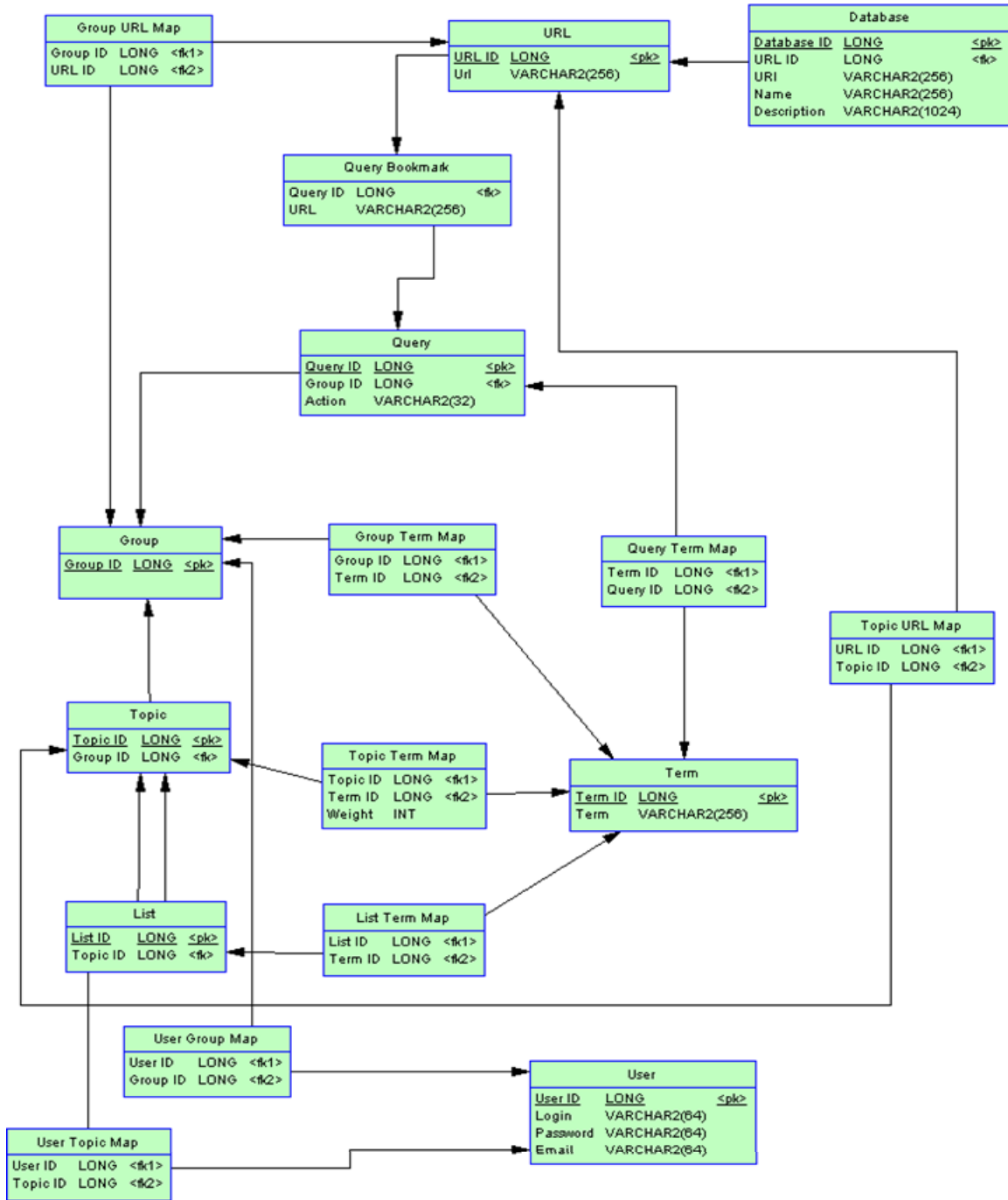


Figure 9. E-R Diagram for Search System Knowledge Base

4 Message Specifications

4.1 Message Protocol

All communication between agents is (or will be) done over HTTP using the SOAP protocol. SOAP was originally designed for communication between distributed applications. It defines the use of XML and HTTP to access services (agents) in platform independent manner making it a natural fit for the various agents of the SIS application. Agents can be transparently distributed or congregated on a single processor as needs.

4.2 Message Formats (Inter-Agent)

The following depicts message records to be sent between agents over the SOAP protocol between Agents in the SIS system. These records show a request and response from the Search Agent to the KB Agent and return:

Request

```
<query>
  <header>
    <queryid>Long</queryid>
    <userid>Long</userid>
  </header>
  <querydata>
    <queryword>String</queryword>
    <queryword>String</queryword>
    <queryword>String</queryword>
    <queryword>String</queryword>
  </querydata>
</query>
```

Response

```
<queryresponse>
  <header>
    <queryid>Long</queryid>
    <userid>Long</userid>
  </header>
  <querydata>
    <url>String</url>
    <url>String</url>
    <url>String</url>
    <url>String</url>
  </querydata>
</queryresponse>
```

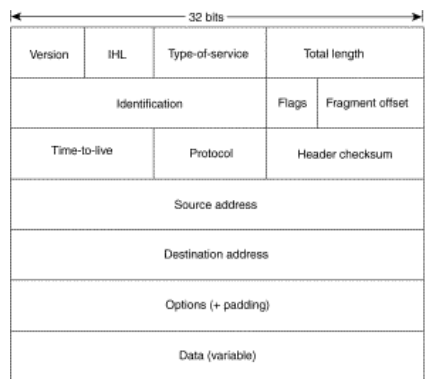
The records show how data will be passed between the Agents in generic xml format.

5 Technology Overview

5.1 HTTP over TCP/IP

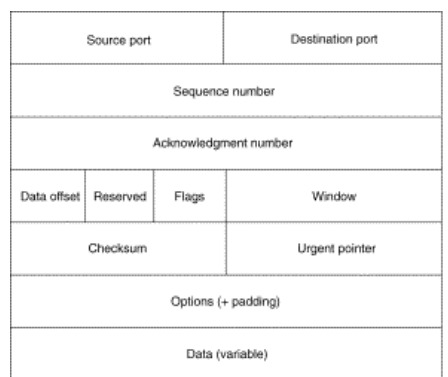
All communication between agents within the search interest system uses the Hypertext Transport Protocol (HTTP) over TCP/IP (Transport control protocol/Internet protocol). These protocols have become the defacto standard for transmitting textual content between web clients (or browsers) and web servers.

The IP protocol resides at layer 3 (or network layer) of the OSI networking model. It governs the transmission of 32 bit IP packets and has two primary responsibilities: 1. providing connectionless, best-effort delivery of datagrams through a network; and 2. providing fragmentation and reassembly of datagrams



IP packet structure

The TCP protocol resides at layer 4 (or transport layer) of the OSI model above the IP protocol, and is responsible ensuring reliable transmission of IP packets. TCP packets are comprised of 12 fields which include source and destination addresses, sequence and acknowledgement information as well as the actual data and a checksum for detecting transmission errors. Together with IP, TCP provides a robust system for transmission of binary data between internet hosts.



TCP packet structure

HTTP protocol resides at layer 5 of the OSI model, and is responsible for initiating and maintaining sessions between web clients and servers. Within the search system HTTP is used to transmit XML between the computational agents, and HTML for communication between the Interface agent to and humans using their web-browsers.

5.2 LAMP System

LAMP is an acronym for Linux, Apache, MySQL, and PHP. It is a term to describe the implementation of open-source tools to create interactive and dynamic websites. It is recognized as one of the most inexpensive and reliable ways to create and deploy web-based systems.

Apache is a multi-threaded webserver originally derived from the public domain HTTP daemon developed at the NCSA. When development of HTTPD stopped, webmasters began writing their own patches to add features and fix bugs in the server and the result has become apache. Apache is the number 1 webserver used on the internet.

MySql is an open-source RDBMS, which supports SQL, a standard high-level language used to query information and data from a database. The MySQL package contains a tool for administration, a client for creating and populating database tables, and a server daemon which listens for requests.

PHP is a scripting language especially suited for web applications. PHP code is processed on the server side, so all the web browser receives is plain html. PHP is easily embedded within your plain html and you are to turn it on and off with the use of a single tag.

5.3 XML

XML is a text-based markup language that is fast becoming the standard for data interchange on the Web. As with HTML, you identify data using tags (identifiers enclosed in angle brackets, like this: <...>). Collectively, the tags are known as "markup". SOAP, as described below makes use of XML to replicate data-interchange and function calls over HTTP.

5.4 SOAP

SOAP is an XML-based object invocation protocol. SOAP was originally developed for distributed applications to communicate over HTTP and through corporate firewalls. SOAP defines the use of XML and HTTP to access services, objects and servers in a platform-independent manner. SOAP provides a way to access services, objects, and servers in a completely platform-independent manner.

6 Design Decisions

6.1 Agent Roles

Agent's roles were originally defined at a very low-level with multiple agents performing similar tasks. During the modeling and analysis phases it was realized that agents could cover a much broader (and more generic) range of functionality.

A specific example is the roles of Clustering/News/Ad functionality. Separate agents were originally defined for each of these tasks but we decided that each of these were actually just a different view of the same task. In reality at a higher level we simply wanted to determine related items given a user and their defining attributes. We were then able to merge these tasks into a single role/agent with increased scope and usefulness.

6.2 Usability considerations

This search interest system was designed with the knowledge that its users would be from a wide variety of backgrounds and often have limited computer experience. As a result interaction with the interface agent was designed to be simple and compelling to encourage user participation, yet still address concerns found within sophisticated knowledge sharing systems. A primary decision was the labels or semantic notation used to represent different actions and sets of knowledge within the system. Simplicity of such factors is necessary to ensure a wide range of users could understand the organization and utility of the system.

One example is the use of the labels "topics" and "terms" to represent the ontologies of utilized search terminology. The understanding that "clicking a suggested term adds it to a topic and modifies my search terms" allows non-technical users to participate, whereas a more accurate description that "selecting extracted concepts from HTML corpora constructs open conceptual ontologies as query refinement activity occurs" would restrict participation to scientists and engineers familiar with knowledge engineering principles. Carefully considering how such semantic factors would affect user interaction allows the search interest system to be useful to average web users across a variety of domains.

6.3 Agent Architecture/Messaging

The original discussions on the underlying architecture for the agents and agent interaction described a much more rigid and involved architecture using JINI etc. Later discussions showed that this was more complex than was required or desired. As a result we arrived at the simpler implementation using XML/SOAP.

References

[MSG] MESSAGE, Methodology for Engineering Systems of Agents EUROSCOM

[GAPI] <http://www.google.com/apis>

[GA] <http://www.google.com/appliance>

[INORDER] <http://inorder.org/>

[SOAP] <http://www.w3.org/TR/soap/>