



Spatially-Aware Information Retrieval on the Internet



SPIRIT is funded by EU IST Programme
Contract Number: IST-2001-35047

Specification of System Functionality

Deliverable number:	D4 1101
Deliverable type:	R
Deliverable nature:	PU
Contributing WP:	WP 1
Contractual date of delivery:	01/01/03
Actual date of delivery:	15/01/03
Authors:	Dave Finch SPIRIT Technical Group

Keywords: Architecture, Design, Functional Specification, Prototype

Abstract: This document presents the proposed design and architecture of the SPIRIT prototype. It identifies the functional components of this prototype and defines the information flows between these components. It also addresses technical and implementation issues.

Contents

1.	INTRODUCTION	5
1.1.	Document Structure	5
	Diagram Conventions.....	5
2.	RUN-TIME BEHAVIOUR.....	6
2.1.	Introduction	6
2.2.	Pre-Processing.....	6
2.3.	Interaction.....	6
	Overview	6
	Query Disambiguation.....	6
	Query Expansion.....	6
	Result Presentation.....	7
	Search Refinement	7
	Control Features	7
	Logging Feature	7
2.4.	Alternative Configurations	7
3.	FUNCTIONAL BREAKDOWN	8
3.1.	Introduction	8
3.2.	Pre-Processing Components	8
	Overview	8
	Web Data Collection	8
	Ontologies	8
	Metadata	8
	Indexes	8
3.3.	Run-Time Components.....	9
	Overview	9
	User Interface.....	9
	Ontologies	9
	Metadata	9
	Indexes.....	9
	Search Engine.....	9
	Relevance Ranking.....	9
	Broker.....	10
4.	ARCHITECTURAL DESIGN	10
4.1.	Introduction	10
	Extensible Architecture	10
4.2.	Pre-Processing Architecture.....	10
	Ontologies	10
	Crawler	11
	Metadata Extraction	11
	Index Generation.....	13
4.3.	Run-Time Architecture.....	14
	Interaction.....	14
	Search Engine.....	15
	Relevance Ranking	15
	Query Refinement	15
4.4.	Overall Architecture	16

Broker 17

4.5. Alternative Configurations 19

 Google Web APIs 19

5. TECHNICAL ISSUES 20

5.1. Development Phases 20

 Prototype Phases 20

 Extensible Architecture 20

5.2. Implementation Issues 21

 Distributed Architecture 21

 Development Languages and Platforms 21

 Communication Protocol 22

 Geometric Operations 22

 Version Management 22

 Documentation 23

6. REFERENCES 24

APPENDIX A: INTERFACE DEFINITIONS 25

 Pre-Processing Procedure Calls 25

 Run-Time Procedure Calls 25

 Administrator Procedure Calls 26

APPENDIX B: FUNCTIONAL COMPONENT DESIGN DIAGRAMS 27

 Introduction 27

 Search Engine 27

 Ontologies 28

 Metadata 30

Document History

<i>Issue</i>	<i>Date</i>	<i>Remarks</i>
v 0.1	Oct/02	Initial Draft (Dave Finch)
v 0.2	Dec/02	Redrafted and reformatted following Tech. Mtg., IGN (DF)
v 0.3	Dec/02	Redraft for review version (DF)
v 1.0	Jan/03	Final version (incorporating referees review comments) (DF)
v 1.1	Jan/03	Final version (incorporating CBJ review comments) (DF)

Executive Summary

This document describes the prototype system that will be developed to demonstrate a spatially-aware search engine as part of the SPIRIT project. In it is defined the run-time functionality provided by the system, and the functional components and the interconnections between them that will realize that behaviour. Architectural, technical and some implementation issues are also addressed here. This document is intended to provide the blueprint from which the SPIRIT prototype will be developed.

D4 1101

Specification of System Functionality

1. Introduction

This document defines the system functionality, functional components, and architectural design of a prototype spatially-aware search engine that will be developed for the SPIRIT project (13). The User Requirements document (16) is intended to describe the user and functional requirements of the prototype. The purpose of this document is to:

- Define the run-time functionality that will actually be implemented in the SPIRIT prototype.
- Provide a functional breakdown of the components in that system.
- Detail the information flow between those components.
- Make recommendations on the proposed architecture, as well as covering technical and implementation issues.

The functionality of the SPIRIT prototype is defined on the basis of what is believed to be attainable within the timeframe and resources of the project. This set of "viable" functions is mapped to major functional, software components, taking into account the options provided by the evaluation of possible architectures (8). A design is presented here that specifies the communication between these software components (in terms of the flow of data between them), and details the communication mechanisms that will support this required interoperability. This report is intended as a blueprint from which the prototype can be implemented.

1.1. Document Structure

This document has the following sections:

- Run-Time Behaviour - defining the functionality that the prototype will present to the user.
- Functional Breakdown - defining the functional components which constitute the prototype.
- Architectural Design - detailing the information flow between functional components.
- Technical Issues - detailing issues of choice of architectural design and communication protocol, as well as addressing design standard issues (e.g. software and platform uniformity, design tools) affecting implementation and development, and documentation.

Diagram Conventions

The following conventions are used in the architectural diagrams within this document:

- Arrows and lines = data, information or request flow
 - Dotted - pre-processing
 - Solid - query time
- Shapes = functional components
 - Rectangle - program components
 - Circle - data components

2. Run-Time Behaviour

2.1. Introduction

The User Requirements document (16) records the functionality project members envisaged the SPIRIT system providing, as generated from their interaction scenarios. This section describes the functionality that will actually be implemented and be provided in the SPIRIT prototype, as determined by the timeframe and resources of the project. This functionality determines the nature of the functional components comprising the prototype and the interactions between these components (as described later in this report).

The functionality of the SPIRIT prototype can be divided into the following sections

- Pre-Processing
- Interaction
- Alternative Configurations

2.2. Pre-Processing

The SPIRIT prototype will require a pre-processing stage to process web data into a form that can be accessed by the run-time parts of the system. This pre-processing is intended to operate on both web documents and geo-datasets, and will either be instigated by an administrator or by a scheduled automated process (e.g. UNIX *cron* job).

The functionality that pre-processing comprises is invisible to the user, and therefore defined elsewhere in this document (see sections 3.2 and 4.2).

2.3. Interaction

Overview

The SPIRIT prototype will present a multi-modal user interface, offering interaction and presentation of results via textual and graphical interfaces:

- Users will be able to describe geographical locations textually, with the system managing terminology for place names, geographical areas and spatial relationships.
- Users will be able to mark the required extent of a geographical search on static or interactive (active) map displays, e.g. selecting displayed place or feature names, drawing areas of interest, using map features to delimit the region of interest.
- Use of a sketching interface will be demonstrated with the SPIRIT prototype, enabling simple drawings to be used for depicting geographical contexts.

Query Disambiguation

The prototype will identify ambiguous, spatial terminology in user queries, which the user will be prompted to disambiguate (e.g. Cambridge, UK or Cambridge, Massachusetts, USA?), prior to the query being processed by the system.

The prototype will identify ambiguous, non-spatial, domain-specific terminology in user queries, which the user will be prompted to disambiguate (e.g. the term "surfing" could indicate surf boarding or wind surfing), prior to the query being processed by the system.

Query Expansion

The prototype will employ automatic query expansion, i.e. the addition of terms to a user query to enable the retrieval of resources with:

- Alternative place names (e.g. historical forms, alternative names and spellings, etc.).
- Spatially-related place names (e.g. nearby in location).
- Fuzzy geographical areas (e.g. South Wales, North East London).
- Geographical concept expansion (e.g. constituent administrative areas of the "Schwarzwald").

Result Presentation

Results will be displayed either textually or graphically or both, according to user selection.

- Textual search results will be displayed in ranked order (as determined by SPIRIT's relevance ranking algorithms).
- Graphical results will be output with ranking annotations on a variety of possible displays, e.g. 2D or 3D visualizations, cartograms, etc. Use will also be made of active and static maps. The interactive (active) map feature will play an important role in presenting, and allowing interaction with, the results of the search (see *Search Refinement* section below).

Search Refinement

The user may select from the ranked set of results those that they consider are of most interest (e.g. on the basis of most appropriate, best fit to their initial query, interesting avenue to further explore, etc.). This set of results will be processed to form a follow-up query that may better describe the user's initial query or describe a line of investigation they would like to pursue further. This query will be sent back to the search engine and processed in the same manner as the original query, with ranked results displayed accordingly.

Control Features

The user interface will provide means for turning on and off certain features when processing a query. This facility will enable features of the system to be demonstrated, and will readily allow evaluation and comparison of different techniques, e.g.

- Query disambiguation (on/off).
- Query expansion (on/off).
- Spatial and Textual Index or just Textual Index (for result comparison purposes).
- Different Spatial and Textual Index implementations.
- Selection of different relevance ranking criteria/different relevance ranking algorithms.
- Selection of manner of presentation, e.g. type of graphical output (static or active map, other format), number of results displayed, etc.
- Use of Google Web API to produce results rather than SPIRIT search engine (see section 2.4).

Logging Feature

The SPIRIT prototype system will allow logs of user sessions to be produced. These will provide a record of system behaviour across all interactions and for individual sessions.

Further administrator functionality may be provided (e.g. update and maintenance of system components), but for the purposes of the prototype these are likely to be back-office functions to be executed off-line by developers.

2.4. Alternative Configurations

The functionality of the SPIRIT prototype will be provided by the functional components defined within this document. For testing and comparison purposes, a test configuration will be provided for the text-based search functionality. This test configuration will access the Google search engine index instead of the SPIRIT indexes, using the Google Web APIs (see section 4.5).

3. Functional Breakdown

3.1. Introduction

This section is intended to identify the individual components of the proposed system. This will be at a level necessary to provide the run-time behaviour described in Chapter 2.

3.2. Pre-Processing Components

Overview

These components comprise the background parts of the system that pre-process web documents and geo-datasets, obtaining data in a format that:

- has a condensed view of their textual or data content.
- identifies their geographical content and allocates a geographical footprint accordingly.
- can readily be searched by the information retrieval (IR) Search Engine when queried.

Web Data Collection

The SPIRIT prototype will be demonstrated on a 1Tb collection of web data (stored on a cluster of 25 PC's). This large set of data will exercise all aspects of the spatially-aware Search Engine functionality and provide a realistic environment for testing and developing the prototype. It will also enable analysis of different data storage mechanisms that may increase performance, e.g. parallelization of collection.

Ontologies

This component will hold structured geographical and domain (i.e. application-specific) information. For demonstration purposes, ontologies will be developed for a target geographical area and a specified application area. These will be used to prove the concepts behind their design and development.

Geographical Ontology

This will model both the vocabulary and the spatial structure of places for the purposes of information retrieval. This ontology will contain a structured hierarchy of geographical terms and spatial relationships between them (10).

Domain Ontology

This will contain structured information on a specified application area (domain) to enable recognition of non-spatial terminology used in association with geographical terms in user queries.

Metadata

This is a repository of data characterizing the geographical content of documents in the Web Data Collection. Each document is analyzed to identify the geographical terms or identifiers they contain, and these are then mapped (using the geographical ontology) to a geographical footprint. This is used to indicate the geographical and spatial frames of reference of the documents in the collection, being used in the generation of the Spatial Index and in determining the relevance of documents to the geographical terms in the initial query.

Indexes

Textual Index

This is the repository of term incidence and frequency within documents comprising the Web Data Collection. All documents are analyzed and an index created of the terms and their frequency within them. This index will be used to match terms in the query with the set of documents that have the largest incidence of those terms.

Spatial Index

This is a repository spatially referencing the documents that comprise the Web Data Collection. All documents are analyzed and allocated a geographical footprint during the Metadata extraction process. This information is used to allocate documents to positions in the Spatial Index, with documents referring to related geographical areas being grouped together in the Index (see section 4.2). This Spatial Index will be used to match documents that overlap or are contained within the query footprint (the geographical area(s) referred to in the query).

3.3. Run-Time Components

Overview

These components comprise the parts of the system involved in the input of user queries, the processing of those queries, their passage through the system to generate a set of results, and the presentation of those results to the user.

User Interface

This component will provide a multi-modal interface for geographical information retrieval, allowing interaction through Textual and Graphical Interfaces. A Sketching Interface will also be developed and demonstrated. Both Textual and Graphical Interfaces will be employed for the formulation of queries, the presentation of results, and the refinement of queries in the light of those results.

Mapping API

It has been proposed that the SPIRIT architecture be provided with a link to a mapping service (e.g. NetSolut On-Line, (15)), which would provide a web mapping engine that could be built into the system through an API, thereby providing a new medium for the presentation of output. This functionality is part of WP4: User Interface, and is proposed to be provided by a Web Feature Server, using Open GIS calls for mapping.

Ontologies

This component, defined in section 3.2 above, is involved in the query expansion process, adding terms to the initial query by determination of alternative and spatially related place names. The role of the Ontologies in these processes is fully detailed in section 4.3 below.

Metadata

This component, defined in section 3.2 above, is involved in run-time relevance ranking (the retrieval and return of previously determined geographical footprints for the identified set of documents best fitting the user query). The role of Metadata in this process is fully detailed in section 4.3 below.

Indexes

This component, comprising Textual and Spatial Indexes, is defined in section 3.2 above. It is involved in the run-time retrieval of documents that best match the initial query (a process that is described in section 4.3 below).

Search Engine

This component provides the core information retrieval functionality of the system, accessing the pre-processed indexes to obtain matches with user queries. This functionality will be based on the GLASS Search Engine, a simple, text retrieval system, providing utilities for indexing, searching and evaluation (4). The open architecture and modular approach used in its implementation mean that it should be readily customizable to introduce new functionality, i.e. addition of spatial features to the current, text-based ones.

Relevance Ranking

This component is responsible for ranking the set of results (identifying web pages) received from the search engine component on the basis of their relevance to the initial query. These

results will be ranked on their textual and spatial relevance to the query, utilizing newly developed similarity measures. Ranked results are sent to the User Interface for presentation to the user (either textually or graphically).

Broker

The Broker component is intended to offer the following functionality:

- Provide a central point of control, which will result in a more secure system.
- Provide a central session manager, which will enable the recording of all interface actions and responses from components into a single, central search session log (which is essential for debugging, monitoring, and testing activities).
- Provide a scheduling activity, which will manage transport of messages between components. This is aimed to aid development, in particular minimizing component mismatches during development.

4. Architectural Design

4.1. Introduction

This section is intended to represent the information flow between the functional components identified and defined in the previous section. As such, this deals with the connectivity between those components, defining which components need to call which other components, and will form the basis for detailing what information will actually be requested and sent by the various components.

Simple architectural diagrams and UML Sequence Diagrams are used to illustrate interaction between functional components and flow of information through the system. UML Class Diagrams and Activity Diagrams illustrating the functional breakdown of individual components and their proposed interaction with other components within the prototype are shown in *Appendix B: Functional Component Design Diagrams*.

Extensible Architecture

The proposed architecture is intended to be extensible to accommodate all proposed development phases of the project. A "plug-and-play" type architecture is described here which should allow the phased developments inherent in the proposal (see section 5.1). This architecture is designed to accommodate evolutions to components in the prototype, e.g. different ranking techniques plugging into the same "Relevance Ranking" functional component, more sophisticated reasoning residing in the geographical ontology component, etc.

4.2. Pre-Processing Architecture

Ontologies

The Geographical and Domain Ontologies are involved in the pre-processing of web pages to produce metadata. In particular this entails the allocation of geographical footprints describing the geographical area that are mentioned in individual web documents (see *Web Page Enrichment* section below). Each web page is analyzed individually and any geographical terms (place names, geographical areas, etc.) extracted. Each of these terms will have an associated geographical footprint associated with it stored in the Geographical Ontology. These footprints are retrieved and, on the basis of them, a new footprint is allocated to the web page in question.

The design and functionality of the ontologies is described in full elsewhere (10).

Crawler

The 1Tb Web Data Collection is a static collection, but the intention is that SPiRiT will be applied to a wider and more dynamic set of data (i.e. the WWW, or a sub-set of it). A number of pre-processing tasks are to be performed on the available web data, i.e. generation of Metadata and both the Textual and Spatial Indexes, and the extraction of geo-features from geo-datasets. A central repository will first be produced to serve as a record of the documents that need to be visited in the pre-processing activities.

The Crawler component is responsible for this, navigating the web collection and generating a record of the sites it comprises. For the Web Data Collection, this functionality will be provided by negotiating the directories and documents constituting the collection and recording those in a database.

For the WWW (or sub-set of it), production of a repository of URL's (a link database) will ensure consistency in generation of Metadata and Indexes from the same initial set of data. This functionality would be provided by a conventional Web Crawler. From a given starting point (a web page), such a crawler would follow every new link to pages within the WWW (or sub-set of it) and follow-up pages from the same site to traverse the complete collection. It would do this by issuing HTTP requests to access sites, recording the URL's of every visited site in the link database.

Figure 1 below illustrates the interconnections between the Crawler and the other pre-processing components.

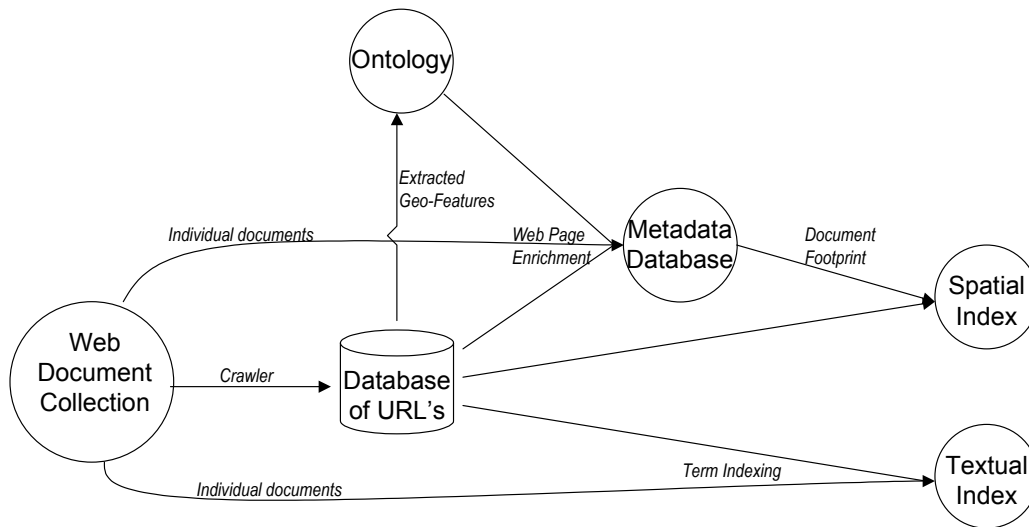


Figure 1 Metadata Extraction and Indexing (Interconnections)

Metadata Extraction

Web Page Enrichment

The web page enrichment (or annotation) activity comprises the identification and characterization of geographical terms within a textual web document, the allocation of geographical footprints to individual documents from these terms (i.e. the area which the

document is determined to encompass) and their storage in a metadata repository. This metadata is then used in the generation of the spatial index (determining how a document will be spatially referenced) and in relevance ranking (determining how closely a document's geographical footprint matches a query footprint).

Geo-Data Enrichment

The geo-data enrichment (or extraction) activity comprises the process of extracting implicit and explicit information from geo-datasets, including:

- Geographical footprints of the area the dataset encompasses and a textual description (label) of that area. These items will be stored in the Metadata repository enabling the search engine to reference the dataset in response to a query. The value is that users are informed of the availability of such data which may be of benefit to them in relation to their query (without this extraction, datasets could get bypassed in the search process). The user is then free to investigate these sources off-line from SPiRiT (organizing any licensing or registration issues regarding access to the datasets themselves).
- Geographical features that are depicted in the dataset, e.g. settlements, areas of vegetation, bodies of water, etc. The datasets will be mined to identify and extract such features. These features constitute enriched geo-data and differ from the geographical footprint metadata described above. As such it will reside in a different repository as the data it contains is in a different form and will be processed differently. To preserve the semantics of the extracted data, this repository will be modelled as an ontology. The aim is that data extracted from geo-datasets will provide an automatic feed into the SPiRiT Ontology (complementing the initial plan of generating this from gazetteers and thesauri).

Figure 2 below illustrates the order of information flow among the pre-processing components, in particular web page enrichment, geo-data enrichment and index generation.

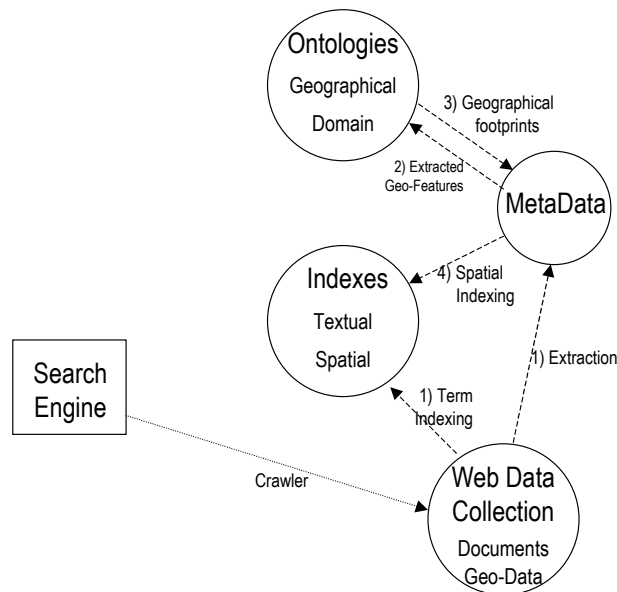


Figure 2 Metadata Extraction and Indexing (Information Flow)

Index Generation

Spatial Index

A Spatial Index will be added to GLASS's current text-based information retrieval mechanism (4). One implementation would be to provide a filter on the term index, identifying a subset of the overall document set through which to search for relevant matches to the user's query. A simplistic description of this process (shown in Figure 3) involves the following steps:

- Divide geographical area into cells (the size of these cells could depend on the area they cover and the amount of data associated with it; different resolution cells could be used to cover areas with a large number of references, e.g. cities).
- Establish which documents are relevant to which cell (using Metadata derived from web documents in conjunction with Geographical Ontology) to establish co-ordinate and/or footprint information for comparison with cell dimensions.
- Generate a term index for *each cell*.
- Queries will have a geographical footprint (the query footprint) allocated to them (also derived in conjunction with the Geographical Ontology).
- Query footprints will enable direction to appropriate "areas" (or cells), whose term indexes will be searched for appropriate (textual) matches to the initial query.

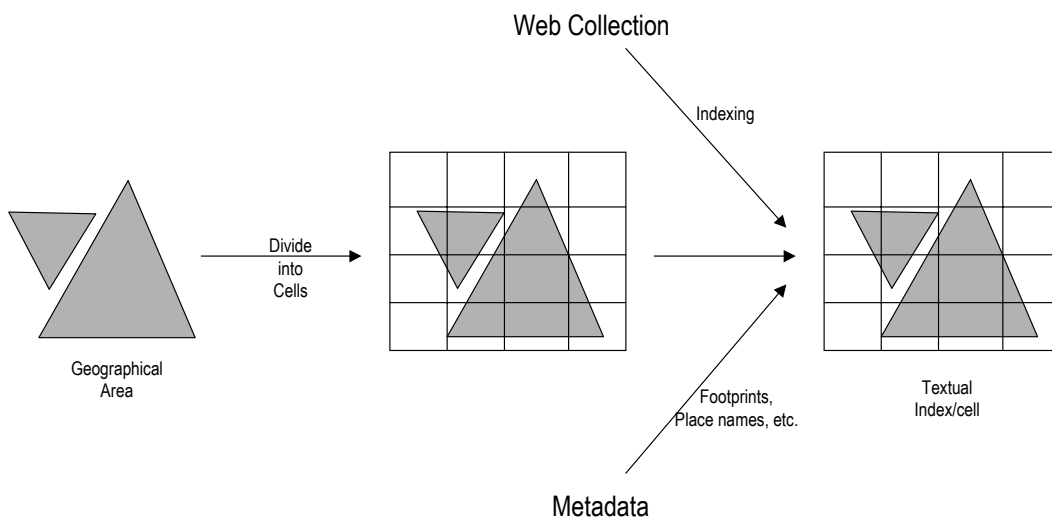


Figure 3 Possible Role of Spatial Indexing

Textual Index

The Textual Index will retain the format used in the GLASS information retrieval system, as described in the GLASS User Manual (4). It would require customization to accommodate the Spatial Indexing mechanism described in the previous section.

4.3. Run-Time Architecture

Interaction

This covers the aspects of the system that deal directly with interaction with the user, via the User Interface. It includes query disambiguation, query expansion, querying, and result presentation, as described in Figure 4.

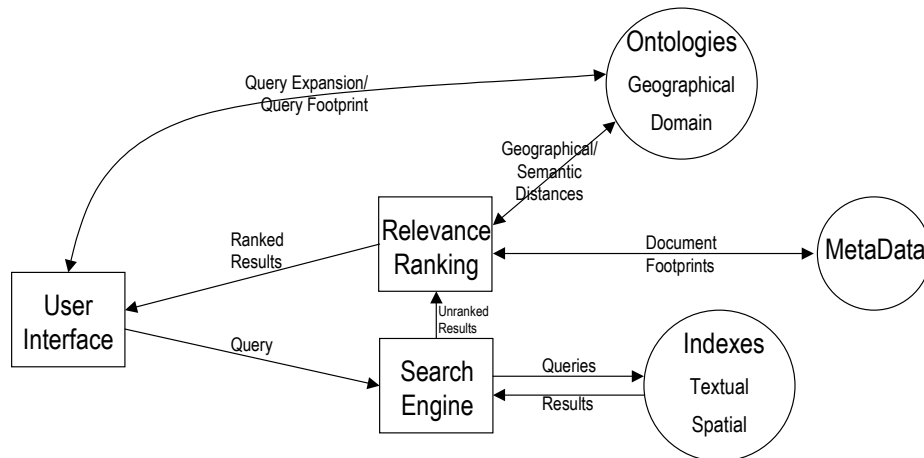


Figure 4 Query Management Architecture

Queries will be input using either the Textual or Graphical Interfaces. Queries entered via the Textual Interface will be processed in their textual form. Queries entered via the Graphical Interface will be processed into a textual form prior to query disambiguation and query expansion.

A Sketching Interface will be demonstrated, which will enable users to represent a query with sketches and text. The sketch will be processed to establish the geographical neighbourhood of the input query and a geographical footprint allocated to it in conjunction with the Geographical Ontology. A textual representation will also be generated from the sketch that, following corroboration from the user as to the appropriate interpretation, will be processed into a textual format for passage to the SPIRIT Search Engine.

Queries will be expanded to add terms to the initial query by determination of alternative and spatially related place names in conjunction with the Geographical Ontology. Depending on the effectiveness of this approach, additional interaction with the Indexes may be required to provide realistic term expansion.

Queries will be disambiguated against the Geographical Ontology before the query is sent to the SPIRIT Search Engine. The user will be prompted for appropriate interpretation.

The geographical terms employed in a query will be identified and expanded to add alternative and spatially-related place names prior to the query being sent to the Search Engine.

Search Engine

The SPiRiT Search Engine will provide the core information retrieval functionality of the system. A query input via the User Interface is directed to the Search Engine, which accesses the Textual and Spatial Indexes to find appropriate matches with documents within the Web Data Collection. It may also prove necessary to provide a link to the Metadata component to facilitate this process.

It is envisaged that the prototype acts as a demonstrator to illustrate different mechanisms of introducing spatial-awareness into a Search Engine. As such, different mechanisms of indexing the collection will be investigated, and the addition of a Spatial Index to the current Textual Index of GLASS analyzed. The manner in which this Spatial Index is introduced will also be explored:

- section 4.2 above describes Spatial Indexing acting as a filter on Textual Indexing, identifying a sub-set of spatially relevant documents to be searched for textual matches.
- alternatives to this include parallel access of the Textual and Spatial Indexes, with both sets of results passed to the Relevance Ranking component.

Relevance Ranking

This component is involved in the ranking of identified documents according to the "most relevant" to the query. The geographical relevance of identified documents may be determined by comparison of the query footprint to the geographical footprint of the document (as retrieved from the Metadata repository), as well as measures based on qualitative spatial relationships. Information retrieval techniques exist to determine the textual relevance of a retrieved document to the terms in the initial query. A scoring system will be developed to determine the relevance of both the textual and spatial components of a query to identified documents.

Query Refinement

This activity is the processing of feedback from the user into a revised query on the basis of user selections from a set of ranked results. From this set, the user may select which results they deem to best answer their query or that constitute an interesting next line of investigation.

4.4. Overall Architecture

The overall architecture shows the connectivity between all functional components in the prototype system, including pre-processing and run-time processes, see Figure 5. Aspects of this overall architecture will be focussed on throughout the remainder of this chapter, to add more detail to individual processes.

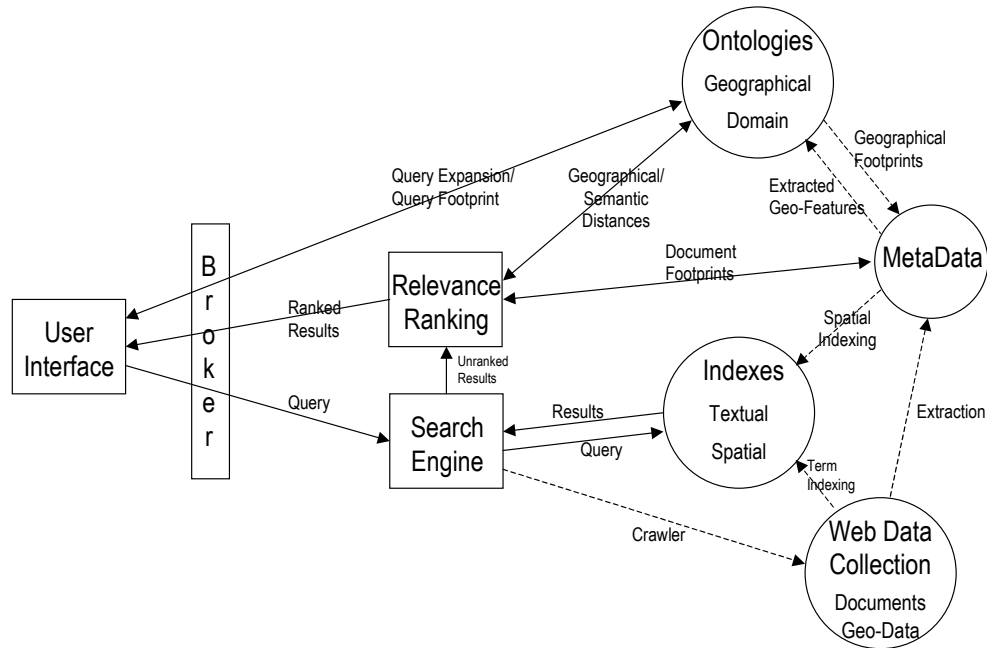


Figure 5 Overall Architecture

The order of the information flow through the run-time architecture shown above is illustrated in a UML Sequence Diagram, see Figure 6. As mentioned previously, other inter-connections may be necessary between components, but are not shown in this diagram or Figure 5 (e.g. between the Ontology and Indexes components to facilitate query expansion, and between the Search Engine and Metadata components to facilitate query processing).

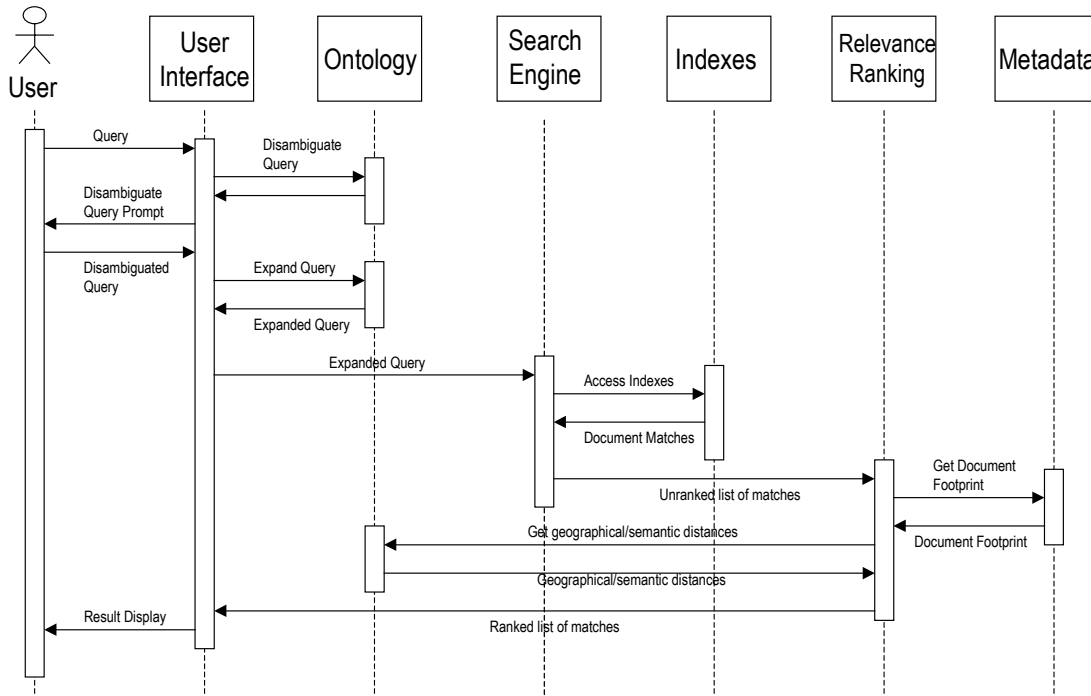


Figure 6 Sequence Diagram of Run-Time information Flow

Broker

The Broker (or session manager) acts as a messaging provider within the SPiRiT prototype architecture, through which run-time interactions pass. It receives events from the run-time components of the prototype system and sends them on to the appropriate run-time component. . Where a component only communicates with one other component (e.g. only the Search Engine calls the Indexes at run-time), there is no value in communicating through the Broker and direct communication will be used.

The Broker can be seen as acting as a buffer between the run-time components of the system, see Figure 7. As such it is able to record all actions and responses from components, generating a central session document, as well as logs of individual user sessions.

The Broker is not intended to replace interface definitions at the component level, but is seen as a messaging provider, receiving SOAP messages and transmitting them on to the relevant component in the system.

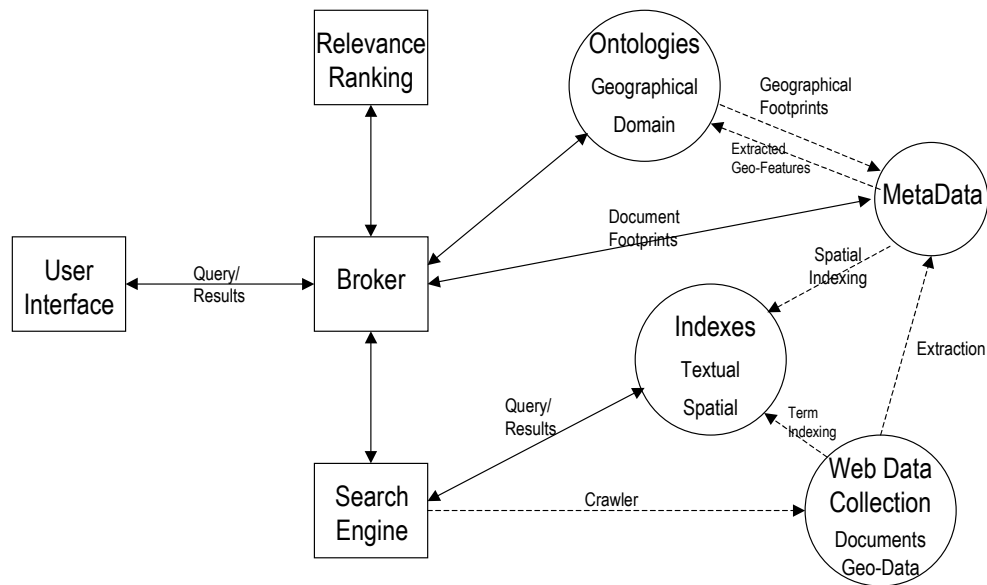


Figure 7 Overall Architecture (including Broker)

The SPIRIT prototype will act as a demonstrator to illustrate different aspects of the project. It will allow comparison and evaluation of the different mechanisms that are being investigated to introduce spatial-awareness into a Search Engine (e.g. expansion of geographical terms in the initial query, spatial indexing, spatial relevance ranking, etc.). The prototype will allow aspects of the architecture to be switched on or off, or alternative mechanisms employed for particular queries (see section 2.3). To facilitate this, a query plan format is proposed (18), which will allow details of the manner in which the query is to be processed recorded for passage through the architecture at run-time (i.e. which Control Features were selected by the user). This query plan will also record which particular component mechanisms have been utilized in processing a query, as these may affect the manner in which the query is processed by subsequent components. For example, the Relevance Ranking mechanism employed could be dependent on whether the set of documents to rank was identified with a Spatial Index filter on Textual Index access or by parallel Spatial and Textual Indexes. This format can be readily incorporated into the Broker design, as it occupies a central part of the architecture and all messages will pass through it allowing update of the query plan accordingly. The query plan format may also have an application in any identification and storage of spatial and non-spatial components of the initial query for separate processing at run-time.

4.5. Alternative Configurations

Google Web APIs

The SPiRiT prototype architecture will be provided with an interchangeable link to the Google Web APIs (11) to explore text-based search engine functionality. Such a link enables the Google Search Engine to be effectively plugged into the SPiRiT architecture. In such a configuration, the SPiRiT querying, relevance ranking and result presentation systems could be retained, but the SPiRiT web collection would be replaced, providing access to a more dynamic and much larger the data source.

The features of the proposed SPiRiT architecture which would be retained include the query interface, query expansion, relevance ranking and result presentation (see Figure 8). In the case of relevance ranking, due to the nature of the Google Web API (see below), the set of results to rank at any one time would never exceed 10. As this number is relatively small, it will allow SPiRiT to generate the metadata necessary for its relevance ranking to function on the fly. That is, since no SPiRiT metadata would have been generated for the documents referred to by the Google results, the footprints for these documents would have to be generated by SPiRiT once the set of results had been returned by Google. This information is necessary for the relevance ranking procedures to find the best geographical match to the initial query footprint.

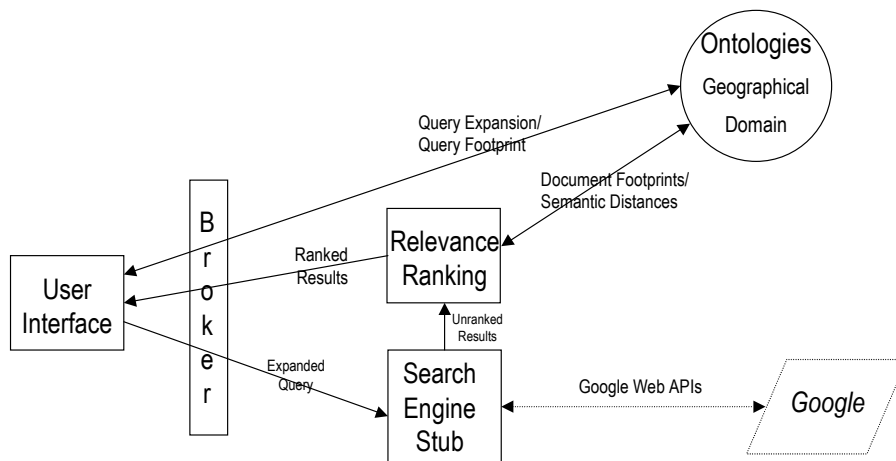


Figure 8 Alternative Configuration (including Google Web APIs)

The Google Web APIs enable developers and researchers to access and use Google as a resource in their applications. Developers write software programs that connect remotely to the Google Web APIs service, using SOAP (Simple Object Access Protocol). The Google Web APIs service, thus, gives users query access to Google's web search and dynamic web collection (more than 3 billion web pages, which are constantly refreshed). Developers can issue search requests to Google's index and receive results as structured data, access information in the Google cache, and check the spelling of words.

The Google Web APIs are classified as a "beta" service (i.e. it is an experimental service that Google does not guarantee availability for), and is limited to 1000 queries per registered user per day. A maximum of 10 results are returned per query at a time (although this limitation can readily be screened by a calling program) and up to the 1000th result for any query retrieved.

Google provide a developer's kit for using the Google Web APIs, which includes a WSDL file describing the Google Web APIs service, a custom Java client library, documentation on how to use the service with Microsoft .NET or Perl, and example SOAP messages. The service should work with any language with web services support, but has been tested with the following programming languages (and SOAP clients): Java (Apache SOAP and Apache Axis), Perl (SOAP::Lite version 0.52), Ruby (SOAP4R), and C# in MS Visual Studio .NET.

5. Technical Issues

5.1. Development Phases

Prototype Phases

The different scheduled development phases of the prototype (including 2 month extension) are:

Interim Search Engine Integration

Due: month 23

Comprising:

- Basic web search
- Interface
- Query expansion
- Ranking procedures
- Comparison of different indexing methods

Preliminary prototype

Due: month 32

Comprising:

- State of progress at month 32

Web software platform

Due: month 35

Comprising:

- Implementation of software to support client-server interactions of web prototype

Full Prototype

Due: month 38

Comprising:

- Live web prototype of spatially-aware search engine

Extensible Architecture

The architecture for the SPiRiT prototype described in this document is intended to accommodate all proposed development and research phases, of which the following have been identified:

Ontology

- Initial implementation
- Representation of imprecise regions and approximation of formal regions

Indexing

- Basic search system
- Parallelized web data collection
- Spatial indexing

Relevance ranking:

- Geographical similarity measures
- Simple multi-attribute relevance ranking
- Abstract multi-attribute relevance ranking

User Interface:

- Textual Interface
- Graphical Interface
- Sketching (to be demonstration with final prototype)

5.2. Implementation Issues

Distributed Architecture

An evaluation of the possible architectures and technologies for the SPIRIT prototype recommended a distributed architecture (8). In this architecture, all the functional components of the system can reside independently and remotely from the others, communicating via defined interfaces. This was considered suitable since it was deemed more compatible with the SPIRIT development framework, i.e. partners at remote sites working on WorkPackage components, and reflected the anticipated SPIRIT architecture, i.e. a number of interconnected and dependent functional components.

The alternative was "flat" architecture whereby all system components reside in one large module (with possible sub-modules), which can be run on one machine with all calls made locally. This approach was deemed to produce a "monolithic" code structure, which would prove cumbersome to operate and maintain, and which would require dedicated resources to manage.

There are advantages and disadvantages to both approaches, which are detailed in the evaluation document (8).

When individual WorkPackage components are first integrated into the initial prototype (see section 5.1 above), it is likely that the components will reside on partners sites, communicating remotely across the Internet. It is understood that there may be an issue of latency of messaging as a result, i.e. messages having a period of delay associated with their passage between components. With all components residing at separate locations with our distributed architecture, this may cause performance problems. A simple solution would be to limit the problem of latency by physically siting components on the same or proximate servers. The prototype will still retain its distributed nature, with components communicating with SOAP through defined interfaces for ease of continued development and maintenance.

Development Languages and Platforms

Interoperability of components will be achieved through a common communication protocol. Theoretically this means that individual functional components can be implemented in any language and on any platform, as long as the SOAP toolkits selected are fully interoperable. In practice, limits should be placed on the variety of development languages and platforms, principally in the interests of coherence and performance:

- The fate of this project is for either it (or parts of it) to be ported to a commercial partner or it to become open-source. In both cases, it is essential that a coherent and consistent software structure be produced, and this will be enabled by placing restrictions on the number of development languages and platforms.
- Performance could be affected by having too many interchanges between different languages and platforms. The selection of a distributed architecture entails a number of interfaces between components. Introduction of further interfaces between sub-

components that have been developed on different languages or platforms will affect performance and add unnecessarily to the complexity of the overall system.

The selection of practicable development languages and platforms is restricted by the fact that the SPIRIT prototype is to be built around an existing system, the GLASS information retrieval system (4). This system consists of C programs, Shell scripts and Perl programs, and runs on LINUX and UNIX platforms. As this component is central to the SPIRIT prototype, these restrictions affect implementation decisions for the other functional components.

The SPIRIT Technical Group are collating software requirements of individual partners separately in order for suitable recommendations to be made, i.e. same versions of compilers and platforms for core development.

Communication Protocol

SOAP (Simple Object Access Protocol) has been recommended to be the communication protocol of the distributed SPIRIT prototype (8). Essentially, SOAP is an open protocol specification that defines a uniform way of performing Remote Procedure Calls (RPC's) using HTTP as the underlying communication protocol and XML as the data serialization format.

SOAP Toolkits

A number of SOAP toolkits are available for use with a number of development languages and for a variety of platforms (9). The choice of SOAP toolkit is dependent on the choice of development languages and platforms for the SPIRIT prototype (see *Development Languages and Platforms* section above).

A SOAP interface has been written for the GLASS information retrieval system (4) using SOAP::Lite for PERL. Use will be made of this interface for evaluating other candidate SOAP toolkits by the SPIRIT Technical Group prior to any direct implementation and integration. This will serve the purpose of de-risking the technology and enabling recommendations to be made about suitability and interoperability of the candidate toolkits

Geometric Operations

Several of the SPIRIT functional components will be involved in geometric operations in the generation, comparison and manipulation of geographical footprints, e.g.

- Geographical Ontology: storage of footprints of known areas, generation of query footprint.
- Metadata: allocation of footprint defining area of interest to web documents.
- Spatial Index: allocation of documents to different parts of index on basis of footprints stored in Metadata.
- Relevance Ranking: determination of "best fit" of results to query footprint by comparison with footprints of result documents as stored in Metadata.

It is important that the prototype has a consolidated approach to these geometric operations, so that each component does not unnecessarily implement common procedures and that such procedures (whether implemented or imported from an external library) cater for the requirements of all components.

The geometric functionality required can be provided, to a certain degree, by a geometric library (e.g. CGAL - the Computational Geometry Algorithm Library (3)), but may require the implementation of additional functions for use in SPIRIT.

Version Management

This project will require an appropriate tool for effective version management of all software code produced. Such a tool should be compatible with the selected distributed architecture, ensuring that system integrity can be maintained throughout the development cycle, i.e. changes made to one component which affect the operation of other, related components being denoted as such within the version management system. Such functionality should allow the storage and retrieval of stable, compatible versions of all constituent components for each stage of development.

CVS (5), a freeware code management system that, although not necessarily supporting distributed architectures, may prove acceptable for SPIRIT requirements. If it proves not to be

powerful enough, BitKeeper (1), a scalable, distributed configuration management system, will be employed.

DocumentationUML

UML has been selected as the medium for documenting software design (12) to ensure consistency and coherence across the individual WorkPackages of the project. A number of tools and drawing packages are available, with Together (17) and Dia (6) being evaluated in order to recommend one, so that designs are freely transferable and editable across the project.

Software Documentation

Since the ultimate aim of this prototype is to be either ported to a commercial partner or become open-source code, it is vital that all software development be documented coherently and consistently across the project.

Various toolkits are available that enable the generation of on-line documentation from software code. The doxygen toolkit (7) is being investigated to determine its usefulness and appropriateness to this activity.

An alternative is the issuing of guidelines on the nature, format and structure of comments to be included in developed code.

6. References

- 1) BitKeeper Web Site (2002). www.bitkeeper.com
- 2) Bucher, B. (2002). "Metadata and Mark-Up Languages." SPIRIT Project Report D2 6101
- 3) CGAL Web Site (2002). www.cgal.org
- 4) Clough, P., Joho, H., & Sanderson, M. (2002). "The GLASS Information Retrieval System." University of Sheffield
- 5) CVS: Concurrent Versions System Web Site (2002). www.cvshome.org
- 6) Dia Web Site (2002). www.lysator.liu.se/~alla/dia
- 7) doxygen Web Site (2002). www.doxygen.org
- 8) Finch, D.J. (2002). "Evaluation of Possible Architectures and Technologies" SPIRIT Technical Report (07/02)
- 9) Finch, D.J. (2002). "SOAP: Simple Object Access Protocol." SPIRIT Presentation, Utrecht (09/02)
- 10) Fu, G., Abdelmoty, A., & Jones, C.B. (2002). "Design of Geographical Ontology." SPIRIT Project Report D5 3101
- 11) Google Web Site (2002). "Google Web APIs": www.google.com/apis
- 12) Jones, C.B. (2002). "SPIRIT Kick-Off Meeting Minutes" (07/02)
- 13) Jones, C.B., Purves, R., Ruas, A., Sanderson, M., Sester, M., vanKreveld, M., & Weibel, R. (2002). "Spatial Information Retrieval and Geographical Ontologies: An Overview of the SPIRIT project." 'SIGIR '02: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval August 11-15, 2002, Tampere, Finland', ACM Press, pp.387 - 388
- 14) Jones, C. B., Purves, R., Ruas, A., Sanderson, M., Sester, M., vanKreveld, M., & Weibel, R. (2002). "SPIRIT Project Annex 1 - Description of Work", IST-2001-35047
- 15) NetSolut On-Line Systems Web Site (2002). www.netsolut.com
- 16) Petrelli, D., Bucher, B., Bailey, S., & Ruas, A. (2002). "User Requirement Specification." SPIRIT Project Report D3 7101
- 17) TogetherSoft Web Site (2002). www.togethersoft.com
- 18) Van Zwol, R. (2002). "UML Diagrams for the Querymanager and the Relevance Ranking module." SPIRIT Technical Report (11/02).

Appendix A: Interface Definitions

This section defines the procedure requests and transmissions from the functional components as determined from the architectural design described above. These procedure calls are currently at a high level of abstraction, which will form the basis of a separate "Interface Definition" document. This is because these definitions are likely to be more dynamic and undergo more changes than this blueprint document and should therefore reside separately.

Pre-Processing Procedure Calls

Ontologies

<i>Procedure Call</i>	<i>Data In</i>	<i>Data Out</i>
Update Ontology	- Extracted Geo-Feature (from Geo-dataset)	- Success/Failure
Get Footprint	- Geographical Term or Identifier	- Geographical Footprint

A more detailed description of planned Ontology procedures is given separately (10).

Crawler

<i>Procedure Call</i>	<i>Data In</i>	<i>Data Out</i>
Initiate Web Crawl	- Starting point - Depth limitation	- Entries in database of visited URLs

Metadata

<i>Procedure Call</i>	<i>Data In</i>	<i>Data Out</i>
Extract Textual Metadata	- Pointer to Database of URLs	- Document Footprint entries in Metadata repository
Extract Geo-Features	- Pointer to Database of URLs	- Extracted Geo-Features

Indexes

<i>Procedure Call</i>	<i>Data In</i>	<i>Data Out</i>
Generate Textual Index	- Pointer to Database of URLs	- Entries in Textual Index
Generate Spatial Index	- Pointer to Database of URLs	- Entries in Spatial Index

Run-Time Procedure Calls

User Interface

<i>Procedure Call</i>	<i>Data In</i>	<i>Data Out</i>
Textual query	- Textual input	- Basic query (for expansion)
Graphical query	- Graphical input - Textual input	- Basic query (for expansion)
Generate Query Footprint	- Expanded query	- Query Footprint
Initiate search	- Expanded query - Query footprint	- Ranked results

Ontologies

<i>Procedure Call</i>	<i>Data In</i>	<i>Data Out</i>
Check for ambiguities	- Geographical Term	- List of possible interpretations
Expand Geographical Term	- Geographical Term	- Expanded list of Geographical Terms
Generate Query Footprint	- Set of Geographical Terms	- Query Footprint

Search Engine

<i>Procedure Call</i>	<i>Data In</i>	<i>Data Out</i>
Access Spatial Index	- Query Footprint	- List of references to documents matching footprint
Access Textual Index	- Query	- List of references to documents matching non-spatial terms of query

Relevance Ranking

<i>Procedure Call</i>	<i>Data In</i>	<i>Data Out</i>
Rank Results	- Unranked results - Query Footprint - Ranking algorithm	- Ranked results

Metadata

<i>Procedure Call</i>	<i>Data In</i>	<i>Data Out</i>
Get Document Footprint	- Document reference	- Document Footprint

Administrator Procedure Calls

User Interface

<i>Procedure Call</i>	<i>Data In</i>	<i>Data Out</i>
View all logs	-	- Complete logs
View individual session log	- Session reference	- Session log

Appendix B: Functional Component Design Diagrams

Introduction

This section is intended to provide a more detailed functional breakdown of the individual components of the SPIRIT prototype system through the medium of UML Class and Activity Diagrams. Personnel from individual WorkPackages have generated these diagrams where possible to illustrate the contents of the identified functional components, and their believed interaction with other components within the system. Where these are works in progress or are incomplete, they have been omitted.

Search Engine

Activity Diagram for core Search Engine WorkPackage by Mark Sanderson and Hideo Joho (University of Sheffield).

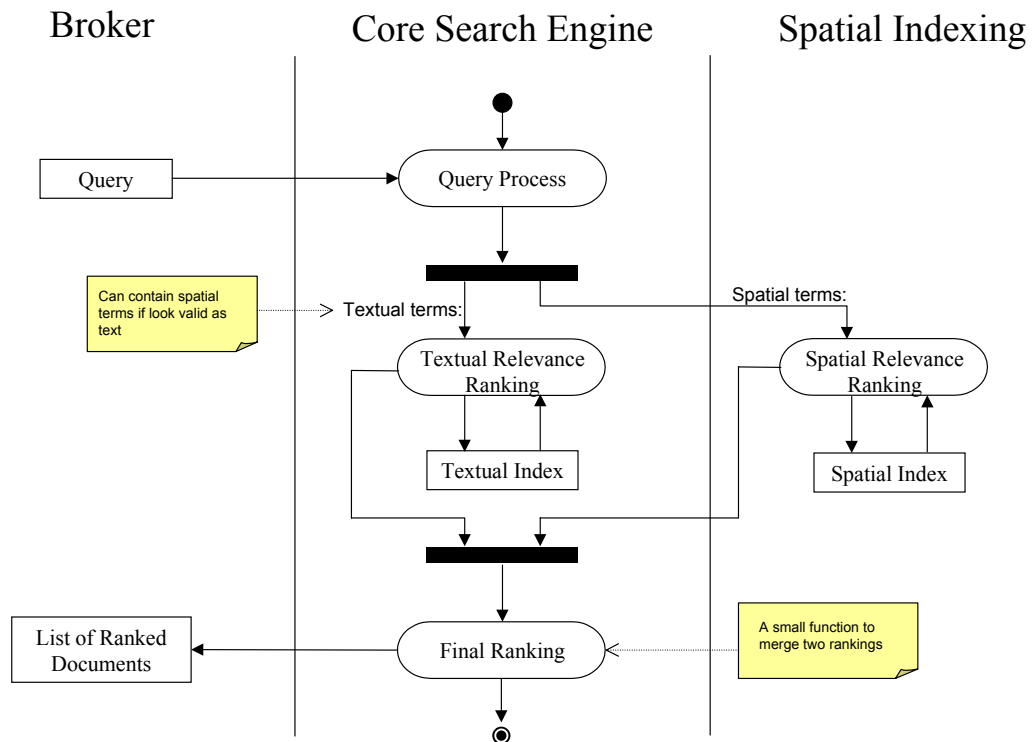


Figure 9 Search Engine Activity Diagram

Ontologies

Class Diagrams for the Ontologies component, as found in the Ontology Design document (10).

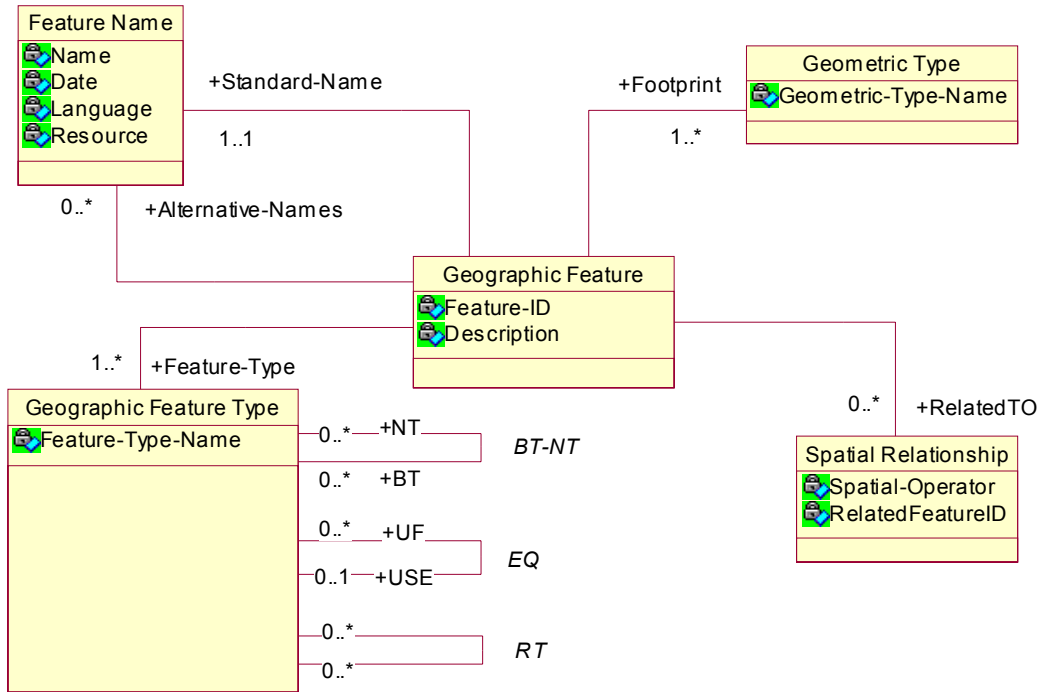


Figure 10 Base Schema of SPIRIT Ontology

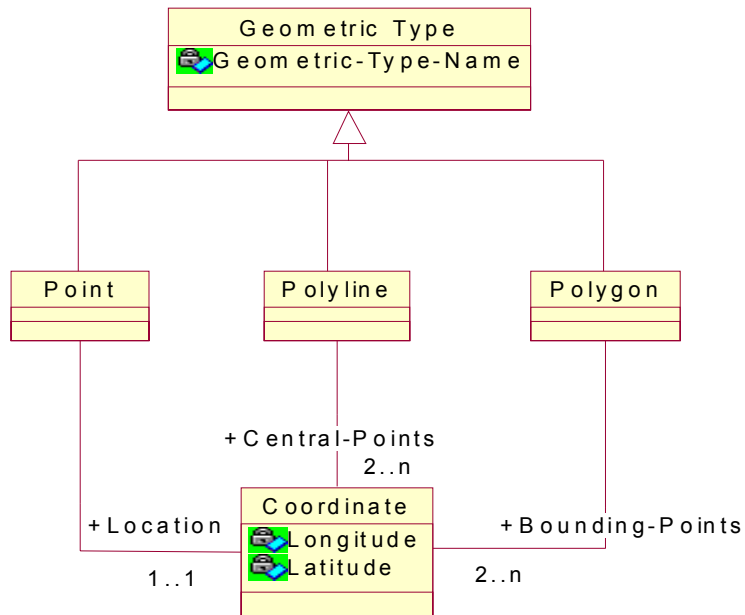


Figure 11 Geometric Feature Types

Metadata

Class and Activity Diagrams for Metadata WorkPackage by Frauke Heinzle (University of Hannover).

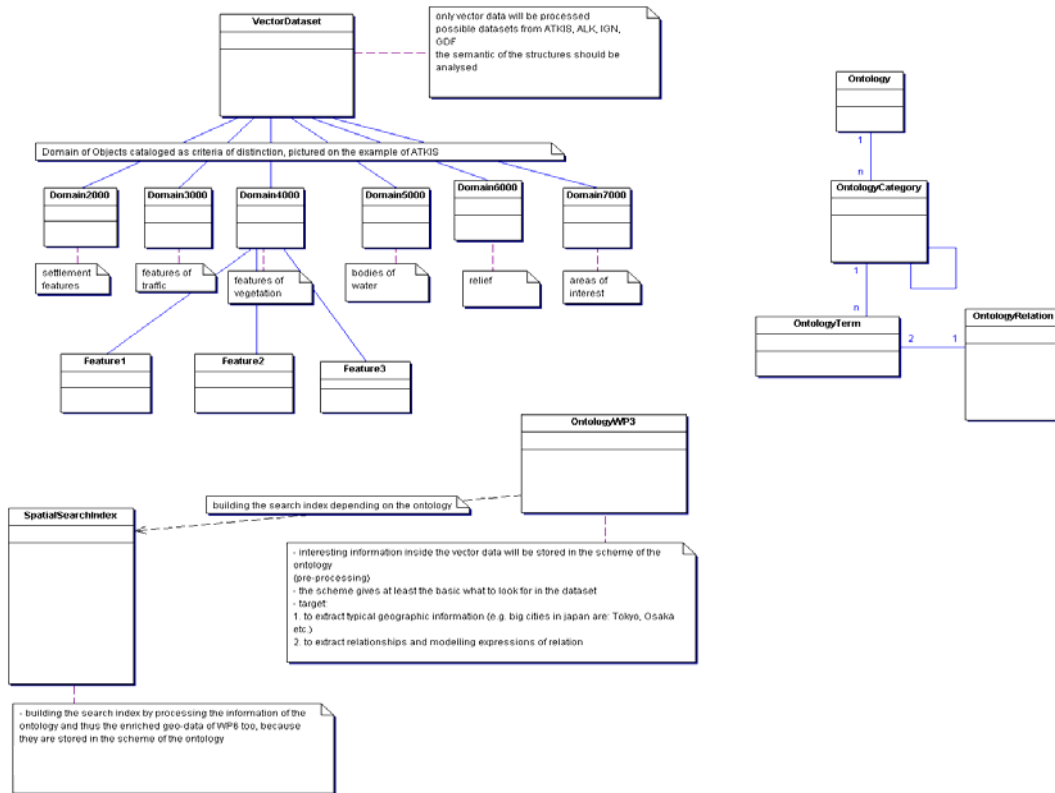


Figure 12 Metadata Class Diagram

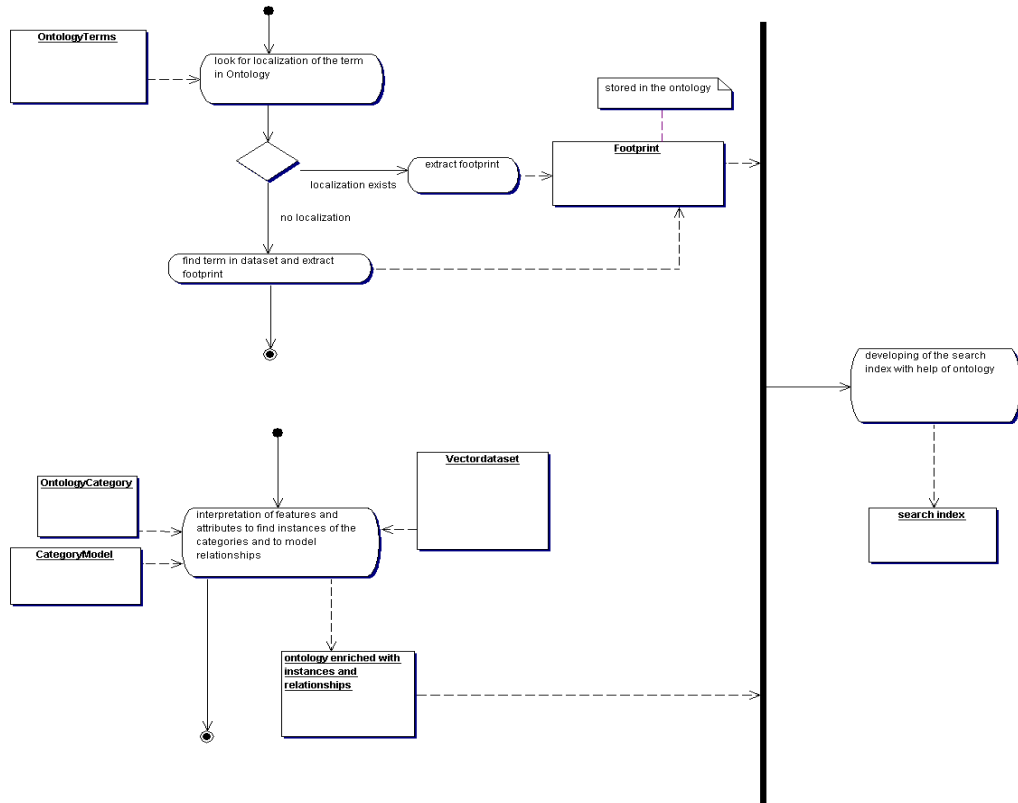


Figure 13 Metadata Activity Diagram