Project ref. no.	*IST-1999-11748*
Project acronym	**LIMBER**
Project full title	**Language Independent Metadata Browsing of European Resources**

Security (distribution level)	*Public*
Contractual date of delivery	*M07 July 2000*
Actual date of delivery	*31 July 2000*
Deliverable number	*D-2*
Deliverable name	*LIMBER System Architecture*
Type	*Report*
Status & version	*Final*
Number of pages	*34*
WP contributing to the deliverable	*WP4*
WP / Task responsible	*Intrasoft International*
Other contributors	*NSD, UKDA, CLRC*
Author(s)	Flora Malamateniou
EC Project Officer	*Mr. Kimmo Rossi*
Keywords	*System architecture, multi-lingual thesaurus, thesaurus management system, search interfaces, metadata standards, usability, inter-operability*
Abstract (for dissemination)	*This report outlines the architecture of a multilingual thesaurus based interface to existing data archive systems. The modular architecture and interface definitions in this report provide the basis for detailed module specification, design and implementation.*

TABLE OF CONTENTS

Executive Summary

This architecture report outlines the architecture of a system to provide a multilingual thesaurus based interface to existing data archive systems to meet the user requirements stated in the D1 deliverable. The modular architecture and interface definitions in this report provide the basis for detailed module specification, design and implementation.

The architecture is divided into:

- the data archive system with its own multilingual user interface client
- the multilingual thesaurus management system with a client user interface
- the metadata indexing tool which draws on the thesaurus management system to add keywords to metadata descriptions of datasets to be added to the data archive system.

Interfaces are defined between these modules, and both the modules and the interface conform to standards where they are available to maximize the breadth of the generic application of the multilingual thesaurus management system.

1. Introduction

The objectives of the LIMBER project are to create a multilingual user interface to access data and metadata from multiple sites around the world based on a multilingual thesaurus to ease search, and to promote cross-domain searching.

The tools produced by the project are intended to be generic, but the project has a specific medium term objective to support the existing NESSTAR data archive system which is currently used to access archived European social science data.

Social science data archives have main objectives to:

- identify, select, catalogue, preserve and disseminate significant social science datasets.
- make the information available to end-users.
- take into account the depositor's commercial interests with regard to provision of access to remote users.
- update the information in the archive on an ongoing and systematic basis, while maintaining date stamped previous editions.
- convert datasets in the archive to new formats as standards change.

The social science data archives holdings contain data from the public sector (statistical agencies, central government etc.), the commercial sector (opinion and market research companies) and academic research. The end-users have been identified as government departments, policy makers, private enterprise, the curious public, serious researchers, depositors, other archives and internal users.

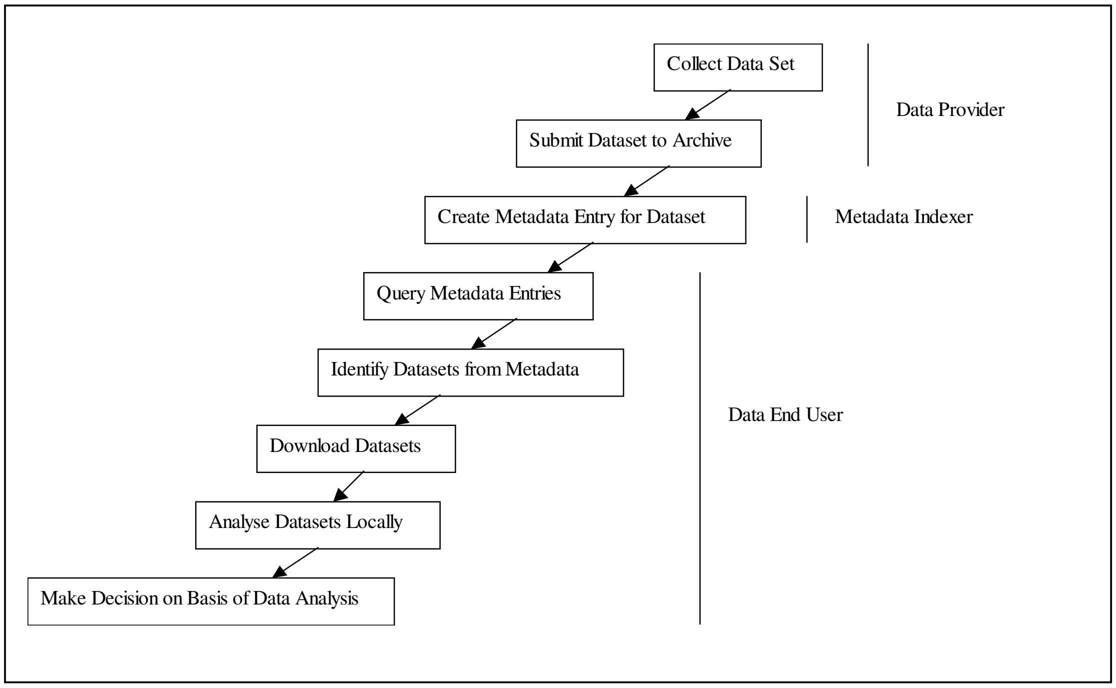
Collect Data Set
Submit Dataset to Archive
Data Provider
Create Metadata Entry for Dataset
Metadata Indexer
Query Metadata Entries
Identify Datasets from Metadata
Data End User
Download Datasets
Analyse Datasets Locally
Make Decision on Basis of Data Analysis

Figure 1. Lifecycle of Data and Metadata for Data archives

The life cycle for data and metadata in these archives is summarized in Figure 1, where LIMBER will increase the number of data end users by providing a multi-lingual interface, and de-skill the specialist role of the metadata indexer by providing support tools.

The user requirements document D1 defines the actors involved in the lifecycle of data and metadata, and provides use cases for them, as well as the thesaurus administrator, which are summarized at a very high level in Figure 1 and Figure 2. The requirements also provide detailed requirements from each of these users on the multilinguality of the interface of a data archive system. Multilinguality in a data archive system means that users should be able to:

- interact with the system in their chosen language;
- to query multilingual data resources;
- to receive the answers to queries in their chosen language where possible.

To meet these requirements, the multilingual data access system should:

- present its interface in multiple languages;
- provide a multilingual thesaurus to support queries in the chosen language.

In addition, there are two further requirements on the LIMBER project from the users of the existing distributed data archive system to:

- Provide an improved metadata format for the description of data sets
- To provide a tool to aid the indexing of data sets according to that metadata format, using the multilingual thesaurus.

The first of these is being met by LIMBER workpackage on metadata; the second should be included in the LIMBER architecture.

Following these requirements, it is necessary that the LIMBER architecture contains the following major components:

- Distributed Data Archive system with its metadata component
- Multilingual Thesaurus Management system with its appropriate thesaurus(i)
- Indexing Tool

Figure 2 shows how these components should be connected at the highest level.

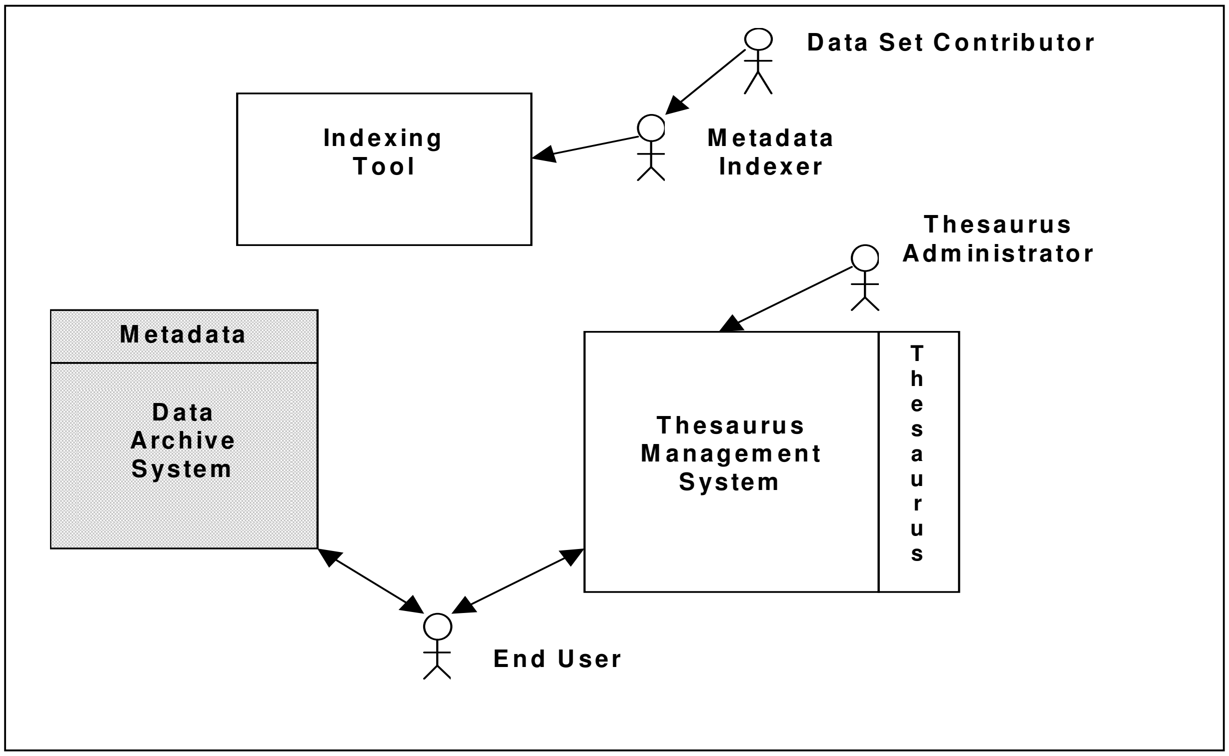

Figure 2. Overall Architecture of the Multilingual Data Archive System LIMBER

The remainder of this document will define each of these components in greater detail, along with their corresponding interfaces. Although it is not to be developed in the LIMBER project, we will present the architecture of a distributed data archive system with special emphasis to Nesstar, since Nesstar is going to be used as the 'base architecture' in Limber.

2. Distributed Data Archive System

To achieve the objective of providing multilingual access to Data Archive Systems, LIMBER must provide an interface to such systems. This section describes the generic architecture of such systems, and the specific architecture of the NESSTAR system to which the thesaurus management system of LIMBER must provide interfaces.

2.1 Distributed Data Archive System Architecture

In a distributed data archive system, data (and corresponding metadata) is stored and maintained at separate holdings across a network. The need to be met by such a system can be seen from two perspectives: that of the end users and of the data providers (local data archive administrators). End users need to locate, access and use the data in the local archives, and data providers need to make their data available to the network. Hence, the ultimate goal of an integrated data archive system is to achieve a uniform access to those separate archives for the end users and also to provide each data archive administrator with a tool to publish its data.

Figure 3 shows a generic architecture for a distributed data archive system. As it can be seen, a local adaptor is attached to each local archive. The local adaptor offers a set of services that make the data kept in the system available to the users. There is a directory service that contains information about the archives that are available on the network (their IP's) and the services they provide. In this way, if a data archive wants to make itself present in the network it first has to inform the directory service in order to be included. Any user action to the data archives should first contact the directory service to locate the available archives. The messaging system is used for the communication between user interfaces and data archives

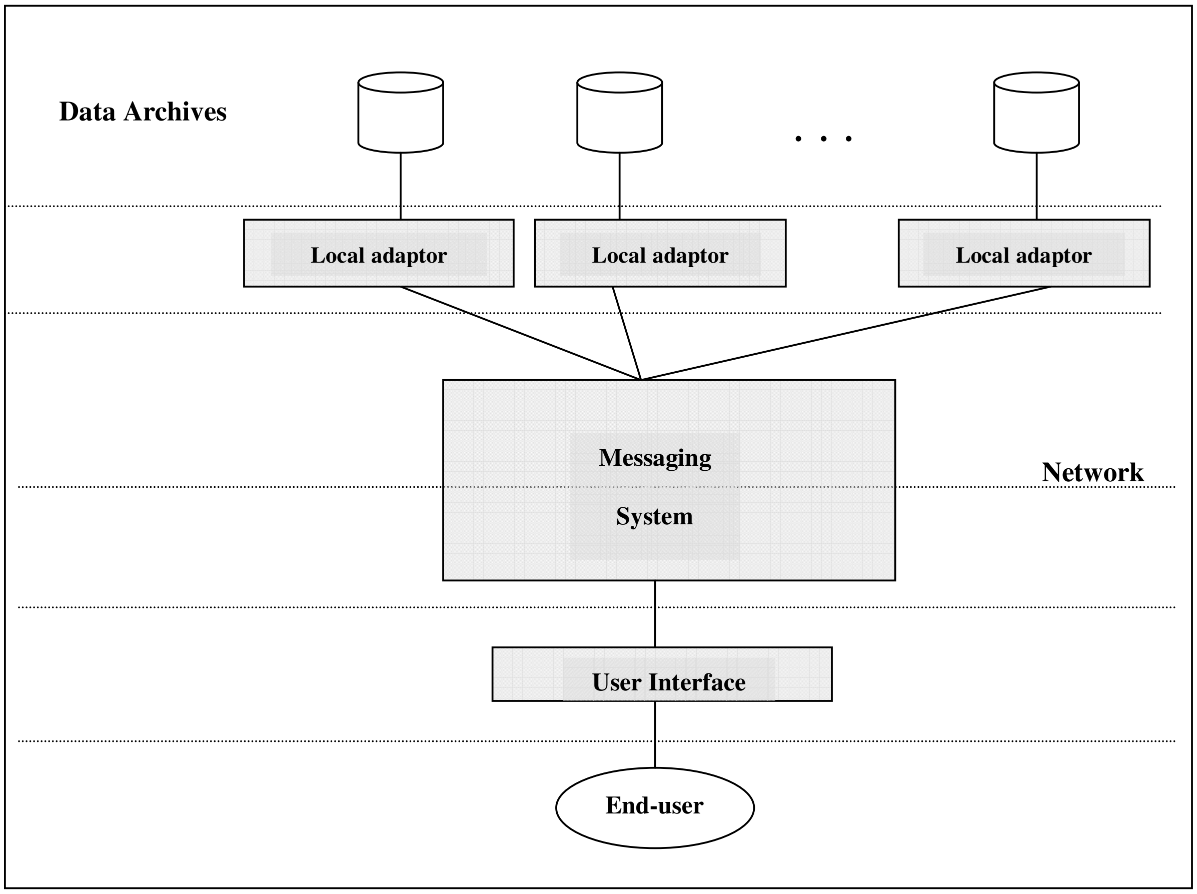

Figure 3. A generic architecture for a distributed data archive system

2.2 NESSTAR Architecture

NESSTAR is an existing and successful implementation of a distributed data archive system. NESSTAR is a distributed data archive system where data is stored and maintained at local holdings across Europe. The present system is available from http://www.NESSTAR.org and provides access to social science data in the UK, Norway and Denmark, as well as several other European countries. NESSTAR provides a test bed for the multilingual developments in LIMBER.

In parallel with the LIMBER project, another EU IST funded project FASTER is advancing NESSTAR's capabilities to store and access datasets at different granularities to the current provision, and to improve the access to the data when downloaded for local analysis by end users. It is important that LIMBER requires minimal changes to NESSTAR to support the interface to the LIMBER's multilingual thesaurus, and that these changes are undertaken by members of the NESSTAR consortium who are also members of the LIMBER project (NSD and UKDA/University of Essex).

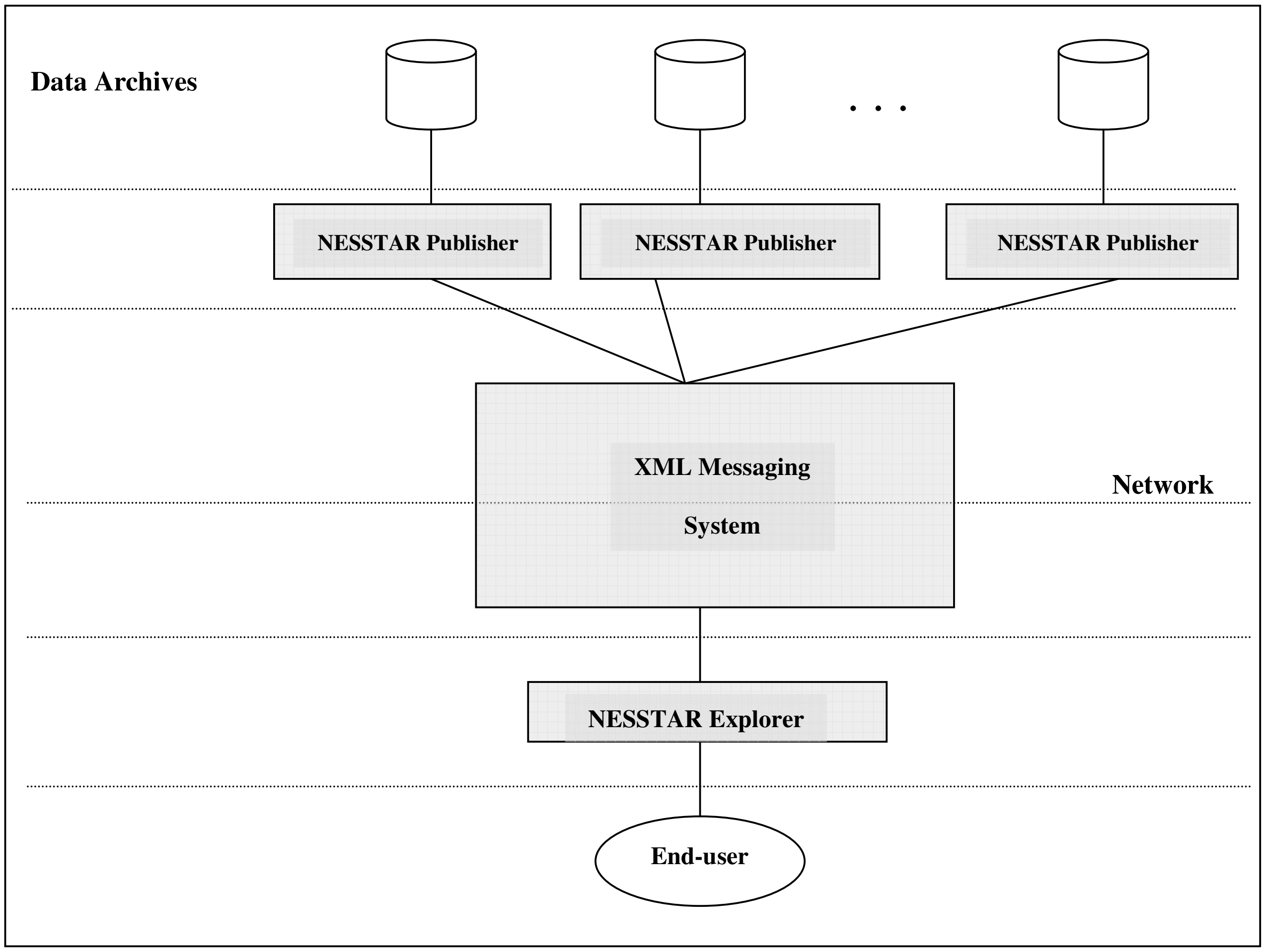

Figure 4. NESSTAR's Distributed Architecture

As it can be seen in Figure 4, each data archive is associated with a publisher. The publisher provides a set of services that provide access to the data and also serves as a means for data providers to integrate their data into the distributed system.

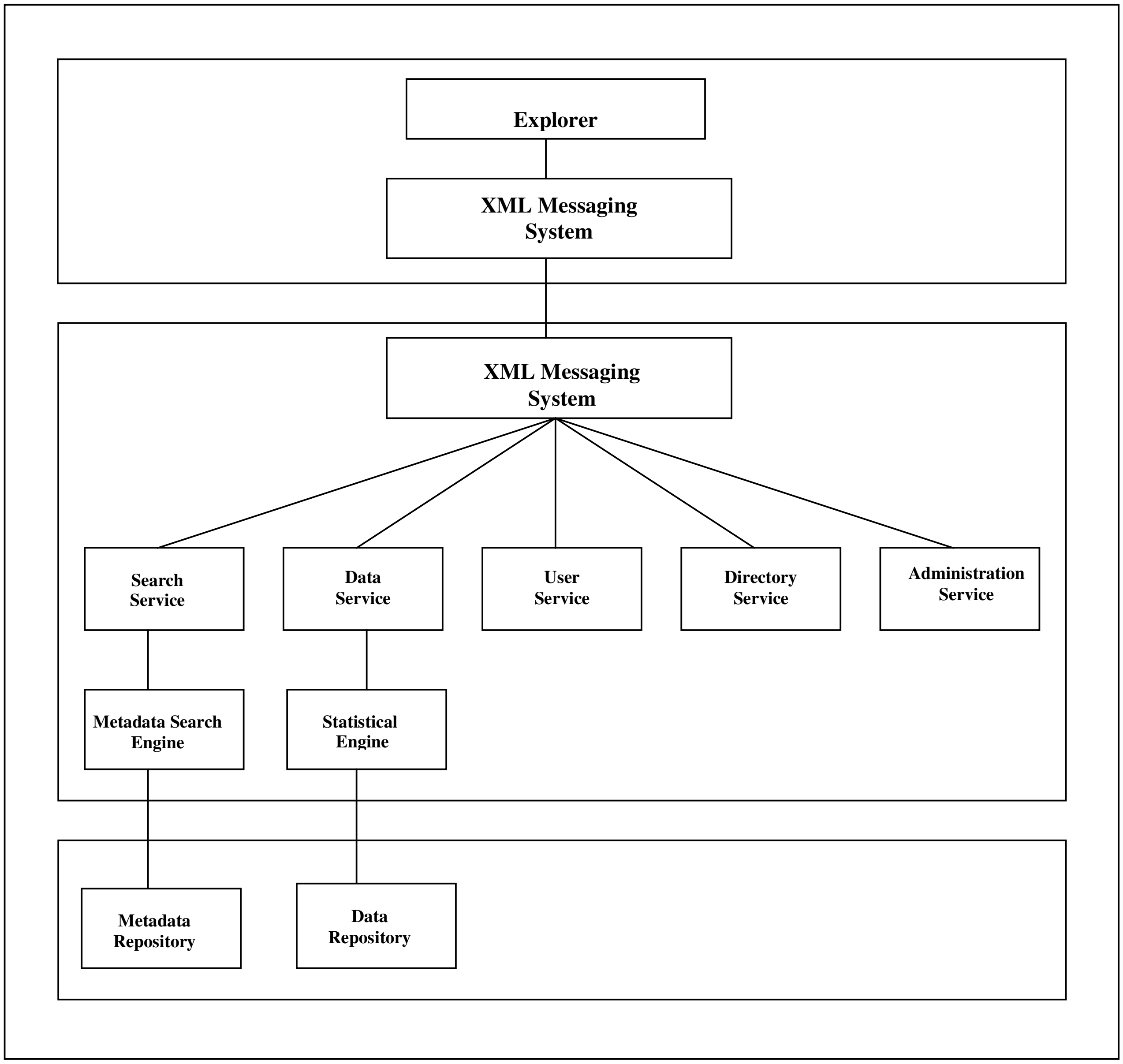

Figure 5. NESSTAR's layered distributed architecture

NESSTAR is composed of three basic elements:

- *NESSTAR Explorer*: the end user client that provides a graphical user interface to the system.
- *NESSTAR Publisher*: The local adaptors that allow data providers to make their data holdings available on the web by offering a set of services to the users.

- *The NESSTAR Messaging System*: An XML over HTTP protocol for the communication between the explorers and the publishers.

NESSTAR's network is described by a Directory Service. The Directory Service maintains the location of the data archives (their IP's) and the services provided by the publishers. Upon start up an Explorer will always call up the Directory Service to download and display the map of available publishers. NESSTAR publishers offer an access control service, an administrative service and a search service.

As shown in Figure 5, NESSTAR is implemented as a layered distributed architecture. A first layer is composed of a user client (the explorer) that provides a graphical user interface to the system. A second layer is composed of the publisher and the services they provide and a third layer is composed of the data and metadata repositories. The messaging system is working as a broker between the explorers and the publishers. The messaging system receives requests from the explorers and it is forwarding them to the appropriate services. It also collects the results and returns the final result to the explorers.

2.2.1 Definition of architectural components

NESSTAR Explorer

The Explorer contains:

- An integrated metadata/data browser used to display metadata and results from on-line data analysis.
- A search window that is used to specify query criteria.
- A metadata editor for metadata publication.
- An administrator window used for server and client management.

NESSTAR Publisher

NESSTAR publishers offer a set of the following services:

The Directory Service

The Directory Service contains the descriptions of the available data and of the organizations that provide it. The Explorer uses this information to present the user with a browseable map of the available data and services.

The Access Control System (User Service)

In order to protect the data and the servers from unauthorized access, the publisher includes an access control system. This system prevents unauthorized use of data and prompts the users for necessary action to be taken in order to access a particular source. The access control system consists of a decision-making access manager, an archive specific description of access policies and rules and a user database holding the information about the users and their rights.

The Administration Service

The administration service is used to remotely manage the publishers (server management) and to keep the explorers up to date (client management).

The Search Service

The search service routes the queries to the metadata search engine.

The Metadata Search Engine

Metadata access is performed by the metadata search engine (Cheshire in NESSTAR).

The Statistical Engine

The statistical processing engine is used for online statistical data processing.

The NESSTAR Messaging System

The information is exchanged among the elements of the system by using XML messages that are implemented on top of HTTP.

2.2.2 NESSTAR Functionality

The main functions provided by NESSTAR are the following:

- Integrated search facilities
- Online statistical processing
- Access control
- Metadata and data editing and publishing
- Remote administration

NESSTAR Explorer

The Explorer comprises the following functional components:

Search window

- Free-text search across the entire load of metadata
- Structured search across a selection of the most frequently used fields
- Advanced Boolean search on all relevant fields of the DDI-DTD

Results browsing

Search results displayed in a hit list, which in addition to the first lines of the abstract can display as much as fifteen fields depending on the user preferences

Metadata browsing

Metadata in NESSTAR are not only used for resource discovery. As soon as a dataset is located the accompanying XML-formatted metadata travel the net and can be viewed in a combined metadata/data browser. In this tool the user can easily jump from full text descriptions of variables to statistical analysis.

On-line data analysis

The on-line data browser in NESSTAR includes basic statistical methods. For every statistical method, relevant graphical visualization methods are available.

Data Export

When the user decides to continue the analysis locally, the dataset can be sub-setted by variable and cases to create manageable selections. The user can also choose between a variety of output formats. In addition an HTML-version of the complete DDI-metadata files can be downloaded together with the datafile.

NESSTAR Publisher

The publisher provides support for NESSTAR's services: user management, resource discovery, query handling, search results management and data/metadata publication.

3. Multilinguality in a Distributed Data Archive System

The view taken here is that LIMBER is not an entire system itself, but a set of components to be integrated with a pre-existing resource discovery system, such as NESSTAR. Throughout this document, NESSTAR is used as the "base architecture", of a distributed data management system so that all examples of use are based on an anticipated NESSTAR/LIMBER integration. It should always be born in mind, however, that NESSTAR should not be the sole target for LIMBER integration.

The "added value" that LIMBER will bring to NESSTAR and similar systems is *multilinguality*. There are two main aspects in supporting multilinguality within a data archive system:

- Multilingual User Interface, i.e. allowing users to interact with the data archive in their chosen language;
- Multilingual Retrieval, i.e. the ability to query multilanguage data resources.

Regarding the first aspect, NESSTAR already has the capability to present its interface in multiple languages, although these are not instantiated for languages other than English and Norwegian. Therefore, there is a need that the data files of terminology used at the interface and the on-line help files for the NESSTAR system to be made available in each of the required languages not currently supported: French, German, Spanish.

The multilingual user interface should also allow the Nesstar end-user and for that any data archive system user to perform the following:

- switch between a free-text and a keyword search
- have a free-text search option that employs translation and synonyms
- perform keyword search that employs the assigned concept classification code
- use stemming, truncation or fuzzy searching that is employed on words from a failed search string to list the terms from a controlled vocabulary which contains those words or derivations.
- search across all sites and in all languages
- save all searches in a search history with the option to combine, using Boolean operators, with other saved searches or new searches.
- drill down into any hierarchy of multilingual terms starting from its top term
- view the complexity of multilingual terms relationships when requested or in a browsing interface. Otherwise to see a simple alphabetic listing of the relationships to the selected multilingual term
- view whole hierarchies of multilingual terms
- view the hit list ranked in the importance he/she defined
- view the hit list accompanied by a suggestion list from a controlled vocabulary
- view the titles of the retrieved datasets in his/her language with the ability to select the language in which to display the full metadata record

With respect to second aspect, the end-or any other system-user gives a query in his/her own language (e.g. in English) and asks the system to search not only the English data archive but also the Norwegian and the French. That means that the system should have the ability to translate the user's query into the above languages. Taken Nesstar's layered architecture (Figure5), a multilingual search service can be added that will be responsible for routing the queries to the metadata search engine. This interaction requires that the search engine used in NESSTAR supports multilinguality e.g. by providing a simple interface to a multilingual thesaurus management system.

From the above analysis regarding the multilinguality in a data archive system, it is concluded that LIMBER must develop components that will satisfy the above end-users requirements. These architectural components are described in detail in the following section.

4. LIMBER Architectural Components

In order that Limber provides multilinguality to Nesstar or any other data archive system, it must contains the following architectural components:

- Multilingual Thesaurus Management System
- Thesaurus
- Indexing Tool

A Multilingual Thesaurus Management system will provide multilingual translations of terms, whenever this function is required, using the appropriate thesaurus(i).

An indexing tool will be used for the indexing of the metadata in the data archive system according to a thesaurus. The above architectural components are shown in Figure 6 and are described in greater detail in the following sections.

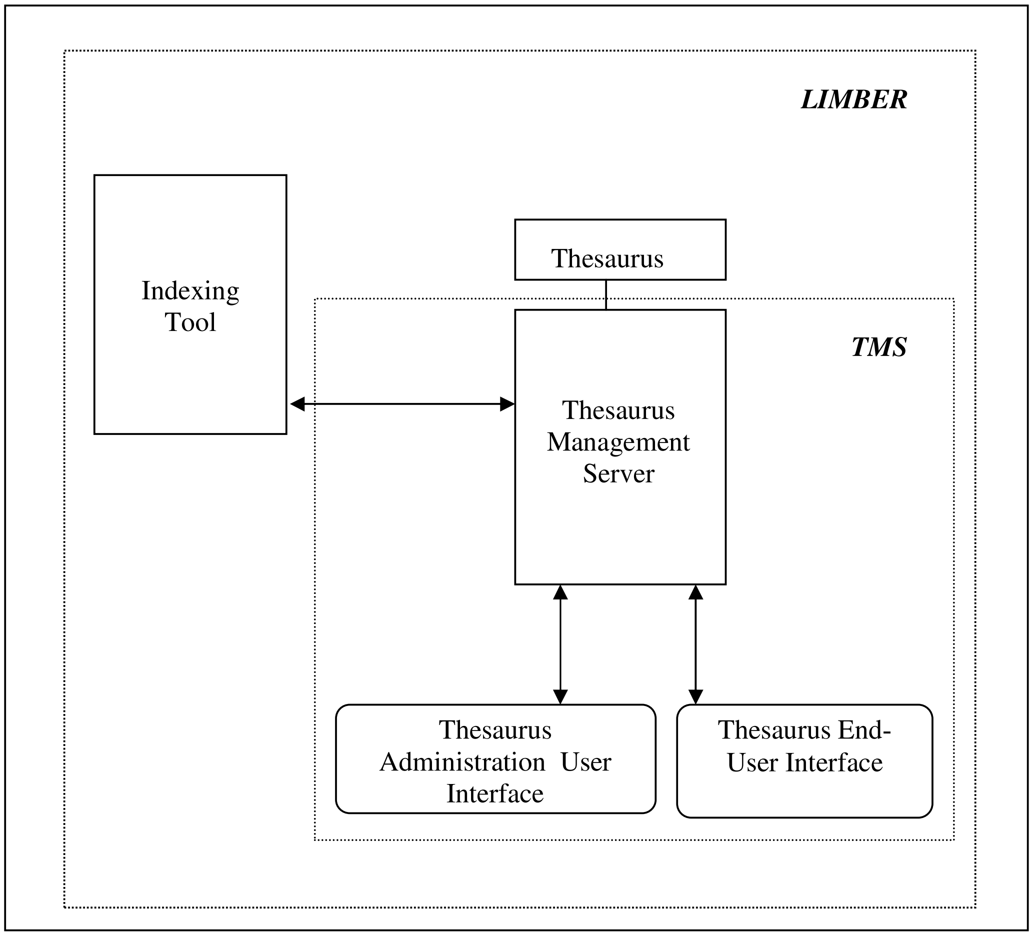
LIMBER
Indexing Tool
Thesaurus
TMS
Thesaurus Management Server
Thesaurus Administration User Interface
Thesaurus End-User Interface

Figure 6. LIMBER Architectural Components

4.1 Multilingual Thesaurus Management System

The overall architecture of the Multilingual Thesaurus Management System is shown in Figure 6 and is composed of the following:

- Thesaurus Management Server
- Thesaurus Administration User Interface
- Thesaurus End-User Interface

4.1.1 Thesaurus Management Server

Use cases are presented in the User Requirements deliverable D1. From these we see several ways in which a thesaurus could be used by end-users within LIMBER:

1. Not used. The user formulates queries without reference to any thesaurus.
2. Semi-automatic: the system applies the thesaurus to the user's search terms, and suggests translations. See below.
3. Interactive browsing. The user can interact with the thesaurus to explore translations of her terms or to manage the thesaurus. See below.
4. Back translation. The thesaurus is used to translate keywords (and possibly titles and abstracts) of retrieved documents into the user's working language.

The first case merely emphasizes the need for separability between the base system and the LIMBER thesaurus components. Existing systems such as NESSTAR (even if they are evolving) cannot be made dependent upon LIMBER.

4.1.2 Thesaurus End-User Interface

The Thesaurus End-User Interface allows the end user to browse the thesaurus selecting terms to be placed into a query.

The client should be available as a web page, or as a portable java program, which communicates with the thesaurus management system server.

The interface should support full UNICODE for character presentation in order to be able to present the full range of possible language preferences.

The menus, error messages and help information to the client will be pointers to text in data files to support multiple languages. Each data file will exist for each language. The languages of English, French, German and Spanish will exist, with the potential to add new languages.

The user will be able to select a menu item for Preferences, one of which will be Language, presenting a sub menu of the list of selectable languages.

4.1.3 Thesaurus Administration User Interface

The Thesaurus Administration User Interface allows the end user to browse the thesaurus selecting terms to be placed into a query.

The client should be available as a web page, or as a portable java program, which communicates with the thesaurus management system server.

The thesaurus administration user interface will include all the functionality of the thesaurus end-user interface with the addition of functions to:

- Update the thesaurus
- Delete items from the thesaurus
- Edit existing items in the thesaurus
- Edit existing links in the thesaurus

4.2 Thesaurus

LIMBER is reducing and translating the HASSET thesaurus of social science terminology into French, German and Spanish in addition to the existing English terms. The thesaurus construction follows the ISO standards for monolingual (ISO 2788) and multilingual (BS 6723:1985) thesaurus construction. This multilingual thesaurus will be formatted into an RDF representation for storage and use by the thesaurus management system.

4.3 Indexing Tool

Another important aspect of LIMBER is the provision of an "indexing tool". This is not considered an end-user application, but is a separate application that can be used by metadata indexers to use a thesaurus to generate keywords for fields in the metadata entry for a data set. The consequence of providing this tool is to de-skill the job of metadata indexers which is currently a bottleneck in the use of metadata, and therefore cross domain data access. Based on the current indexing procedure use case in section 1.2.1 of the User Requirements deliverable D1.

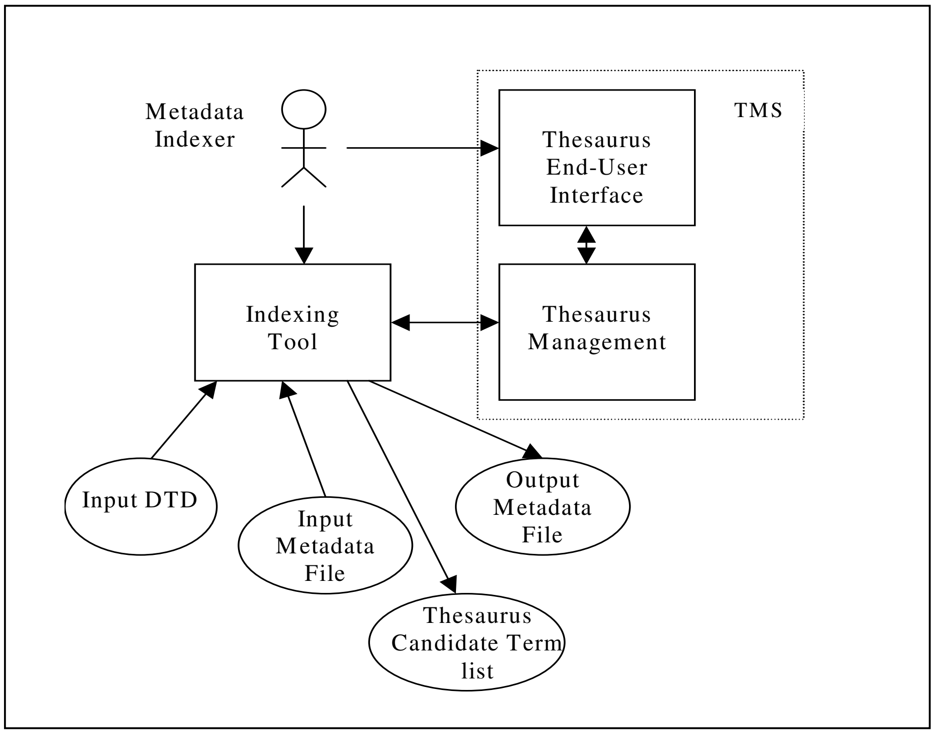

Figure 7. Architecture of the Indexing Tool

As it can be seen in Figure 7, the indexing tool takes as input and produces as output the following:

Input

1. A machine readable XML DTD of the metadata file (or XML schema definition or RDF schema definition as appropriate) – about 200 different fields that can contain text, about 20 of these can contain keywords, but some (e.g. variables) can be repeated many times in an actual metadata instance.

2. a machine-readable file containing a metadata representation in the DDI XML (or XML schema, or RDF) format for metadata. It will contain the text entries for fields in the natural language and fields to contain keywords. Such a metadata entry may contain 500 fields for which key words are required.

3. The user will also state the natural language.

Output

1. The output will be the input-2 metadata document for a dataset, with the additional inclusion of keywords for each field that is appropriate.

2. A machine readable file of a list of terms which are candidates for inclusion in a thesaurus

5. Interfacing Thesaurus Management System with Data Archive Systems

The Data Archive System (instantiated as NESSTAR for the demonstration purposes of the LIMBER project) should be able to use an interface for supporting multilinguality and thesaurus existence to communicate with the thesaurus management system. This section investigates two alternatives for a thesaurus management system to interface with the Nesstar system.

The following diagram (which owes considerable IPR debt to NESSTAR, drawing as it (literally) does on the NESSTAR Technical Overview) illustrates a tightly coupled LIMBER/NESSTAR integration.

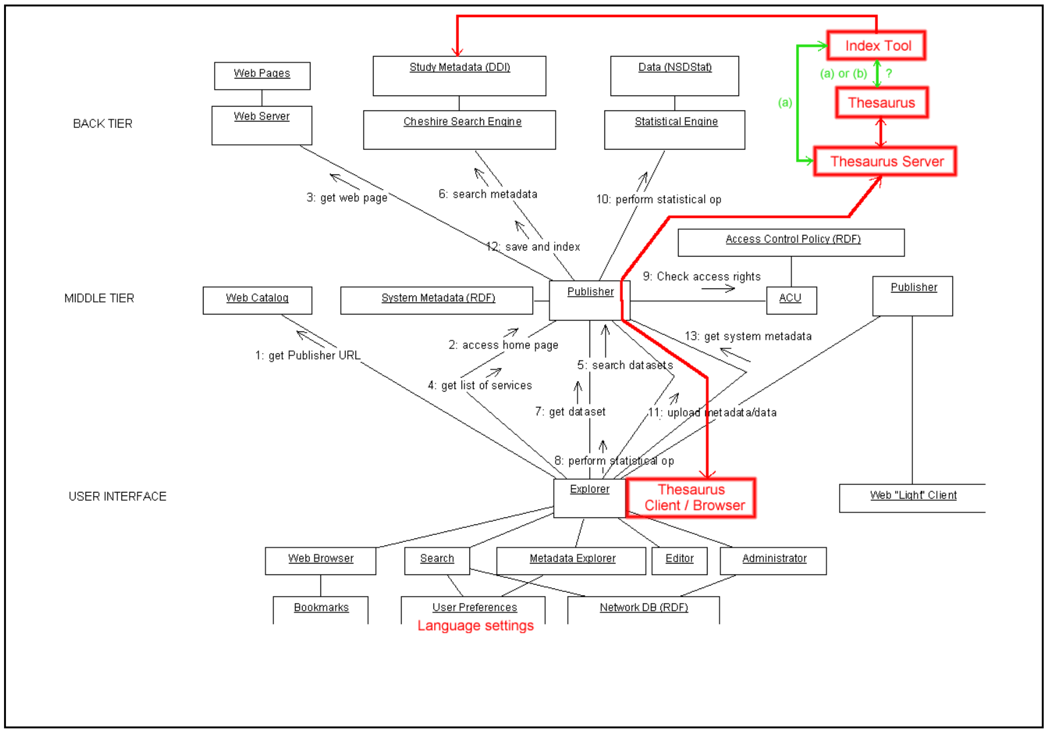

Figure 8 Tight Integration of the Thesaurus Management System and the NESSTAR Data Archive System

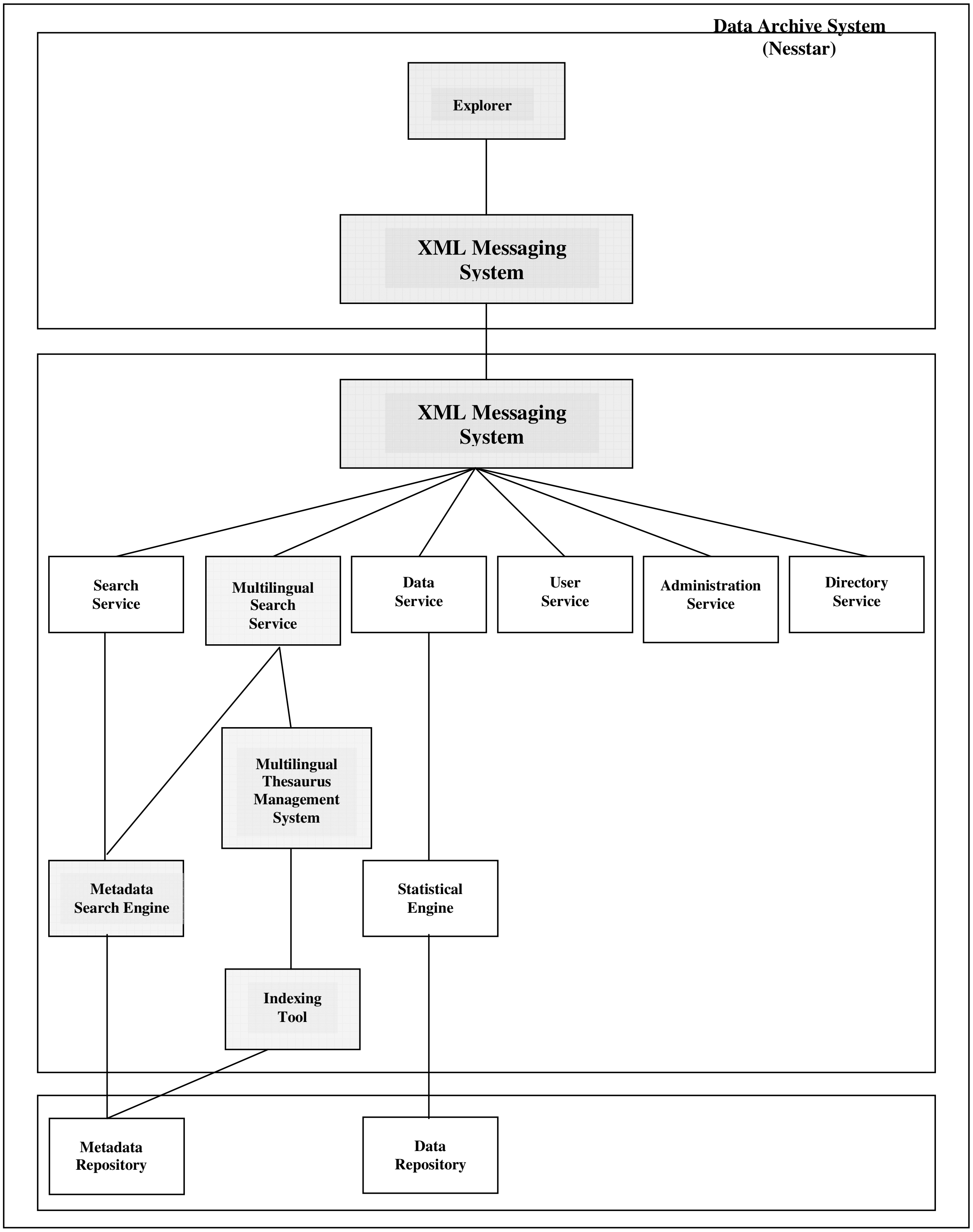

Figure 9. Tightly coupled NESSTAR/TMS integration in a layered architecture

Figure 8 shows the thesaurus browser (thesaurus end-user interface) as closely coupled with the NESSTAR Explorer. We draw attention to the need for user preferences to include language settings (e.g. preferred interaction language, source language and target language). Communication between the thesaurus browser and server is channeled via the NESSTAR Publisher, though the Publisher is only used as a communications medium. The thesaurus server front-ends one or more thesauri. The indexing tool should work via the thesaurus server to any particular thesaurus.

Figure 9 shows a tightly coupled integration of the thesaurus management system and the Nesstar Data Archive System in a layered architecture.

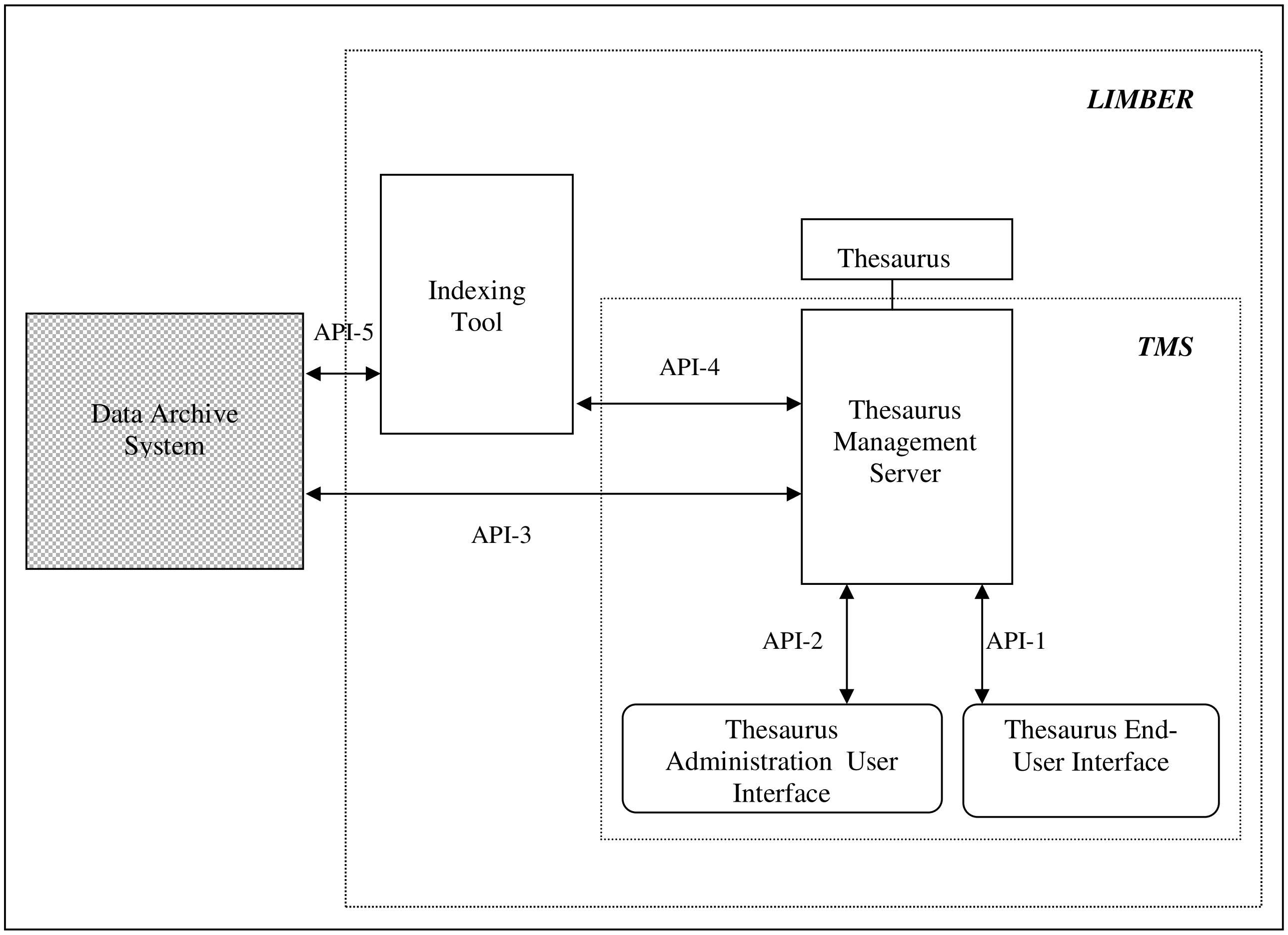
LIMBER
Indexing Tool
Thesaurus
TMS
API-5
API-4
Data Archive System
Thesaurus Management Server
API-3
API-2
API-1
Thesaurus Administration User Interface
Thesaurus End-User Interface

Figure 10. Loosely coupled integration of the Thesaurus Management System and the Nesstar Data Archive System .

Figure 10 shows a loosely coupled integration of the thesaurus management system and the Nesstar data archive system. This loosely coupled integration can be interpreted in two ways: a) the Nesstar user explicitly "switches" to the thesaurus end-user interface and using that to suggest alternatives to his/her query terms. In that case, it is of high importance that the user should be able to pass his/her existing query to this interface, and he/she should be able to pass the thesaurus-modified terms back to the Nesstar system; so at the very least a cut-and-paste mechanism should be possible between LIMBER and Nesstar system. The key aspect of this use model is that the user's contact with the thesaurus management system is minimal: a single-shot call-and-response (e.g. "give me your best French translations for this term"). Clearly, the ability to configure the interface parameters would be useful, for example the source and target languages, the maximum number of translations to return, and some valuation settings (e.g. prefer narrower to wider terms, or vice versa). From the user's viewpoint, he/she should not have to define these parameters for every interaction (though the *system* might do this, so that the "thesaurus management server" need not preserve any state for this form of interaction), b) the Nesstar user is not involved directly but whatever system component is responsible for distributing a user query to multiple data sources could call upon the thesaurus management system through an API that is provided by the thesaurus management server, in order for the later to provide multilingual translation of query terms.

Having investigated a closely coupled architecture between LIMBER and NESSTAR, the project will not develop a closely coupled thesaurus management server linked to the internals of NESSTAR since this would reduce the generic application of the thesaurus management server and would limit potential future exploitation. Instead the project wishes that the Limber components can be used not only by Nesstar but by any other data archive systems and therefore it will develop a loosely coupled system where the thesaurus management server provides an API which NESSTAR or any other Data Archive System can use to call on the multilingual thesaurus management system (see Figure 10).

6. Definition of LIMBER Component Interfaces

In this section we define the component interfaces with regard to Limber architecture defined in section 4.

Figure 10 shows the API's between the Limber components and between the Limber components and the data archive system. Hence, API-1 is defined as the API between the thesaurus end-user interface and the thesaurus management server. API-1 is the same API that is going to be used by the data archive system (API-3). API-2 is defined as the API between the thesaurus administration user interface and the thesaurus management server. API-2 includes a few additional functions to the API-1 functions. API-4 is defined as the API between the indexing tool and the thesaurus management server and it is a subset of API-1. The API between the indexing tool and the data archive system (API 5) is described in detail in section 4.3. The above API's are described in detail in the following sections.

6.1 Interface between the Thesaurus Management Server and the Thesaurus End-User Interface (API-1)

In broad terms, we expect a multilingual thesaurus to be able to:

- suggest alternatives to a user phrase that is not a term in the thesaurus (a keywords-in-context list, most probably);
- translate a term from one language into another (or several languages);
- ditto, but at the same time apply another relation (broader, narrower, preferred, etc.);
- perform root/stem analysis if required (this may not be a thesaurus function *per se*, but may be performed by a separate "stemming tool" that generates further terms to supply to the thesaurus);
- list the root terms (for hierarchical browsing).

There are a number of higher-level operations that are closer to the user view of the system, e.g. to provide a list of broader terms as suggested alternatives; but the thesaurus client can provide these by performing sequences of the above thesaurus-level operations.

We assume that a multilingual thesaurus will take the following types of input:

- individual terms or phrases;
- canonical classification code (as returned by the thesaurus, useful for hierarchical browsing);
- source and (possibly multiple) target languages
- choice of relation (broader, preferred, narrower, ...); note that some of these return multiple objects;
- whether to apply stemming analysis etc. (though this may not be a thesaurus function, it may be provided by the thesaurus server via a separate stem generator).

It will be more efficient for a client to supply the classification code in thesaurus requests whenever it is known, e.g. during browsing, or to obtain the broader terms of a known term.

We expect a multilingual thesaurus to produce outputs of the following types:
- term
- classification code
- scope note
- sets of terms and/or codes (for requests under multi-target relations)

In practice, the thesaurus might always return a compound object – a term together with its classification code, for example.

For generality, we could assume that the result is always a set, though for some relations this will never have more than one element (and for some of those, it may *always* have a single element). We assume that the client remembers what it asked for, rather than expecting the thesaurus to return a structure that incorporates the original request.

This suggests that at a very abstract level, the thesaurus API uses the following data types (along with sets and tuples thereof):

- term

- language (e.g. ISO 2-character code)
- classification code
- relation
- scope note

The RDF model that forms the basis of defining the thesaurus API follows the structure of thesaurus content as defined in the ISO standards 2788 and 5964. The thesaurus contains a set of *concepts.* Each concept is seen as a single unit of meaning in the thesaurus. Each concept has a unique *classification code.*

Each concept *is indicated by* a set of *terms* defining words and phrases which represent the concept. Each concept can have more than one term; each term is in one and only one concept. Each term has a *language* and a *value*. Languages are given by a reference to a standard language code given in ISO 639, and the value by a *literal string.* Certain terms are *preferred terms* and represent the canonical term for that concept; there should be at most one preferred term for each language.

A concept can have a set of *scope notes.* A scope note has a value, being a literal string, a language, giving the language of the notes, and a *scope note type*, three of which are defined: *Regular, Translation, Hierarchy*. Only top concepts (see below) can have hierarchy scope notes.

Concepts are arranged in *hierarchies*. At the top of the hierarchy is a *top concept.* Below the top concept, concepts are arranged in a tree structure. . Each concept has a relation *top* to the unique top concept in its hierarchy.

The parent concept in this tree is in the *broader concept relation* for that concept; a child concept is a *narrower concept relation.* If concept A is a narrower concept for concept B then B must be the broader concept of A. All broader and narrower terms must share the same top concept.

Additionally, any concept can have a *related concept* relation to any other concept. If concept A is a related concept of concept B, then B must be a related concept of A.

Figure 11 gives a diagrammatic representation of the Friend concept, with classification code 620.

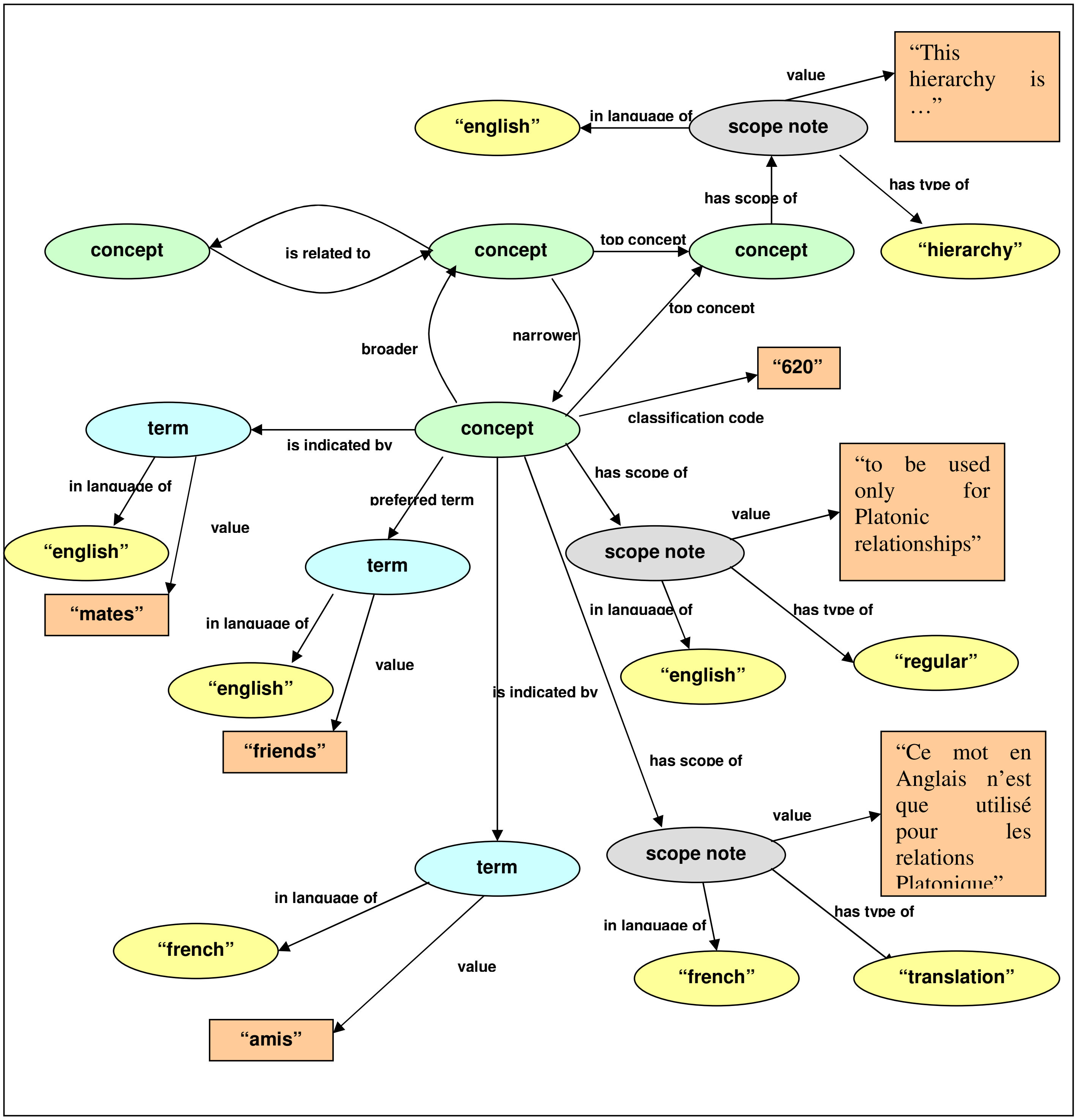

Figure 11. A RDF instance diagram for the multilingual *Friend* concept.

6.1.1 Standard Thesaurus API Access Functions

The standard thesaurus API access functions at level two would be expected to provide the following functions following the 6 relations using 2 or three letter acronyms used in the ISO standards for mono-lingual (ISO 2788) and multi-lingual (BS 6723:1985) thesaurus construction. Other functions are added to provide single relations used in the UNESCO thesaurus, which can be obtained by sequences of the simpler relations, but which would be more efficiently performed within the thesaurus management system:

1. Get all terms related to a term X (across trees) - RT
2. Get all terms broader than a term X (within a tree) - BT
3. Get all terms narrower than a term X (within a tree) - NT
4. Get all synonyms for a term – UF
5. Get the root term for a term – TT
6. Get the scope node – SN – extended for the 3 types of scope node used in the thesaurus model.
7. Get the whole Hierarchy
8. Get all terms containing word X
9. Get classification code for term X
10. Get term X in language Y
11. Get preferred term for X – USE

To specify some of these in more detail, we use the following generic definitions:

```
GetRootTerms
  takes no arguments (unless a multilingual thesaurus can have
    different roots for different languages, OR we want users
    to be able to request translation of the root term; in
    either case, this might take a target language as a
    parameter);
  returns the top-level terms of the thesaurus
  used by a thesaurus browser to enable top-down tree-like
    browsing

GetRelatedTerms
  takes a source term (variant: a classification code), source
    and target languages, and a relation
  returns a set of terms (variant: tuples containing term,
    target language, classification code, ...)
  the returned terms are those found under the thesaurus that
    relate to the given term in the respective languages;
    for some relations, the set will always contain one element.
```

Of course, we should not ignore the fact that a thesaurus should be able to describe itself:

```
GetThesaurusMetadata
  takes no arguments
  returns a description of the thesaurus
```

From this small API we can build a number of more powerful and more user-centered operations.

The above functions apply to a particular thesaurus. In a system with multiple thesauri (and possibly multiple thesaurus servers), clearly we need a parameter to identify them. We could use distinct URI's for the servers; we could extend this to each individual thesaurus (assuming one server can front-end more than one thesaurus). The following assumes there is only one thesaurus; extensions to the multi-thesaurus case are straightforward.

From these generic functions a number of derived functions can be built, with hardwired values for the parameters, e.g. GetNarrowerTerms, GetPreferredTerm, GetBroaderTerm.

Users could explicitly select the source/target languages and relation (as well as the term, of course); or the source language could come from the user's profile (in the host system) but in this case the user should always be reminded about the consequences of his/her settings.

An extreme case would be where the target language is chosen from the metadata for a particular data repository (so that the thesaurus is used to translate queries for particular data resources, possibly completely behind the scenes with respect to the user). This would require quite tight integration with the host system, and may not be possible with the architecture.

6.2 Interface between the Thesaurus Management Server and the Thesaurus Administration User Interface (API-2)

The thesaurus administration user interface will require all the functions of the thesaurus server API (API-1) with the addition of the following functions for thesaurus maintenance purposes:

1. insert new items in the thesaurus
2. update relations in the thesaurus
3. edit an existing item in the thesaurus
4. edit existing relations in the thesaurus
5. delete items in the thesaurus

6.3 Interface between the Thesaurus Management Server and the Indexing Tool (API-4)

The indexing tool will require the following functions form the thesaurus management system:

1. Get the set of Thesauri Supported – return list of thesauri, or null.
2. Get the list of preferred terms from Thesaurus X – return list of terms, or null.
3. Is Term X in the thesaurus – return 0 for false, 1 for true.
4. Get all terms containing string X – return list of terms, or null.

5. Get the whole Thesaurus Hierarchy from Thesaurus X – return list of terms, or null.
6. Get all synonyms to term X (UF relationship) – return list of terms, or null.

6.4 Interface between the Thesaurus Management Server and the Data Archive System (API-3)

The Data Archive System could be offered a range of API varying from the very simple function to a range of more advanced ones:

1. Get the whole thesaurus
2. Standard thesaurus API access functions
3. Re-use of the thesaurus server interface with a new client, therefore the thesaurus server interface must be fully documented.
4. The ability to modify the UI properties of the client and existing look and feel from the Data Archive System - *optional*
5. An API to higher level thesaurus functions, e.g. browse the thesaurus as a whole - *optional*

The later two levels of access will be considered in the detailed specification in the light of the effort and budget available to the project. If an existing multilingual thesaurus management system such as Ortellius from the University of Rome is used as the basis for modification rather than developing a system from scratch, the availability of these functions will also depend on the structure of such an existing system.

The API between NESSTAR and the Thesaurus Management System will call on the following functions:

In a Free Text Search

1. Check whether search term exists in thesaurus (eg ELLST) (GetTerm)
2. If term exists as a non-preferred term then switch to preferred (GetPref)
3. If term exists as a preferred term (directly or from 2) then get all that terms synonyms (GetUFS)
4. The NESSTAR system can then search on the term and all its synonyms
5. If search term not in the thesaurus (eg ELLST) then the NESSTAR system would search on entered term
6. If search is in another language then get translation (GetLang) and all synonyms in that language (GetUFS), then search as in 4
7. If users requests help then generate a KWIC listing (GetWord, GetStem, GetTrunc)

In a Controlled Vocabulary search

1. Check whether search term exists in thesaurus (eg ELLST) (GetTerm)
2. If term exists as a non-preferred term then switch to preferred (GetPref)
3. If term exists as a preferred term (directly or from 2) and the resources being search are all using the ELSST thesaurus then get notation code (GetCode)

4. The NESSTAR system can then search all resources on notation code
5. If term exists as a preferred term (directly or from 2) and the resources being searched are NOT all using the ELSST thesaurus then get translation for each language (GetLang)
6. The NESSTAR system can then search on preferred term in every language of resources selected
7. If search term not in the thesaurus (eg ELLST) then generate a KWIC listing (GetWord, GetStem, GetTrunc)

For Relevance Feedback

1. Translate keyword (GetLang)
2. The NESSTAR system can display keywords in user preference language

For Browsing Thesaurus (direct user interface to Thesaurus Management system ??)

1. Get all top terms (GetTTS)
2. The system displays all top terms so that user can select one
3. Get full hierarchy of selected top term (GetHier)
4. The system displays the full hierarchy of the selected top term so that user can select one
5. Get all relationships UF,NT,BT,TT,RT of selected term (GetThes)
6. Get notation code of selected term (GetCode)
7. Get all scope notes of selected term (GetScope)
8. The system displays notation code, scope notes and all relationships of selected term

Expansion of controlled vocabulary searches

1. Get all narrower terms (GetNTS)
2. Get all broader terms (GetBTS)
3. Get all related terms (GetRTS)
4. Get all relationships (GetThes)
5. The NESSTAR system can search on term and all relationships requested

8. If the resources being searched are all using the ELSST thesaurus then get notation code (GetCode)

6. The NESSTAR system can then search all resources on notation code

9. If the resources being searched are NOT all using the ELSST thesaurus then get translation for each language (GetLang)

7. The NESSTAR system can then search on preferred term in every language of resources selected

6.5 Interface between the Indexing Tool and the Data Archive System (API-5)

Indexing Tool Interfaces

The thesaurus end-user interface will be made available to users of the indexing tool so that they can browse the hierarchy in their preferred natural language to choose keywords and enter them into the metadata definition.

The use of the thesaurus end-user interface from the indexing tool may take various forms to support the Metadata Indexer browsing for terms, including a cut/paste function on terms in the hierarchy, cutting them from the thesaurus end-user interface and pasting them into the metadata document, or transparent access of the thesaurus from the metadata editing environment (e.g. MS-Word).

The user interface of the indexing tool will be able to support the following functions for the keyword creation task:

1) load a DTD
2) load a metadata file conforming to the DTD
3) save a metadata file
4) display a metadata file
5) scroll through a metadata file
6) add keyword
7) delete keyword
8) replace keyword
9) find string
10) run automatic keyword generation on a metadata file
11) select a thesaurus management system
12) select a thesaurus
13) The user will be able to select a menu item for Preferences, one of which will be Language, presenting a sub menu of the list of selectable languages.

The user will be able to use the following functions for the task of identifying terms which could be added to the thesaurus:

1) Create a file for a list of candidate thesaurus entries
2) Append a list generated from a run to an existing file

The menus, error messages and help information to the client will be pointers to text in data files to support multiple languages. Each data file will exist for each language. The languages of English, French, German and Spanish will exist, with the potential to add new languages.

Local Data Sources

1. stop list of common words to exclude from indexing search for each natural language supported
2. dictionary for each natural language supported – *optional*
3. thesaurus for each natural language supported - *optional*

Indexing Tool Operation Specification

Term – a term is a string containing a word alone, or up to five words conforming to the definition for the thesaurus.

The sequence of operation of the indexing tool on an input metadata file will be:

1. Read in XML DTD
2. Parse DTD into tree
3. Read in XML file
4. Parse XML file into tree
    4a. Create File or Append to file for terms not found in thesaurus
5. Navigate XML file to derive the thesaurus that defines the limited vocabulary from which keywords should chosen for this metadata file
6. API call 1 to TMS to get list of thesauri supported
7. Match that thesaurus from 5 is in set from 6 – exit if fail
8. Navigate XML DTD to identify
    a) Keyword field
    b) Associated text field from which keyword should be derived
9. Navigate XML file to extract text string from which keyword should be derived
10. Wheel stop list (local data source 1) for language against text string to exclude terms
11. Root reduction on excluded text string – *optional*
    11a. Pluralise terms to meet thesaurus standard
12. Loop through excluded text string – for each term
    API call 3 to thesaurus to get synonyms
    If non-null return then put term in keyword list
    If null returned then put term in thesaurus candidate list or increment word frequency of existing term
13. Match excluded text string against local network and cat return to keyword list - *optional*
14. Get preferred terms for terms in keyword list
    14a. Put preferred term list into XML file at location of keyword
15. Continue at 8 until all keyword fields have been addressed
16. Present the metadata file to the user for manual browsing, supporting an insert and replace functions over the keywords selected automatically, or empty keyword fields. The user should be able to browse the thesaurus hierarchy at the same time to cut/paste words into fields.
17. Write out metadata file and close

Design option for step 11 - root reduction on excluded text string

For English, French, Spanish and German it would not be possible to write code to reduce words to their roots within the effort and budget available to the project.

If pre-existing libraries or tools can be obtained to provide these functions for research use within the project, and for exploitation after the project, within the budget available to the project, then they can be included.

Design options for step 13 - match excluded text string against local network

In a perfect world this stage would perform full natural language understanding on the input text string or paragraph, and use this to locate the appropriate term in the thesaurus management system. Such a complex application of natural language processing techniques is not justified in this case, so simpler language technology approaches will be considered.

There are two options for this stage in processing which would introduce a local data source to provide extra power in providing keywords.

Use of commercial natural language dictionaries or thesauri.

Words in the excluded text string could be matched against a dictionary of that language to provide the extra terms used in the definition. This would increase the size of the hit list, but would also introduce false positives (e.g. where a definition includes opposites etc).

Words in the excluded string could be matched against a thesaurus to provide a set of synonyms as extra terms. This would increase the size of the hit list, but would introduce false positives since word sense would be ignored.

If the cost of acquiring dictionaries or thesauri for use within the project or for future exploitation is within the budget of the project, then this approach will be investigated to see if the benefits of the increased hit list outweigh the errors introduced.

Probabilistic Information Retrieval links between text and thesaurus

The multilingual social science thesaurus is expected to hold about 1500 terms. Each metadata document will contain up to 500 fields each with between one and 500 words of natural language text in them.

It could be possible to use probabilistic information retrieval techniques to build a weighted network of links between terms, their co-occurrence etc. in the natural language text and terms in the thesaurus.

Such a network would have to go through two phases, one of learning, the other of use. In the learning phase the pre-existing relationships between natural language texts and keywords would be taught to the network. In the use phase, it would be fed the natural language text, and return keywords above a threshold of probable association.

In the case of the UK data archive, there already exist about 5000 metadata records which are already indexed by terms in the thesaurus, so the learning phase could take place during the project, prior to use. For the other data archives, there are pre-existing metadata records, but they are not indexed by keywords in the thesaurus, so learning would have to take place during the first period of use.

Several information retrieval techniques are candidates including neural networks and Bayesian reasoning. These will be briefly investigated, and if they appear to provide sufficient benefit, and can be implemented within the budget and effort available within the project, then one will.

6.6 How might the thesaurus end-user interface use API-1?

"*Search mode*": initiated by user entering a term, source and target languages, then choosing to find related terms (via a set of buttons, one for each relation). Of course, the source language may already be determined from the user's profile; the target language(s) could be selected from a fixed list. When the results come back, the system moves into Browse Mode.

"*Browse mode*": if the user starts from scratch, then the root terms of the thesaurus are displayed (variant: the top 1 or 2 levels of the broader/narrower tree). If the user has asked for terms related to their initial term, then a list of those terms is displayed (variant: a tree showing how they relate to the original term). The user should be able to select any of these terms (more than one at a time) and search for terms related to them.

For tree browsing, we presume that it would be more efficient for the thesaurus browser to talk to the thesaurus server in terms of classification codes rather than terms, wherever possible. (Otherwise, the thesaurus will waste time searching for the tree-position of the same term that it has only recently returned to the browser).

If no related terms are found, the browser should say so, but may also offer:

- nearby terms under a different relation (for example, if the original relation was equivalence, then offer a broader term)
- to work with a broader term in the source language

6.7 How might a separate Data Archive System query-building tool use API-1?

Clearly, it means having a reasonably tight coupling between the query builder and the thesaurus server, or maybe the back-end of the thesaurus browser. This query building tool is provided in the client data archive system, not in the thesaurus management system.

In this tighter integration, the query-building tool (qbt) would call the above API directly. Amongst the possible models of use are:

1. *(user driven, but not using the thesaurus browser).* Add extra controls to the qbt that allow the user to set source/target language, thesaurus search relation, etc.; the user sets these, and then asks to translate the query terms. For each term in the query, the qbt calls GetRelatedTerms; if the result is a singleton, this is substituted in the query (or more likely, a copy of the query

This results in the construction of a new query. The simplest model is to generate a new, translated query. Or the qbt could build a compound query, along the lines of:

((original query in source language) and (language = source-lang))
OR
((translated query in target language) and (language = target-lang))

When querying multiple data repositories with multiple languages, this could lead to large queries being sent to each of them; and we're relying on those that don't deal with a particular language quickly pruning the query (and not searching for the same phrase in inappropriate languages).

2. *(Implicit).* qbt tailors the original query by obtaining metadata of the required repositories and using this to determine the target language(s) for each repository. The query for a particular target language is obtained by applying GetEquivalentTerm to each term in the query.

If no equivalent term is found then a null response will be provided by the Thesaurus Management System.

7. Conclusion

The overall architecture of LIMBER can be seen as consisting of the following elements which require specification, design and implementation:

Multilingual Thesaurus Management System

Thesaurus Management Server

Acts as a front-end to one (or more) thesauri; receives configuration information and operation requests from thesaurus end-user interface.

Thesaurus End-User Interface

The user-client-side component; capable of supporting both "one-shot" and fully interactive thesaurus browsing.

Thesaurus Administration User Interface

The thesaurus administration user interface will include all the functionality of the thesaurus end-user interface with the addition of few functions for thesaurus maintenance purposes.

Thesaurus

LIMBER is reducing and translating the HASSET thesaurus of social science terminology into French, German and Spanish in addition to the existing English terms. This multilingual thesaurus will be formatted into an RDF representation for storage and use by the thesaurus management system.

Indexing tool

It uses the thesaurus through the thesaurus management system and generates metadata data documents to be loaded into the data archive system.

Data Archive System Multilingual User Interface

LIMBER will not solve the generic problem of multilinguality in user interface to Data Archive Systems, but it will provide the NESSTAR system with data files of interface terms and help information translated into French, German and Spanish to add to the existing English and Norwegian interfaces.

Metadata Format

The LIMBER consortium has presented an RDF format proposal to the DDI format committee for its adoption as the international standard for metadata in social science data archives. It is planned to submit a proposal for a higher level conceptual model than individual representation language bindings, and with wider coverage than the current questionnaire data alone.

This architecture will meet the objectives of the project, and of the wider value chain involving European data archives in that:

Potential to result in four commercially exploitable products and a public domain metadata standard to encourage the adoption of the products from which training and consultancy exploitation is possible.

In relation to the business model in the introduction (and more completely discussed in the user requirements deliverable D1), the multilingual thesaurus management system and the multilingual user interface to the data archive system provide benefits to users in wider access to data and de-skilling of metadata indexers.

The wider access to data by users has the consequence for evidence based decision making and planning in business, local, national and European levels of government which will be improved with the increased access to data.