

**510035**

**CACAO**

**Definition of the structure and programmatic interfaces for  
components access.**

<b>Deliverable number</b>	<i>D.1.2</i>
<b>Dissemination level</b>	<i>Public</i>
<b>Delivery date</b>	<i>30 May 2008</i>
<b>Status</b>	<i>Final</i>
<b>Author(s)</b>	<i>Alessio Bosca</i>



***eContentplus***

This project is funded under the *eContentplus* programme<sup>1</sup>,  
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.

---

<sup>1</sup> OJ L 79, 24.3.2005, p. 1.

## Table Of Contents

<b>1 GOAL</b> .....	<b>3</b>
1.1 GENERAL VIEW.....	4
<b>2 CORPUS ANALYSIS</b> .....	<b>4</b>
2.1 ENRICHEDTRANSLATION_MANAGER.....	5
2.2 WORDTOCATEGORY_MANAGER.....	6
2.3 CORPUSTHESAURUS_MANAGER.....	6
2.4 INDEX_MANAGER.....	7
<b>3 QUERY PROCESSING</b> .....	<b>7</b>
3.1 NATURAL LANGUAGE PROCESSING WEB SERVICE.....	8
3.2 QUERYPROCESSING ACTIVITY DIAGRAM.....	9

## WP1: Multilingual access

<b>Work package number :</b>	<b>1</b>	<b>Start date:</b>	<b>0</b>	<b>End date:</b>	<b>12</b>						
<b>Work package title:</b>	<b>Multilingual access</b>										
<b>Applicants involved:</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>7</b>	<b>8</b>						

### 1 Goal

The goal of this work package consists in the design and configuration of a standard information retrieval system that allows users to type some words in his/her language and retrieve all the relevant books or texts in all the available languages.

In the context of the work package, the aim of this document consists in providing a description of the architecture and of the programming interface (API) that will be used in the CACAO project in order to reach this goal. In particular the document covers the different software modules that enables multilingualism in CACAO and describes how these components will be accessed and integrated in the system.

Since this document is strongly related to D.3.1 (that already presented the programmatic interfaces of the different components along with the definition of CACAO architecture), such contents will constitute an important reference for this document. The diagrams and API proposed in D.3.1 are in fact also repeated here, however this document specifically focuses on information retrieval components of the CACAO architecture.

Currently a prototype of the Cross Language Information Retrieval system has been designed for the participation to a track of the CLEF campaign (see <http://www.clef-campaign.org/>) specifically concerning digital library metadata. CLEF campaign offers an infrastructure for testing, tuning and evaluation of information retrieval systems operating on European languages in both monolingual and cross-language contexts; therefore it constitutes a very interesting opportunity for testing and validating the information retrieval system promoted by the CACAO consortium.

## 1.1 General View

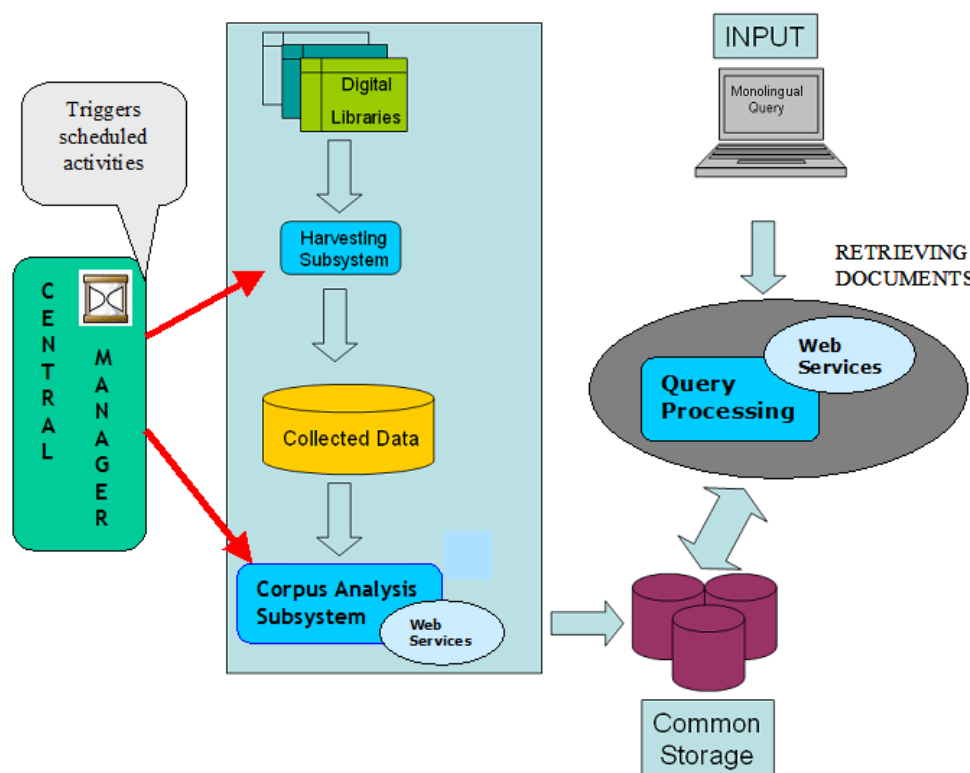


Figure 1- Architecture Overview

Figure 1 presents a general overview of the architecture of CACAO system (described with finer details in the first deliverable of WP3). The three main activities of the CACAO framework (depicted in the diagram as light blue squares) comprise data harvesting, corpus analysis and query processing. The first two activities are off-line tasks scheduled by the Central Manager, while the last one is triggered by users requests.

The Cross Language Information Retrieval (CLIR) capabilities of the system originate both in the processing and analysis of collected data (i.e. text lemmatisation, words distribution over categories) as well as in the translation and enrichment of query terms in order to yield documents in other languages.

Therefore in the following sections the software components involved in such activities (Corpus Analysis and Query Processing) will be presented along with their programmatic interfaces (API), delineating the elements presented in deliverable D3.1 and integrating the overview on multilingual resources of deliverable D2.2.

## 2Corpus Analysis

The Corpus Analysis activity aims at processing the data harvested from digital libraries in order to infer new information as the distribution of words within librarian categories or the semantic similarity of terms. Being corpus-based information specifically tailored to the domain covered in the data, it could be very useful in order to enrich and improve the general purpose linguistic data (i.e. dictionaries) or to automatically disambiguate among translations and exclude the irrelevant ones.

These components will be activated every time the library's catalogues are updated. The information extracted from corpus and log analysis will be maintained within CACAO system and only accessed by means of internal components. The following picture shows the components involved in corpus analysis along with the programmatic interface of the methods realizing the data analysis.

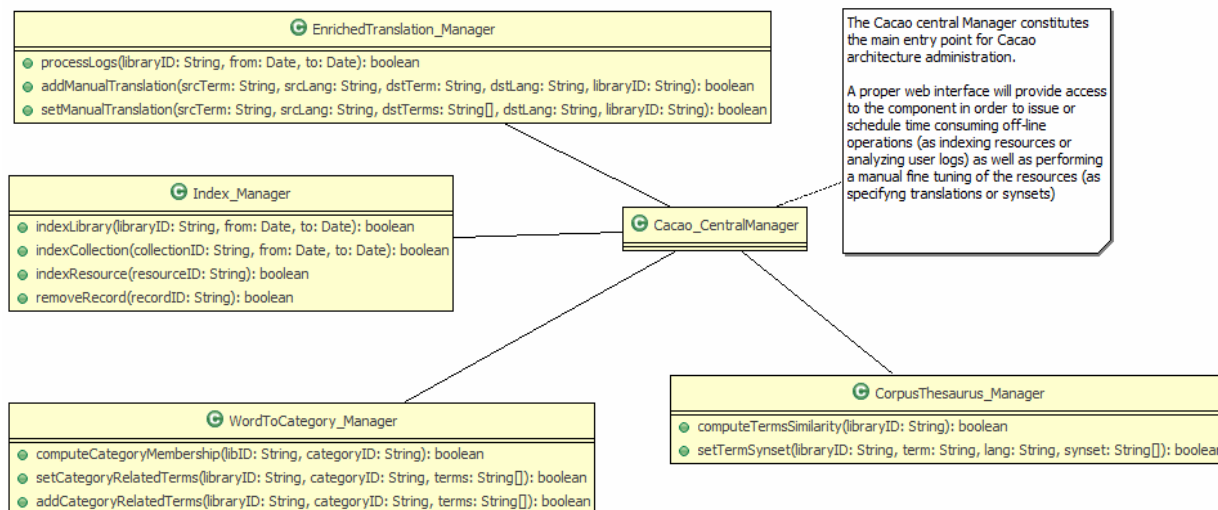


Figure 2 - Corpus Analysis Components

## 2.1 EnrichedTranslation\_Manager

The *EnrichedTranslation\_Manager* is devoted to analyse library logs that comprise the history of users requests, particularly in the context of multilingual libraries (as university ones) where users commonly use different languages for their queries. For a given session, users will probably try different queries in order to search for the same information, thus implicitly providing translations or synonyms that could be added to the system resources (for more details on this component see D.2.2 on multilingual resources).

The component programmatic interfaces comprises the following methods:

- *boolean processLogs(libraryID, from, to)*: processes users logs, from a given library in a time interval
- *boolean seManualTranslations(srcTerm, srcLang, dstTerm[], dstLang, libraryID)*: convenience method for manually setting the translations.
- *boolean seManualTranslations(srcTerm, srcLang, dstTerm, dstLang, libraryID)*: convenience method for manually adding a translation.
- *String getManuallyCodedTranslations(term, langFrom, langTo)*
- *String getLogsBasedTranslation(term, langFrom, langTo)*

## 2.2 WordToCategory\_Manager

The *WordToCategory\_Manager* main activity consists in analysing the textual resources associated to a given librarian classification (see D.7.1 for an overview of classifications systems within CACAO) and extracting relevant words from them. Such textual resources include the labels associated to the classification category and Subject Headings as well as

titles of the volumes pertaining to that class (and abstracts whenever available). The outcome of this operation will be accessed in order to remove additional meanings from the translated terms during the query-processing phase (for more details on the component see D.3.1, section 3.3.2).

The component programmatic interfaces comprises the following methods:

- *boolean computeCategoryMembership(libraryID, categoryID)*: it computes the relations between words and categories.
- *boolean setCategoryRelatedTerms(libraryID, categoryID, terms[])*: convenience method for manually setting the relations.
- *boolean addCategoryRelatedTerms(libraryID, categoryID, terms[])*: convenience method for manually adding elements to a already existing relation.
- *String[] getCategoriesByWord(libraryID, term, lang)*: retrieves a list of categories related to a given word.
- *String[] getWordsByCategory(libraryID, category, lang)*: retrieves a set of terms related to a given librarian category.

### **2.3CorpusThesaurus\_Manager**

The *CorpusThesaurus\_Manager* aims at creating a resource that associates to every word a flat list of related terms as a means to perform monolingual query expansion as well as disambiguating queries with multiple words. It analyses textual resources and builds a semantic word space model by exploiting Latent Semantic Analysis or similar approaches in order to create the term lists.

Vectors corresponding to different target words in multi-words queries will be accessed in order to measure semantic similarity among target translations, thus issuing a semantically coherent query to the system (for more details on the component see D.3.1, section 3.3.3.).

The component programmatic interfaces comprises the following methods:

- *boolean computeTermSimilarity(libraryID)*: it computes semantic similarity of terms on the basis of term co-occurrences within the corpus
- *boolean setTermSynset(libraryID, term, lang, synset)*: utility methods allowing to manually specify synsets for a given term.
- *boolean getRelatedTerms(libraryID, term, lang)*: retrieves a flat list of word related to a given term

### **2.4Index\_Manager**

The *Index\_Manager* is designed to index the metadata collected from libraries by means of a standard software tool as Lucen and make them available for users queries. The outcome of the indexing process consists in an inverted index, a data structure commonly used in Information Retrieval applications in order to obtain a fast access to documents starting from contained terms or metadata; typically terms from user queries are used as search keys in order to retrieve documents in the index.

The component programmatic interfaces comprises the following methods:

- *boolean indexLibrary(libraryID, from, to)*
- *boolean indexCollection(collectionID, from, to)*

- *boolean indexResource(resourceID)*
- *boolean removeRecord(recordID)*.

### 3Query Processing

The *Query Manager* coordinates the different phases and activities of query processing that include analysing, translating and expanding query terms.

The *NLP Web Service* performs linguistic analysis on the user query in order to obtain a lemmatisation of the input text and to identify named entities (i.e. persons, places, companies) within the query.

In order to get translation alternatives this component relies on the *Translation WS* and exploits *WordToCategory\_Manager* resources as a means to disambiguate them. It could also use the output of the *CorpusThesaurus\_Manager* to “simplify” the translation space. In case of missing translations it tries to substitute the translation with a guessed translation exploiting internal resources of *EnrichedTranslation\_Manager*.

The query expansion is performed by means of the *CorpusThesaurus* resource as well as by means of the external Thesaurus Web Service; however in certain cases it might access a specific external resource for performing non linguistic expansion (e.g. geographical entities).

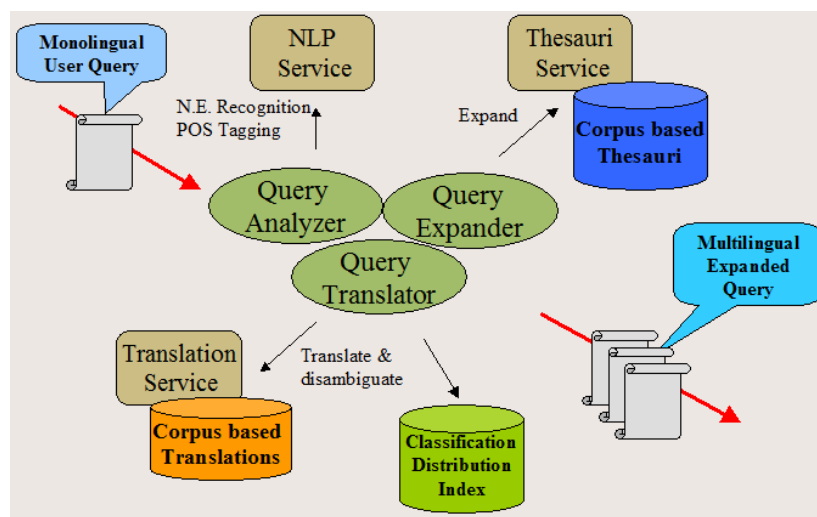


Figure 3 - Query Processing overview

Figure 3 presents a sketch of the interactions of the Query Processing subsystem with the other components (Web Services are depicted as rounded boxes while cylinders represent the data produced by the Corpus Analysis subsystem).

A set of different components is devoted to process the original monolingual user query, transforming and enriching it by means of translations and expansions.

The Translation Web Service aims at translating the queries produced by the user in her own language to the different target languages while the Thesauri Web Service is required by CACAO system to perform tasks such as query expansion or disambiguation.

Both Translation and Thesauri Web Service are specifically covered by D2.2 therefore they will not further described in this document.

### 3.1 Natural Language Processing Web Service

Natural Language Processing Web Service has the goal of performing the following tasks:

- Lemmatization (for query analysis: reduction of one word to one or more stems)
- Lemmatization with part of speech (POS) disambiguation (for indexing: reduction of one word to one stem syntactically disambiguated)
- Named Entity recognition (for extracting named entities (Person names and Geographical Names) out of metadata and full texts). For instance, book titles often contain references to locations or people. A book about “Turkey” as a country should not be indexed as a book about the homonymous bird.

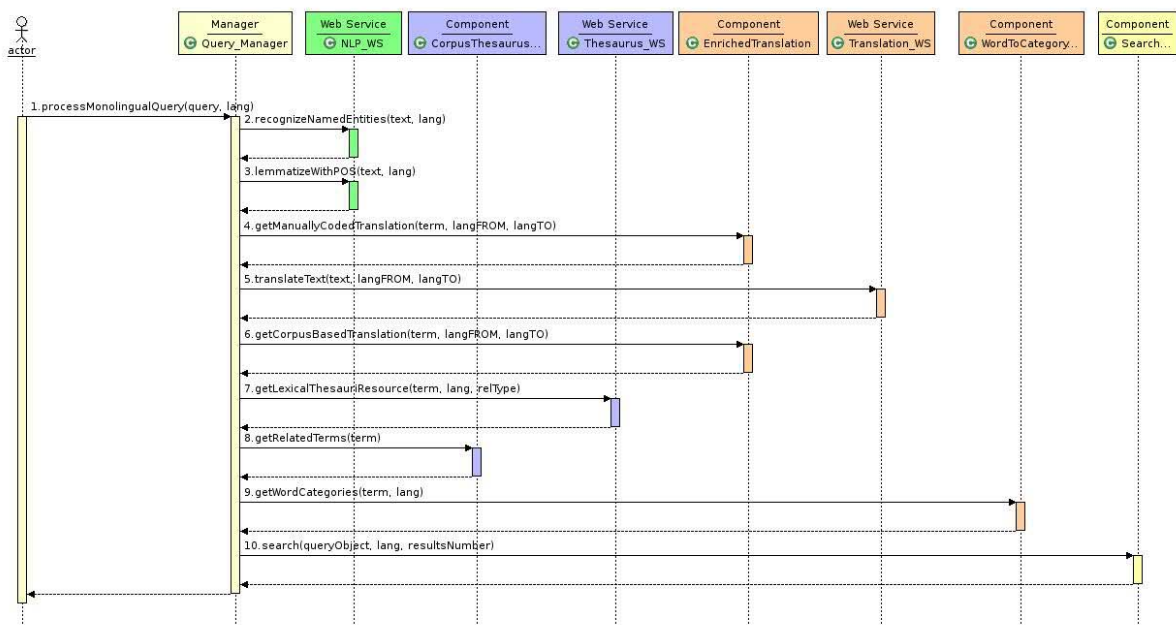
A crucial issue is represented by the “alignment” between the output of NLP WS and the input expected by Translation WS as terms would be lemmatised by NLP WS before being translated to target languages. Two possibilities arise: either the Translation WS is able to perform a morphological analysis (e.g. Machine Translation Systems) or it is not (e.g. bilingual dictionary lookup). In the former case, lemmatisation by the NLP WS can be performed on the target language, so no alignment problems arise. In the latter, the system must ensure that the lemmatised words in the source language are valid keys for retrieving translations from the translation service.

The component programmatic interfaces comprises the following methods:

- *String lemmatize(String text, String language)*
- *String lemmatizeWithPOS(String text, String language)*
- *String recognizeNamedEntities(String text, String Language)*

### 3.2 Query Processing activity diagram

The following diagram describes in details the various phases that a user query must undergo in order to provide her with desired results (hits of catalogue):



*Figure 4 - Query Processing: sequence diagram*

As it can be noticed the user query is passed to the *Query\_Manager* (1) first undergoes the process of recognition of named entities (2) and lemmatisation(3). At this point the system is in the position of looking for translations in the target language. The system will privilege manually coded translations (4) over translations provided by a Web service(5). Translated terms are then expanded in order to increase the recall of the query (6 and 7). In the case in which none of the translation components is able to provide a translation of (some of) the query terms, then the fall-back solution of the *WordToCategory\_Manager* is adopted in order to provide the user at least with partial topic-oriented results (9).

All the gathered information (original terms, lemmatised terms, translations, expansions, categories, etc.) is searched in the inverted index generated by the index manager in order to retrieve relevant hits(10).