Grant Agreement Number: 257528

KHRESMOI
www.khresmoi.eu

State of the art, concepts and specification for the "Early software architecture"

Deliverable number	*D6.3.1*
Dissemination level	*public*
Delivery date	*10th of May 2011*
Status	*Version 1.1*
Author(s)	*Emmanuel Jamin, Vassil Montchev, Konstantin Pentchev, revised by Allan Hanbury*

This project is supported by the European Commission under the Information and Communication Technologies (ICT) Theme of the 7th Framework Programme for Research and Technological Development.

Executive summary

This document reports on the foundations of the architecture and integration aspects of the Khresmoi project. This includes the technical and functional requirements that imply technical choices and integration decisions for the architecture design. To provide the architecture description and the system specification, an iterative approach has been followed. The structure of the document reflects the main aspects of this approach.

At first, two kinds of state of the art are presented. The first one compiles the systems that use similar technologies into four different domains (information retrieval, semantic web systems, multimedia search engines, and platforms for natural language processing). This permits to situate the technical position of the Khresmoi system and provides recommendations in terms of technical requirements. The second state of the art retraces the main technologies for the system integration. Based on this state of the art, the decision is made to adopt the SOA approach, as it offers the best features in terms of flexibility and scalability to be used in the Khresmoi system. In the list of SOA approaches, the SCA approach provides the most generic specification of software components and the mechanism to compose services. Thus, SCA provides a useful software infrastructure to integrate and deploy various software components as loosely coupled services.

Based on the general requirements and the integration decisions, the abstract and logical structure proposed by the SOA approach is used to define the Khresmoi architecture. Then the logical layers and their respective functional modules are described. After several iterations between top-down (components analysis) and bottom-up (user requirements) approaches, the main services of Khresmoi were identified.

Finally, as the main components of the system were identified, their specification as services and their interactions with other services are described. The integration mechanism proposed by the Service Component Architecture (SCA) is described as an integration guideline and examples of integration are provided. To complete the specification section, the development environment is described. Descriptions of tools and development guidelines are provided.

1 Table of contents

2 List of abbreviations

DoW	Description of Work
TSMC	Technical and Scientific Management Committee
EC	Exploitation Committee
PC	Project Coordinator
STC	Scientific and Technical Coordinator
PA	Project Administrator
AB	Advisory Board
WP	Workpackage
WPL	Workpackage Leader
QA	Quality Assurance
R1	Reporting Period 1 (means the first of the four monthly reports)
Tbd	To be defined
TM	Technical Manager
PM	Person Month
CSA	Chief Software Architect
TET	Technical Expert Team
AR	Assigned reviewers
KIF	Khresmoi Integration Framework
SOA	Service Oriented Architecture
SCA	Service Component Architecture
EAI	Enterprise Application Integration
ESB	Enterprise Service Bus
UP	Unified Process
SOMA	Service-Oriented Modeling Architecture

Table.1: Abbreviations and acronyms

3 List of figures

4 List of tables

5 Introduction

5.1 Objectives

The Khresmoi project is an Integrated Project that should specify and implement a new generation of search engine focused on medical information. As many different technologies, such as multilingual translation, image and textual analysis, are now mature enough to index many different kinds of information and to enable powerful search functionalities, one of the most important challenges of the project is to integrate these technologies to enable very powerful search functionalities.

To achieve this goal, the project is organised with three important milestones that correspond to three important statuses of the Khresmoi system. As planned in the DoW, the three important steps are planned over the four years:

- Software design (M12): the first year is dedicated to design the basic architecture that will support the software integration.
- Cloud Infrastructure (M24): the second period of the project will define the best cloud infrastructure to deploy the software and to obtain a scalable system.
- Integration (M36): finally, the last period will be used to deploy the system in a real situation and deal with many different repositories to provide relevant information according to different contexts.

These three points follow the steps fixed by a classical software engineering method to synchronize the specification and implementation tasks and ensure that difficulties are addressed progressively all along the project.

In the early stage of the project efforts were focused on analysis to identify the best integration technologies and then design a coherent software system. This activity was done in parallel with other activities in the project that are related to the main requirements for the search engine. These requirements depend on the software components that are developed in the technical work packages (WP1–WP5) and on the user requirements identified in WP8 and WP9.

In this deliverable, we explain how the complementary requirements need to be taken into account because their relative restrictions have an important impact on the architecture design. To introduce the followed approach to achieve our goal, the main aspects of the integration framework are presented.

5.2 Summary of the Integration Framework

A main focus in the distributed development of a system is the specification of the architecture and the integration processes to support the integration of several components into a single system. It is also relevant to propose a modular configuration of the system to fit with the different use cases defined in the project.

Architecture and integration standards are separated into different aspects of development. The architecture describes the structure of a system while the integration focuses on the process of fulfilling this architecture. The architectural standards focus on the correlation of all services and components and the “integration” of external legacy systems into the Khresmoi system while the integration aspects focus on the set of tools, procedure and interfaces to support the communication and computation of Khresmoi. Therefore the ICT architectural and integration aspects focuses on two main processes:

1. the definition of an integrated architecture supporting the conceptual structure of the Khresmoi system and,
2. the set of processes and tools to support the distributed development and further combination into one system.

5.2.1 Approach

The approach used for the Khresmoi specification and development is a classical approach of software engineering. It is based on the Unified Process (UP), in particular the OpenUP[1] that integrates the main Agile principles in the method to enrich it. From this method, we essentially focused on two important aspects for the organisation of the Khresmoi project: the conceptual and the technical integration.

5.2.2 Conceptual integration

At first, we have to consider the conceptual integration, platform-independent, computer-independent, which focuses on the business and process aspects of the project. We have to identify and describe the most relevant business processes (from the users' and other stakeholders' perspectives) that have to be reified by the integrated software implemented within the project. This task determines the technical integration aspects to consider. This conceptual integration implies:

- Identification and description of the main Use Cases that the integrated software solution should reify.
- Identification of actors/roles and software components that participate in these Use Cases.
- Identification of data objects (messages) interchanged by actors and components during the enactment of project Use Cases.
- Identification of preconditions/effects that influence components and the environment during the enactment of the processes and for each message exchange.
- Technical description (behavioural) of processes, components, messages, etc.

This conceptual integration is a key factor for the technical integration, since otherwise it is not possible to make correct architectural and technical choices. Moreover, this conceptual integration ensures a correct provisioning of components and data objects for the successful enactment of project use cases. Furthermore, potential integration misconceptions are identified and solved at this stage.

5.2.3 Technical integration

The technical integration aspects need to be considered carefully, such as integration architecture choices, tight versus loosely coupled (SOA, IDE, etc) architectures, development frameworks (JAX-WS, etc), an integration roadmap and plan, team building, etc.

The technical integration can start in parallel to the conceptual one but technical choices are only relevant after considering the results of the conceptual integration.

The technical integration has to consider the following aspects:

- Team building to organize the team in charge of the specification and the development. Identification of the main team roles and responsibilities.
- Project management to schedule all the important tasks and synchronize them to facilitate the integration. Setting of tight deadlines, identification of task dependencies, detection of deviations and correcting actions, etc.

[1] http://es.wikipedia.org/wiki/OpenUP

- Development guidelines to define common practices of development in a shared environment.
- Architectural and technical solutions (SOA approach, ESBs, etc.) to analyze which solution is the most appropriate regarding the Khresmoi requirements. This includes a detailed integration architecture design, based on the work done for the conceptual integration described above. This detailed design provides structural and behavioural views (depending on the requirements) of the whole integrated project solution.
- Implementation and instantiation of the integration environment, endowing developers with integration tools.

The description of the environment for development will be summarised at the end of this document.

5.2.4 Work plan for the integration

As explained before, the work plan is an important factor of success. As we decided to follow the OpenUP process, we should apply the different steps for every milestone of the project the work plan. This is summarised by the following table (Table 2):

Project milestones	**OpenUP steps**	**Khresmoi tasks and status**
M12 **Software system**	Inception	- State of the art about software solutions - Components analysis - User requirements
	Elaboration	- Specification of the architecture - Integration decisions
	Construction	- Implementation of the Early prototype
	Transition	- Test / evaluation of the Early prototype - Refinement of the architecture
M24 **Cloud infrastructure**	Inception	- Software analysis - State of the art about Cloud infrastructures
	Elaboration	- Specification of the Cloud infrastructure - Specification of the full prototype for the Cloud infrastructure
	Construction	- Installation of the Cloud Infrastructure - Deployment of the full prototype in the cloud infrastructure
	Transition	- Test / Evaluation of the Cloud prototype
M36 **Integrated System**	Inception	- Requirements for the real situation
	Elaboration	- Specification of the Integrated Prototype - Preparation of the deployment of the Integrated System in a real situation
	Construction	- Deployment of the system in the real situation (Cloud, data, end users and connection with external components)
	Transition	- Test / Evaluation of the Integrated System

Table 2. OpenUP Steps according to the Khresmoi milestones

For every milestone, the different versions of the software are specified in UML. Then, especially for the Elaboration step, several UML diagrams are produced to define clearly what the developments tasks will be.

5.3 Plan

This section provides the outline of Sections 6–8.

5.3.1 State of the art

As the Khresmoi project aims to go beyond existing search engines by integrating advanced scientific results, we need to learn from past experience and analysis of existing systems. The state of the art has several objectives specially oriented towards the architecture design. Interesting integration principles can also be deduced from the relevant results identified in the literature.

Our main objectives to be achieved by the state of the art are:

- Find the functional characteristics that could have an impact on the architecture.
- List the important functionalities:
    - for positioning the technical improvements that the Khresmoi project can offer,
    - to ensure that important aspects are not forgotten.
- Take advantage of the lessons learnt in other projects and in consequence try to avoid the same errors.

For this, four scientific domains relevant to the project were identified. Each one proposes very interesting and advanced solutions for concrete problems targeted by the Khresmoi project. These domains are Information Retrieval, RDF repository for information retrieval, multimedia search systems, and the NLP platform. For every domain, the best systems and technologies are analysed to compare and enrich the Khresmoi solutions. Finally, a study was also realized to compare the existing approaches for integration in software engineering. For every approach different solutions are presented.

The conclusion of the SotA should provide a list of recommendations for our technological choices and integration decisions.

5.3.2 Concepts of the Khresmoi architecture

After the identification of the important functionalities provided by the state of the art, the component analysis and the user requirements, the Service Oriented Architecture (SOA) technologies seems to fit very well with the integration requirements. Effectively, the SOA approach aims to implement loosely coupled services that could be a flexible baseline for the Khresmoi system.

On the other hand the SOA is a very abstract way to define software systems. So, to define more concretely the Khresmoi system, a method is followed to identify and specify the main services. The Service-Oriented Modeling Architecture (SOMA) provides guidelines to take into account the use cases but also the existing software components.

Based on the SOMA approach, the logical structure and the corresponding services will be identified. The logical view is composed by different functional categories that are composed of atomic services with functional similarities. Thus, the Khresmoi architecture corresponds to the functional decomposition into different functional layers and categories.

To complete the architecture concepts, the principles of the service's integration provided by the SCA specification will be presented. Then, the specification of the SCA formalism and implemented by the Apache Tuscany framework will impose some integration guidelines and explain how the Khresmoi architecture can be integrated and deployed as a distributed and flexible system.

After the section related to the status of the technologies/systems, the Khresmoi architecture will be presented. At first, the very general principles of the architecture will be introduced. Based on this very abstract overview, the technical choices to implement the architecture will be presented. Finally, before presenting the specification of the different Khresmoi components and their interfaces, the integration strategies will be discussed according to the technical requirements.

5.3.3 Specification of the Khresmoi system

The state of the art and the different analysis (components and user requirements) permitted to define the architecture (the logical structure of the system). Based on this structure, the main services have been defined and the ways to integrate them are explained. Thus, now we need to specify concretely the services. For every service, the interface and the references should be defined. Also, the different possible bindings will be presented.

When all the services are clearly specified, the scenario for integration is presented. Effectively, the different services are integrated according to specific sequences. Even if the communication between the services is dynamic according to the interactive process of the user, the main sequences can be defined. Then, some service composition can be defined to express some specific integration between the components. The composition of different services with the SCA approach is called a composite.

Finally, after the composites specification, the system can be deployed easily within the SCA runtime.

6 State of the art

6.1 Introduction

The Khresmoi project encompasses a large set of existing technologies (partner's components) that should be adapted, extended, and integrated to support the functional objectives of the project. Accordingly, the chosen architectural approach is based on service-oriented principles. A Service-oriented Architecture (SOA) is an architecture principle that is composed of several loosely coupled services, and supports the composition and re-use of them, where a service is a basic software component that provides a specific functionality. A service can be considered as the functionality provided by an existing component or the new functionality implemented especially for the Khresmoi system.

The search engine for medical information should provide specific functionalities:

- Different kinds of resources (documents, web pages, etc.),
- Different media support (image, text, etc.),
- Translation into different languages (French, Czech, German and English as a default language),
- Different kinds of information (granularity of the knowledge) for different user profile (expertise levels)
- Different mining techniques to produce several complementary indexes used by the search engine.

The functional challenge is a very high level of expectation, in particular regarding the integration aspects:

- Flexible, scalable, robust, distributed architecture,
- Functional integration, and
- Data integration, in particular with the Linked Data

For this, a prerequisite study of existing systems is very important. For the architecture, we have two important objectives regarding the state of the art. The first one is a technical objective; it should analyse the different systems to find and establish the main functionalities and their impact on the architecture. The second objective is related to the integration approach; it should study the different technology available for system integration to find the best way to integrate the components and to build a robust and scalable system.

To find the important characteristics and study the functional comparison with existing systems or similar research projects, we present:

- Four states of the art that are dedicated to the main domains of Khresmoi, allowing the work to be influenced by these and to take care of possible errors to avoid. Specificities of those domains imply some restrictions and requirements that could have an impact on the architecture.
- After the identification of the possible restrictions/requirements, the different approaches to integrate software components for the implementation of a large system will be also studied. The result should provide us with a clear view about the technology to be used for the architecture implementation.

6.2 Architectural viewpoint

6.2.1 Architecture for Information Retrieval systems

An Information Retrieval system for biomedical information needs to combine features from the domain of large-scale full-text search and retrieval systems with features that leverage the semantic schemas and ontologies that exist in the biomedical space. The drivers for the architecture of semantic systems are described in the next section. With regard to the text search, we have identified the following potential search engines:

- Lucene (http://lucene.apache.org) / Solr (http://lucene.apache.org/solr/)
  - **"**Apache Lucene Core (formerly named Lucene Java) [provides] Java-based indexing and search implementation, as well as spellchecking, hit highlighting and advanced analysis/tokenization capabilities."
  - "Solr is the popular, blazing fast open source enterprise search platform from the Apache Lucene project. Its major features include powerful full-text search, hit highlighting, faceted search, dynamic clustering, database integration, rich document (e.g., Word, PDF) handling, and geospatial search. Solr is highly scalable, providing distributed search and index replication, and it powers the search and navigation features of many of the world's largest internet sites. "
- MIMIR (http://gate.ac.uk/family/mimir.html)
  - "Mímir is a multi-paradigm information management index and repository which can be used to index and search over text, annotations, semantic schemas (ontologies), and semantic meta-data (instance data). It allows queries that arbitrarily mix full-text, structural, linguistic and semantic queries and that can scale to gigabytes of text. A typical semantic annotation project deals with large quantities of data of different kinds. Mímir provides a framework for implementing indexing and search functionality across all these data type."
- MG4J (http://mg4j.dsi.unimi.it/)
  - "MG4J (*Managing Gigabytes* for Java) is a free full-text search engine for large document collections written in Java. MG4J is a highly customisable, high-performance, full-fledged search engine providing state-of-the-art features (such as BM25/BM25F scoring) and new research algorithms. "
- Lemur/Indri (http://www.lemurproject.org/indri.php)
  - "The Lemur Project develops search engines, browser toolbars, text analysis tools, and data resources that support research and development of information retrieval and text mining software. The project is best known for its Indri search engine, Lemur Toolbar, and ClueWeb09 dataset."
- Terrier (http://www.terrier.org/)
  - "Terrier is a highly flexible, efficient, and effective open source search engine, readily deployable on large-scale collections of documents. Terrier implements state-of-the-art indexing and retrieval functionalities, and provides an ideal platform for the rapid development and evaluation of large-scale retrieval applications. Terrier is open source, and is a comprehensive, flexible and transparent platform for research and experimentation in text retrieval."

Regarding the state of the art, these are some of the features required from a good search engine for biomedical information and how many of the potential search engines provide them:

Main features covered by existing system / project	Ranked result lists	Wildcards in queries	Fields	Keyword highlighting in results
Lucene/Solr	+	+	+	+
MIMIR	-	+	+	+
MG4J	+	+	+	-
Lemur/Indri	+	+	+	-
Terrier	+	-	+	-

Table 3. Main features provided by the IR systems

The technical gaps that the Khresmoi architecture will have to face include:

Important feature that should be addressed by the Khresmoi system	Technical requirement	Impact on the architecture	Risks/difficulties	Importance for the Khresmoi system
Combining search results	Merging ranked results	How to merge and how much data is needed for the merge (metadata, content data?)	Incompatible ranking paradigms. Network is a bottleneck to many approaches.	High
Result presentation	Highlighting and explaining why this was found; selecting snippets and creating a summarized result	Balance between providing too much or too little information to the users	No component is single-handedly responsible	High
Multiparadigm search	Searching text, ontologies, images, annotations, metadata	How to select which components are needed for the task at hand	Different content types and search tasks require adapted search technologies, otherwise the result quality is low	High
User interface	Modularity, customizability, flexibility	How to adapt user interface to user, task and available information while not confusing the user?	Difficulties in defining a simple and self-explanatory interface that works for many different use cases. Existing search engines do not provide much support.	Very high
Search profiles and history	Store queries selected by user	Where to store user profiles?	Security, user accounts necessary	Medium (could allow many additional features like alerting)

Table 4. Challenges and technical gaps in terms of Information Retrieval

6.2.2 Architecture for Semantic Web systems used as a knowledge base server

Semantic Web and Linked Data are promising technologies that enable the knowledge base publishing on the web. They define a set of standards and language, which makes possible the exchange of information by computers in a way that preserves its semantics, relationships and context.

There are several established services demonstrating very good scaling and able to integrate information from the biomedical domain like Linked Life Data (LLD) [1], Bio2RDF [2], HCLS KB [3] that apply the RDF technology as "semantic glue". From the enterprise point of view the semantic data integration technology is regarded as way to reduce the complexity of combining data from multiple sources and resolving classical integration problems related to the information accessibility

and distribution. All the three listed services LLD, Bio2RDF and HCLS KB demonstrate similarities with respect to the underlying data model (RDF) and the supported interfaces (SPARQL endpoint).

Import feature that should be addressed by the Khresmoi system	**Technical requirement**	**Impact on the architecture**	**Risks/difficulties**	**Importance for the Khresmoi system**
RDF persistence	Efficient process (load, modify, delete) RDF information	Global impact over the system scalability and capability to operate with information	Face very highly dynamic data that is combined with inference	Highly critical which has global impact over the global system scalability
SPARQL query support	Execute and optimize complex SPARQL queries	Use RDF/SPARQL and core data model for information exchange	Some of the components may require more expressive queries than SPARQL (i.e. image search)	Highly critical which has global impact over the global system scalability and supported functionality
Formal reasoning	Compute the inference closure for formal ontologies	Requires tight-coupling between the information store and the reasoning	Utilize very expressive ontology languages with big scales of data	Important for overall system functionality and capabilities to use semantics

Table 5. Challenges and technical gaps in terms of SW technologies

6.2.3 Architecture for Multimedia search engine

Multimedia retrieval can be really effective for mining biomedical information. A large amount of 2D, 3D and 4D images are daily produced in hospitals and annotations are often missing or incorrect. Information of the images' content can be exploited to provide quick and straightforward search in the electronic health records and the literature.

In order to define the specifications for an efficient multimedia search engine that corresponds to real-life needs, a list of the most important existing multimedia search engines was made:

- IRMA (Image Retrieval in Medical Applications)
  - IRMA[2] is a cooperative project that aims to develop and implement high-level methods for content-based image retrieval with prototypical application to medico-diagnostic tasks on a radiologic image archive.
- GIFT (GNU Image Finding Tool)
  - GIFT[3] is a content-based image retrieval system. That enables the user to do query by example) on images, giving the opportunity to improve query results by relevance feedback. For processing queries the program relies entirely on the visual content of the images, freeing the user from the need to annotate images before querying the collection.
- FIRE (Flexible Image Retrieval Engine)

[2] http://ganymed.imib.rwth-aachen.de/irma/index_en.php

[3] http://www.gnu.org/software/gift/

- FIRE[4] is an image retrieval system designed for research in this area. It is focused on evaluating various image descriptors, not on indices for efficient search.

- LIRE (Lucene Image REtrieval)
  - The LIRE[5] library provides a simple way to retrieve images and photos based on their color and texture characteristics. LIRE creates a Lucene index of image features for content-based image retrieval. Furthermore, simple methods for searching the index and result browsing are provided by LIRE.
- Theseus Medico
  - The Theseus MEDICO[6] project has as one goal to improve the quality of medical diagnoses with the help of imaging procedures. Semantic technologies are planned to help physicians recognize abnormalities in medical images and improve their diagnoses and therapeutic decisions by analyzing image databases and specialized literature. However this project is still in development and so the comparison with other systems is difficult.
- Other systems for medical image retrieval exist and are often used by clinicians such as Goldminer[7] or Yottalook. These systems are based on needs of radiologists to search for images via keywords and then allow filtering by modality or age groups. These systems do not currently allow for a full visual similarity search.

After taking into account the services provided by the existing systems (Table 6) the main features that should be included in the Khresmoi image retrieval engine are identified. Among others, highly significant are the use of state-of-the-art visual features, support of using a set of images as query or query by region of interest (ROI) and support for user query refinement such as relevance feedback.

Main features covered by existing system / project	**Variety of Visual Features**	**Variety of Querying possibilities**	**Easy to Integrate**	**User interaction and feedback**
IRMA	+	+ (query by ROI)	-	+ (several different models and multi step)
GIFT	-	+ (set of images as query)	+ (use of MRML messages to interface)	+
FIRE	+ (use of state-of the-art features)	+ (query by ROI)	-	+
LIRE	+	-	+ (Use of Lucene search server)	-

Table 6. Main features provided by the selected Multimedia search engines that allow for visual search.

[4] http://thomas.deselaers.de/fire/

[5] http://www.semanticmetadata.net/lire/

[6] http://theseus-programm.de/en-us/theseus-application-scenarios/medico/default.aspx

[7] http://goldminer.arrs.org/

New important features, such as 3D/4D image analysis, semantic retrieval using ontologies and multimodal fusion search are only very little covered by the existing multimedia search engines and should be addressed by the Khresmoi system. They are presented in Table 7.

Important feature that should be addressed by the Khresmoi system	**Technical requirement**	**Impact on the architecture**	**Risks/difficulties**	**Importance for the Khresmoi system**
3D, 4D analysis and retrieval	3D, 4D features, Simple interface for 3D, 4D visualisation	Low Impact	Large amount of data, little work has been done so far on 4D analysis	High (Radiologists use 3D, 4D images on a daily basis)
Very large image database management	Efficient indexation – storage -	High Impact (Scalable architecture, efficient communication between components)	Performance/ efficiency trade-off	High (Overwhelming amount of medical images in Hospitals)
Use of semantics for visual retrieval	Mapping of anatomic location/ pathology to medical terminology (e.g. MeSH)	Medium impact (Semantic Knowledge base)	Mapping of images onto ontologies can be difficult due to the large variety of visual content; specific parts need to be developed	Medium (Searching through image content using ontologies is a realistic scenario from the user perspective and would make the system better usable.)
Fusion of text and content-based retrieval results	Appropriate fusion rules depending on the type of query	Little impact (Necessity of fusion component but all basic components exist)	Most appropriate fusion algorithm to be decided	Medium (The fusion of modalities has proven to improve retrieval results)

Table 7. Challenges and technical gaps in terms of multimedia information retrieval.

6.2.4 Architecture for Platform for Natural Language Processing

In looking at the state of the art for NLP platforms for biomedical text processing, we are not concerned with the boundaries of deep NLP techniques over a small number of documents. Rather, we are concerned with processing large numbers of documents, often with relatively shallow techniques, and integrating the resultant extracted data within the rich framework of life-science linked data and ontologies. In short, we require support for distributed semantic annotation i.e. large-scale information extraction against concept systems and ontologies.

Biomedical text processing is a moving target at the forefront of NLP, with the regular introduction of new techniques. Our platform must therefore remain flexible, and not be tied to a single processing paradigm. It must support and provide tools for the full range of NLP techniques, from lexico-syntactic techniques through heavily hand engineered grammars, to supervised and unsupervised ML techniques.

When working at scale, parts of the text processing life cycle and process that could previously be handled in an ad-hoc fashion can no longer be supported in this way. We therefore require full life-cycle support, from manual annotation of example documents, through quality assurance, integration of QA results into updated systems, and finally delivery of annotations integrated into a semantic framework.

Regarding the state of the art, the following table lists some of the features required from a good NLP platform for biomedical information and how some of the current NLP offerings provide them. The table includes only a few examples of the more popular open source or freely available platforms. There are many more with limited reach. Non-open source platforms have not been considered.

Main features covered by existing system / project	**Pluggable architecture, allowing composition of a wide range of processing resources into pipelines**	**Multilingual support and availability of multilingual processing resources**	**Support for manual annotation**	**Evaluation and quality assurance functionality**
GATE	+	+	+	+
UIMA (includes several UIMA promoters and consortia such as OpenHealth)	+	In theory, however limited availability	Limited (external tools often used)	(some QA available via U-Compare system)
OpenNLP	+	Some non-English language models available	-	-
OpenPipeline	+	-	-	-
Lingpipe	+	Limited: Chinese models available; in theory could be retrained for other languages	-	-
NLTK	+	Supported in theory, but not widely found in practice	-	+

Table 8. Main features provided by the NLP platforms

The technical gaps that the Khresmoi architecture will have to face include:

Important feature that should be addressed by the Khresmoi system	Technical requirement	Impact on the architecture	Risks/difficulties	Importance for the Khresmoi system
Integration of manual and automated annotation	Requirement to iteratively apply manual corrections to automatic annotations, and to feedback corrections for improvement of annotation pipelines	Low impact on the end user system, impact will be at batch level processing of data	The greatest risks stem from human factors – organisation of annotators and the annotation process, and the use of technology to avert these risks	Medium
Annotation of very large corpora	Parallelisation and distribution of annotation platforms, most likely on cloud platforms	High impact	Relatively new and fluid technologies	High
Integration of annotation and linked data	The requirement within Khresmoi is both to annotate against linked data, and also to use linked data to assist with annotation	High impact	Although high impact, risk is probably low – failure does not jeopardise the Khresmoi platform, but would impact performance	High
Allow search across annotations to be constrained by linked data semantics	Support for indexing across both annotation and linked data	Low impact – although important, this requirement does not raise architectural concerns	Integration of multiple search technologies runs the risk of tuning to one search paradigm at the expense of others	High

Table 9. Challenges and technical gaps in terms of NLP technologies

6.2.5 Main requirements coming from the state of the art

Main	Description	Importance	Risks	Step that will

requirements				**address it**
End User adaptation	Searching text, ontologies, images, annotations, metadata	High	No because addressed by semantic annotation (RDF)	Software system
Data Integration	The requirement within Khresmoi is both to annotate against linked data, and also to use linked data to assist with annotation	High	No because addressed by Talend	Software system
Architecture	Modularity, customizability, flexibility, scalability	High	No because addressed by SCA	Software system
Persistence	Efficient process (load, modify, delete) RDF information	High	No because addressed by distributed repositories and SPARQL endpoints (Owlim)	Cloud infrastructure
Scalability / distributed deployment		High	Yes, Cloud infrastructure should still be selected but different solutions already identified.	Cloud infrastructure
Parallelisation / Distribution	Parallelisation and distribution of annotation platforms, most likely on cloud platforms	High	No, partially addressed by the Gate platform.	Cloud infrastructure

Table 10. Important requirements for the Khresmoi architecture

In Khresmoi, we have many different requirements. Therefore, to obtain a homogeneous system, it is very important to implement an early prototype that will be:

- presented to the end users to stimulate the feedback to enrich the functional requirements, and
- tested and evaluated to adjust the technical requirements and to refine the architecture design.

Thus, we need the most open / flexible technologies:

- Not intrusive in terms of developments
- Not restrictive in terms of component integration
- Not restrictive in terms for the implementation in the Cloud infrastructure
- Not restrictive in terms of persistence
- Providing an easy deployment of the prototype (locally, distributed and virtualized system)

For this, the pragmatic approach seems to be the best option for the system design and implementation. All challenges are identified at the early stage of the project but technical challenges will be introduced progressively in the development process. Thus, little by little, more complex functionalities will be added in the early prototype.

6.3 Existing technologies for integration aspects

In the previous sections, the main requirements found from the analysis of existing systems could be solved with specific technologies. The objective is to identify those technologies and compare with the requirements previously identified to select the best approach for the architecture (abstract level).

6.3.1 Different approaches for the system integration

The Khresmoi project is composed of several components which are developed by different organizations (i.e. Mimir by USFD and OntoText, Owlim by OntoText, GIFT by HES-SO, etc.). A scalable and efficient integration platform is required in order to build a complete system. There are several architecture options available for enterprise integration. This section introduces the different architecture options. Then, based on the different characteristics of the different options, their suitability for the Khresmoi system is deduced.

1- Enterprise Application Integration

The first possible approach for integration comes from the business context. To work with a large business, an enterprise needs to build diverse applications that include partners systems and that are able to communicate with internal systems or with each other to achieve a business objective. This kind of platform, called Enterprise Application Integration (EAI), should provide a way to connect software components independently of the technologies or geographical location. For this, the EAI provides mechanisms to take care of the message acceptance, transformation, translation, routing, message delivery and business process acceptance. As described by [4], two different kinds of EAI architectures are available.

a. Broker based approach

The broker based approach is fully centralized, which means that all applications send and receive data through a server or engine. Specialized adapters receive the application data and interpret it before sending it to the central engine. Then, the engine transforms and routes the data to one or more target applications according to the different rules of routing or transformation defined within the engine [5].

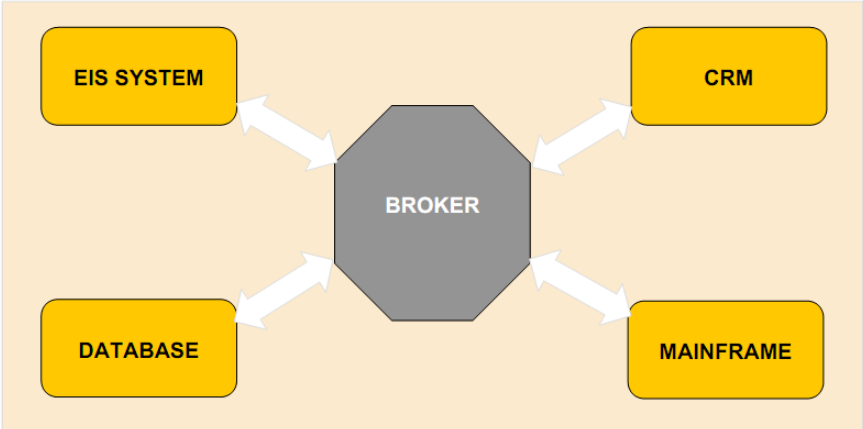


Figure 1. Broker based architecture for EAI

Even if the architecture is very open and permits the integration of many different applications, the risks of this very centralized architecture are all focused on the central engine. One failure on it could imply many perturbations in the whole system.

b. Bus based approach

The second possible architecture of EAI is based on integration-bus topology, which means that all nodes are linked in a series along a common communication backbone. Data or messages that are sent between interconnected applications travel along the bus to the adapter, which handles the data transformation, translation, and subsequent routing to the receiving node. The data sent by the source applications are directly published to the bus. Then, the source or sending application, also called "publisher" or "producer" generates an event in the system. The event is intercepted by the application expecting this kind of message and will receive the corresponding message. The receiving application is called "subscriber" or "consumer". Thus, this kind of EAI architecture refers to a "publisher/subscriber" architecture [4].

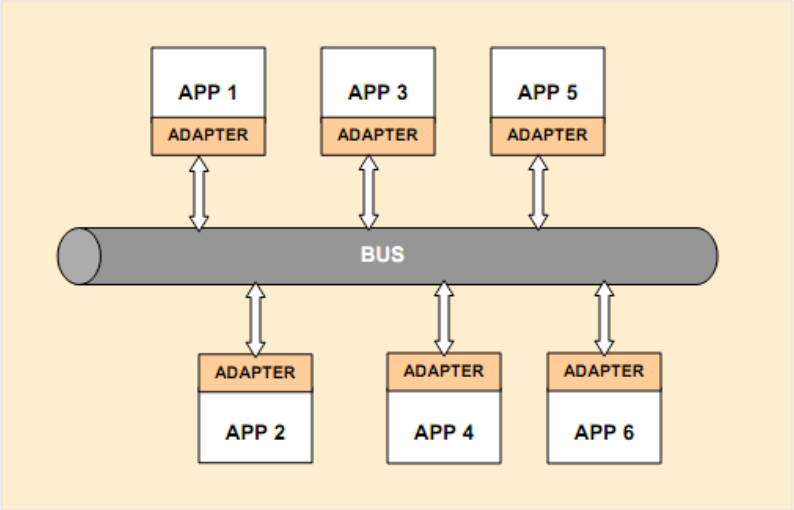


Figure 2. Bus based architecture for EAI

This kind of architecture is interesting because if one point of the system fails it does not disturb the whole system. Moreover, it permits to connect several applications through the "publish/subscribe" mechanism. This mechanism does not imply any restrictions about application numbers and can significantly increase the scalability of the system.

2- SOA

A Service Oriented Architecture (SOA) represents a logical way of structuring a software system into a set of loosely coupled components whose interface can be described, published and invoked over a network. As a central component of the architecture, the service broker acts as an intermediate component between the other interacting components (service provider and service consumer). The components are deployed as services with standardised interfaces, independent of any specific platform or implementation technology (thus separating implementation aspects from interaction specifications), and carry out together a high-level function or business process.

The Organisation for the Advancement of Structured Information Standards (OASIS) specifies a service in the scope of SOA as "a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description." [6]. But SOA itself is merely an architectural approach and not bound to any specific technology; there are a lot of different ways how to integrate and implement based on SOA principles. Especially, two mainly representative approaches of SOA are now presented.

a. Enterprise Service Bus

One of the most important approaches to build some SOA-based infrastructures is called Enterprise Service Bus (ESB). The ESB technology helps simplifying the integration of several different business cases/units and bridging heterogeneous platforms and environments. As defined by Gartner, ESB is "a new architecture that exploits Web services, messaging middleware, intelligent routing, and transformation. ESBs act as a lightweight, ubiquitous integration backbone through which software services and application components flow."

It combines the benefits of standards based Web Services with traditional features of Enterprise Application Integration (EAI) products, while avoiding the great complexity, inflexibility and hence project cost associated with those products. Then, an ESB can be regarded as an intermediary layer that enables the deployment, management and interoperation of independent components or applications. It supports service, message, and event-based interactions using synchronous or asynchronous communications, which can be brought in relation to interaction paradigms, hence facilitating interactions between one or many actors (one-to-one or many-to-many). The concept is based on separated components via standard-based adapters and interfaces. For example, it can be based on Web services standards such as SOAP, WS-RM, WS-Addressing, etc. or not like HTTP, JMS, .NET, etc.

One main benefit of an ESB infrastructure is that it allows decisions about the flow of data and the flow of control to move out of the endpoint applications and into the broker. This can be routed accordingly to business needs based on rules or other configurations, keeping it modular and adaptable.

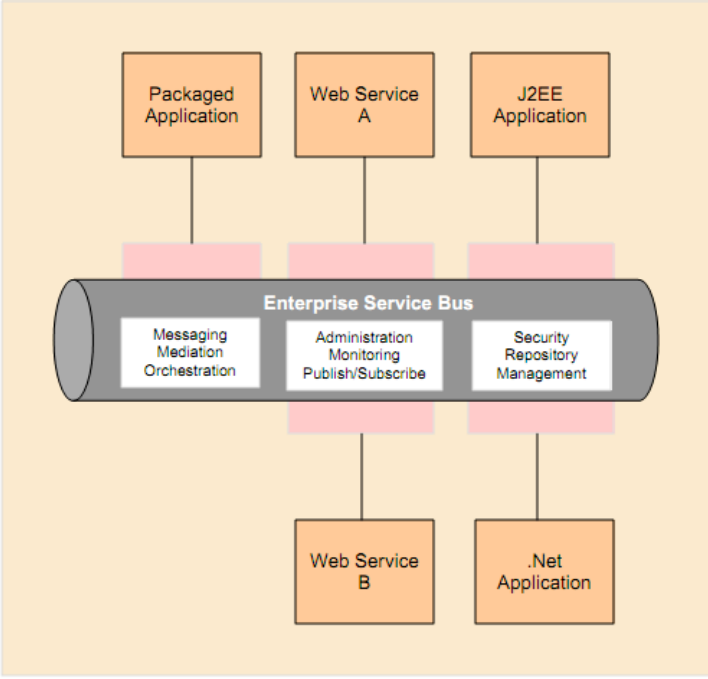


Figure 3. Example of the ESB architecture

b. Service Component Architecture (SCA)

A second approach that emerges as a relevant SOA solution is called Service Component Architecture (SCA). Initiated by a group of vendors (including BEA, IBM, Oracle, SAP, etc.) and now continued by the OASIS group, the SCA formalization contains a set of specifications to describe how the different parts of an application work together. In the scope of SCA the distinct parts of an application are called components and the combination of several components into an application forms a so-called composite.

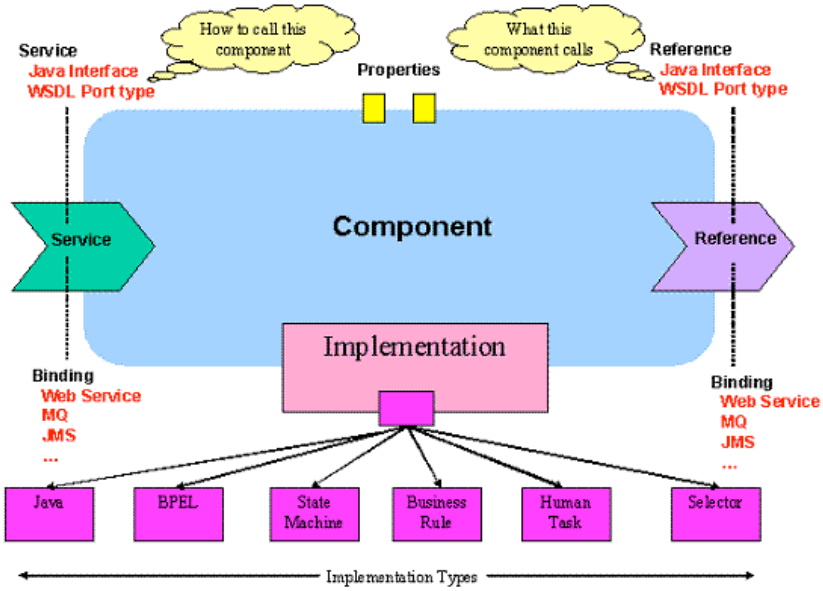


Figure 4. Representation of the component as defined by SCA

The very modular architecture allows to combine the atomic (business) logic of single components reusable to sets as composites which may also comprise external/legacy systems in the processing (e.g. a workflow definition which is concluded of three components and one existing company data provider system).

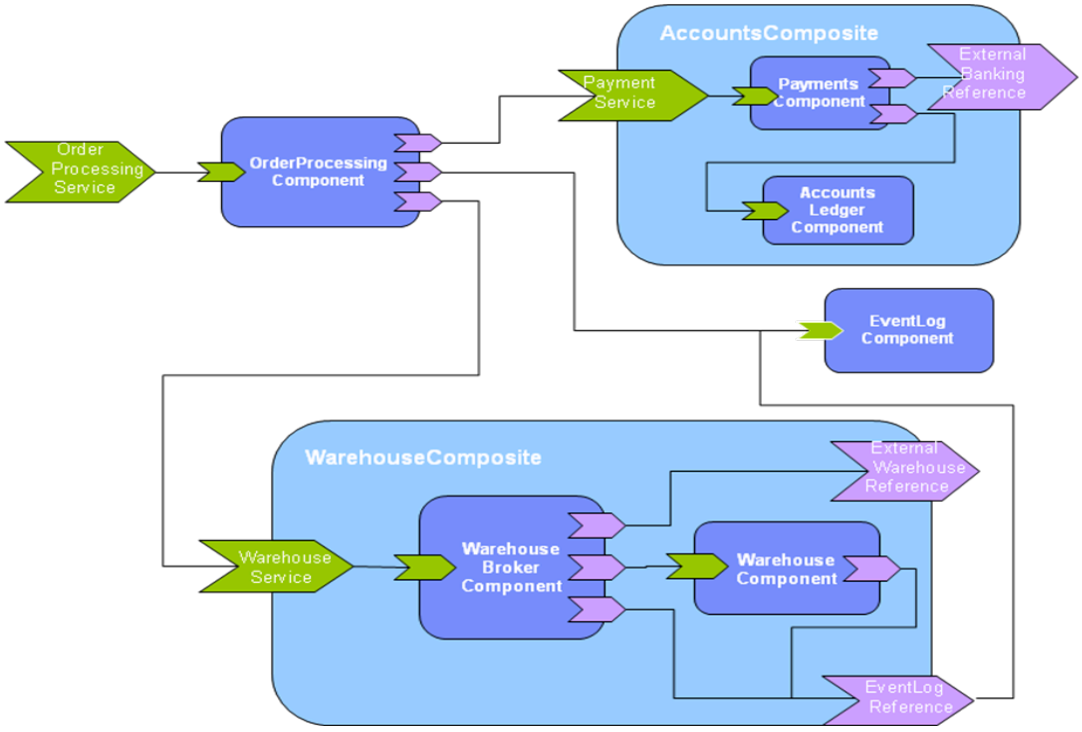


Figure 5. An example of SCA architecture based on composites specification

Although the SCA approach itself is technology-independent it leverages several open standards, such as Web Services [7], can be technically implemented using different technologies and may be distributed over different networked machines as it follows the principle of loosely coupled services.

The further communication/interfacing is a mere part if technologies e.g. like the mentioned Web-Services (e.g. REST), JMS or SDO.

c. OSGi Framework

The OSGi Framework by the OSGi Alliance (OSGi) is a programming framework, which is based on components and bundles integrated into a managed environment. OSGi itself is modular and scalable architecture approach of a service delivery platform, which is often referenced specifically to Java, as the initial implementation is based on the Java VM but not limited in its principles to it. OSGi mainly consists of Modules, Life-cycle Management, Service Registry, Bundles and an Execution Environment. While the EE may be anything (i.e. Java VM) the Service Registry registers all components while the Modules define the loading policies of the components to stay in relation to the Life-cycle Management which is main uniqueness factor of OSGi: it allows a change/add/deletion of components during runtime with an easy integration system. The services layer then serves as a discovery and bind part of the architecture. The bundles are then further the complete set of components, registries, managers and policies defined. OSGi is an increasingly accepted approach to build especially plug-in based systems (e.g. Eclipse) because of its Life-cycle features and loose component architecture. A common complaint is that only the OSGi Alliance defines the framework and therefore it only serves the needs of the members.

3- Relevant results for Khresmoi

Many other approaches for integration are available in the literature. To ensure the comparison of these different technologies, the main principles will be presented. At the end of the section, a table to summarise the main features of those approaches and to compare them with respect to the Khresmoi needs will be proposed.

Approach	**Method**	**Advantages**	**Disadvantages**
Enterprise Application Integration	Broker based approach	The transformation and routing rules don't have to be hard-coded within the applications that decrease complexity.	High communication cost because of the broker.
		Tightly coupled	Single point of failure
		Less installation effort compared to solutions with bus architecture.	High costs
		Easy to maintain and administrate because of central hub.	
		High scalability if federated architecture is used otherwise limited by the hardware of box used to host Hub	Mostly standard based but may use proprietary internal formats.
	Bus based approach	No single point of failure	Development of an adapter for each application to be integrated
		Moderate effort of installation	Administration may be complex depending upon the integrated systems
			High costs

		Loosely coupled	Not suitable for applications needing to share information with an unpredictable number of entities residing at different locations
		Highly scalable	Mostly standard based but may use proprietary internal formats.
SOA	ESB	Based on SOA standards	Most of the ESB products do not provide service orchestration and mediation (transformation and routing)
		Light-weight compared to the message brokers or hubs	Early to implement "pure" ESBs that are built totally on SOA standards, since most of the applications are far from being web services enabled
		Moderate effort of installation	Administration may be complex depending upon the integrated systems
		Low cost because it does not use proprietary formats to enhance performance. Also it does not provide all the services usually provided by proprietary product suits.	
		Highly scalable	
	SCA	Lightweight	No supervision
		Very flexible, easy mechanism to compose service	
		Less installation effort compared to solutions with ESB architecture.	
		Easy administration	
		Low costs	
	OSGi framework	The same as SCA but limited as a Java implementation.	Too restrictive because only available with the Java implementation.

Table 11. Comparison of technologies for system integration

6.3.2 Methodology to build SOA system

As we adopted an SOA approach for the system integration, a method to identify the main services and integrate them as a full system should be followed.

1- SOA methodology and system specification

SOA is a very abstract approach to specify how to organise a complex system with many different software components, but it does not prescribe guidelines to identify and specify the different services of the system. [8] proposed a method called Service-oriented modelling and architecture (SOMA) that provides some methodological principles to organise the SOA systems.

At the beginning, the SOMA method proposes a logical structure. For this, it differentiates nine layers of the SOA system as shown in Figure 6.

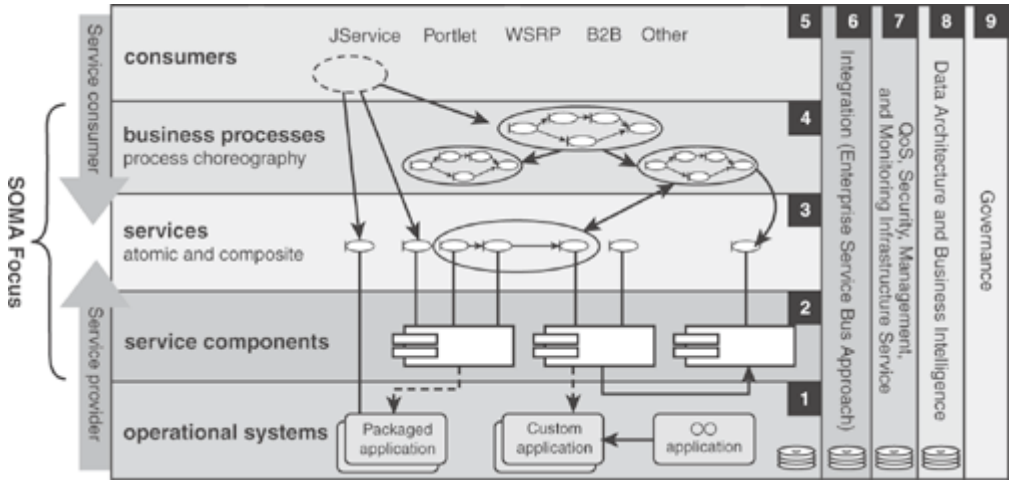


Figure 6. SOA reference architecture

- Layer 1: is the Operational System layer that contains all the different existing systems or software components that should be integrated in the new system.
- Layer 2: is the Enterprise Components layer, which is the intermediate layer used by the Enterprise to deploy the existing systems previously described.
- Layer 3: is the Services layer that is composed of all the main services or atomic software components.
- Layer 4: is the Business Process Composition or Choreography layer. This layer corresponds to the service compositions that are required by the business processes.
- Layer 5: is the Presentation layer. This is the User Interface layer that will permit the end-users to realize the different business processes.
- Layer 6: is the Integration layer that provides a homogenous framework to enable the different layer communications.

Finally, the three other layers are complementary layers that are important to take into account, but not as important as the previous ones. For example, the quality or the security will be assigned to a specific layer and should contain some services that will address it in the system.

All the different layers are used to design the Khresmoi architecture. Of course, the specific requirements identified will imply many adaptations of this very abstract structure.

As already said in a previous section (Cf. section 6.1), the specification of the SOA systems depends on the user requirements and the existing components. The idea is to find an intermediate solution to satisfy those two important constraints. For this, the method SOMA proposes to combine a top-down approach (focused on business processes) and a bottom-up approach (focused on the analysis of the existing components). The method SOMA is composed of three different steps:

- Identification of the services,
- Specification, and
- Realization.

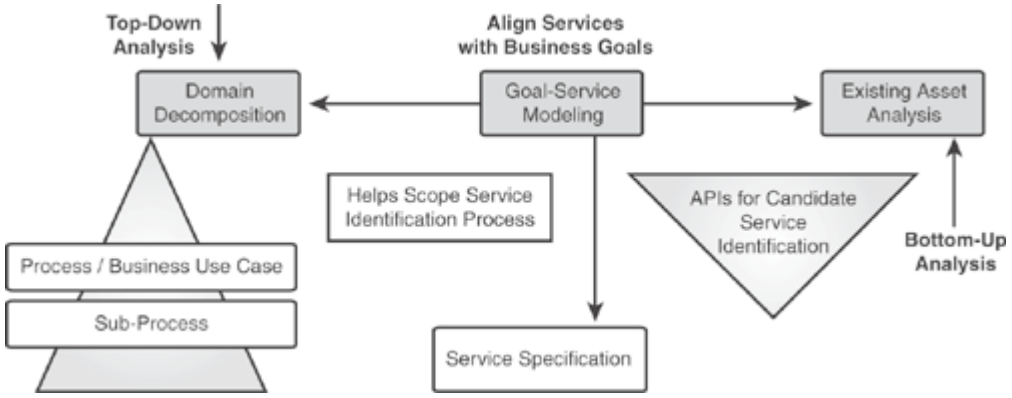


Figure 7. Process for the service specification

a- Identification

The first approach should permit to identify the main functionalities that are derived from the business processes, contrary to the second approach that should identify the relevant functionalities already available in the existing components. Then, based on both results, the new functionalities that should be implemented or the ones that should be extended can be identified.

b- Specification

When all the service are identified, those that already exist but should be adapted to the Khresmoi system, and the new ones that should be implemented, their interface and references should be specified.

Whatever the implementation language used by the service, it should declare the standard interface such as SOAP, REST, etc. (cf. in next section for the possible formalisms). This interface will enable the interoperability with the other services that are deployed in the system.

Furthermore, to work properly, the service can require data from other components. This dependency is called reference and should be also defined. This part contains the protocol and the bindings to use the required service.

In our case, the interface and the reference declaration will be specified with the SCA formalism. Effectively, SCA provides an XML mechanism to declare the service definition.

c- Realization

Based on the service specification, the realisation of the services consists in the development and the deployment. In fact, thanks to the service specification, the service will be implemented to be deployed through the SCA runtime. Then, the services and the composition of the services will be tested within a homogeneous framework.

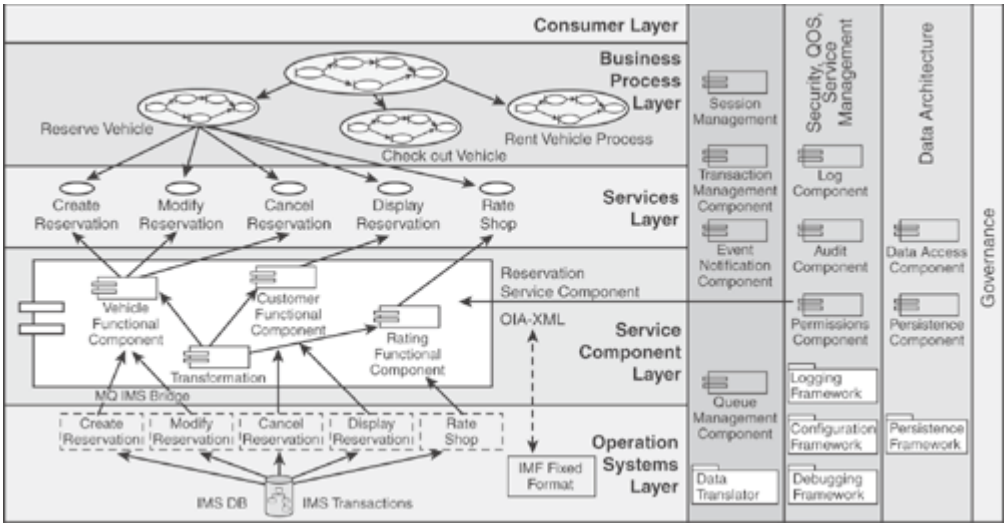


Figure 8. Example of SOA application

Finally, Figure 8 shows an example of the SOA systems with the different services defined in each layer, and their respective interactions. After the successive iterations of the architecture design with always the bottom-up and the top-down analysis, the Khresmoi system would be also defined like this. As the user requirements activities are still taking place, some design iterations are still missing to obtain this full picture of Khresmoi.

2- SCA technologies for service integration

As described before SCA (Service Component Architecture) is a component for SOA. It is hosted by the Open SOA consortium (OSOA: http://www.osoa.org). It is standardised by OASIS (OASIS: http://www.oasis-opencsa.org).

Different providers offer their own infrastructure to implement SOA solutions based on the SCA formalism. Some are Open Source (i.e. Apache Tuscany, Fabric3, FraSCAti, Mule Newton, etc.) and others are vendors (i.e. IBM WebSphere FP for SOA, TIBCO ActiveMatrix, Covansys SCA framework, etc.).

Now the two most advanced solutions based on open source licensing will be presented and compared.

Firstly, FraSCAti is a platform provided by the OW2 consortium (http://www.ow2.org/). The platform proposes a component framework providing an SCA runtime support. This means that the components could be formalised with SCA formalism, they could combined as composites and finally they could be deployed through the FraSCAti platform. In addition to the dynamic deployment for distributed systems, the complementary tool Fractal (http://fractal.ow2.org/) permits the adaption and reconfiguration of the runtime.

Similarly to FraSCAti, Apache Tuscany is an open source implementation of the SCA specifications. It is developed by the Apache Software Foundation (ASF) with the aim of providing a simpler way of creating applications using a service-oriented approach. Tuscany provides a lightweight SCA runtime that can be used both as a stand-alone solution and as an embedded solution within an application server (e.g. Tomcat, Jetty) to deploy easily the services.

Furthermore it supports components implemented in different programming (e.g. Java, Python) and scripting languages (like JavaScript, Groovy) as well as using different frameworks such as OSGi. For orchestrating services/components together a wide variety of technologies may be employed: bindings like Web Services, JMS, JSON-RPC or Enterprise Java Beans (EJB). Tuscany is one of the specification's most advanced open source implementations of the SCA specification. This solution is

enough mature and stable to initiate the integration work in the Khresmoi project. If further requirements will be identified, and as the SCA formalism is not intrusive for developments, it would be possible to use more robust technologies, such as ESB for example.

The following table presents the main characteristics of the most important implementation frameworks dedicated to SCA. This comparison would permit us to select the SOA framework that will be used to develop the Khresmoi system.

Platform characteristics	**FraSCAti**	**Apache Tuscany**
SCA features	++	+++
Possible bindings	+	+++
Ecosystem	+	+++
Runtime platform	+++	+++
Additional tools	++	++
XML SCA editor	+++	+++

Table 12. Comparisons between the best SCA platforms

Finally, like Apache Tuscany, the FraSCAti platform is a reflective SCA platform based on lightweight, efficient, predictable and scalable principles. But regarding the comparison results, Apache Tuscany seems to be more mature. Also, it has a better ecosystem (document, examples, developer community, etc.).

3- Potential technologies to extend the SCA integration with ESB

Beyond the SCA solutions, they could be replaced or completed with an Enterprise Service Bus (ESB). So, in the context of Khresmoi, the ESB is not required for the early prototype, but could be required later if more dynamic composition of services is required. Thus, at this stage, the ESB's compliant with the previous SCA platform are introduced.

Petals ESB is the open source Enterprise Service bus proposed by the OW2 consortium. This is the service oriented infrastructure for SOA solutions. Petals ESB is suitable for deployment in real-time, rapidly changing environments that are typically common in large scale enterprise SOA solutions. Many additional tools are also provided to offer a powerful service oriented infrastructure to build a specific SOA system. For example, Petals View for the monitoring and Petals Master for the SOA governance are useful complementary tools.

The whole Petals environment provides efficient tools to design the system based on loosely coupled services, and to ensure the runtime scalability to deploy a fully distributed architecture. In addition, Petals already takes into account the SCA implementation for the system integration and deployment.

Apache Service Mix is an open-source and standard based ESB, based on the Java Business Interface (JBI) specification and developed under the umbrella of the Apache Software Foundation, with the aim of facilitating the development and deployment of composite enterprise applications. ServiceMix supports the interoperation of services according to the SOA principles and at the same time allows services to operate in an event-driven way. This means that services connected to the ServiceMix bus are completely decoupled and that they listen to the bus for service requests.

Furthermore, ServiceMix supports interactions based on events that can occur both internally or externally to the ESB. The fact that ServiceMix is based on a standardised specification allows flexibility, reuse and protection from technology changes (i.e. it protects investment). For instance, any component that conforms to the JBI standard can not only inter-operate with other components registered on the bus but can also easily be replaced with alternate components that provide the same services. An ESB like Apache Service Mix provides a very powerful tool for building an SOA. However, ESB's are usually more focused on complex business workflows with a large number of routing rules and connected legacy applications. Thus it has to be analysed if an ESB may be applicable to the Khresmoi system because of its routing and legacy addition principles. Probably, it can be interesting later to implement the Batch system of the search engine.

The advantage of the ESB is that it is fully compliant with the SCA formalisation. Thus, it can replace or complete/enforce the Khresmoi architecture if new requirements are required. Probably, it will be required when the requirements for the Batch system will be studied deeply.

4- Web Services

Web Services are a widely used technology that enables realising different architectural communication principles, especially of SOA. They represent self-contained modular business applications that have open, Internet-oriented, standards-based interfaces. The W3C consortium defines a web service as "applications identified by a URI, whose interfaces and bindings are capable of being defined, described and discovered as XML artefacts. A Web service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols" (W3C). The Web Service concept relies on a widely available and strongly supported set of standards and protocols. Common protocols are the REST and SOAP protocol.

REST (Representational State Transfer) itself is an architectural idea for distributed computing solutions. Strictly seen the Internet today is a RESTful system [9]. But nowadays it is mainly identified and used as a Web-Service paradigm for distributed communication over HTTP. REST is based on an easy three-part system defining the web-service source, the data (in any form but often represented by JSON) and the web-service HTTP operation (CRUD operations). These simple actions make it simple to use and it is preferred by large companies. This is also the main weakness as it is simply a retrieve, alter and delete process of information. SOAP (until version 1.1 referenced as Simple Object Access Protocol) on the other hand is a classic exchange protocol mainly used for Web-Service implementations [10]. Utilizing existing HTTP(S) networks as transport layer for its communication the XML-based SOAP protocol utilizes "envelopes" to send its data as well as communication initiation. It is supported by a wide set of tools and accepted for using the XML standard. But this is also the main drawback as the communication with XML based on SOAP specifications need a large amount of information around the actual communication data.

In Khresmoi, many components provided by partners are already integrated or used by other systems. In that case, the components already provide some REST methods or SOAP services to interact with external components.

In the other case, components should specify a Web Services interface to offer software interoperability. Finally, after stabilization of the components integration, a standardization of message exchange between Khresmoi components would be envisaged.

5- Orchestration / negotiation for the Search system and Pipeline for the Batch system

In some systems, the description of the logic processes for Use Cases is very important because those processes are so important that they should be formalised in the system. The process specification permits to facilitate and enable an efficient orchestration of the different required services. Different standards exist to describe and specify such processes. For example, the Business Process Modelling

Notation (BPMN2) is a standardized graphical notation for drawing business processes in a workflow or the Unified Modelling Language (UML2), a general purpose modelling language that includes a standardized graphical notation, may be used to create an abstract model of a system. In combination with an orchestration supporting architectural platform it is possible to utilize the models as reference input to align the services into one bundled system. Several implementations try to allow such correlation of models and implementations (e.g. Apache ODE or Oracle BPEL Process Manager). However it was demonstrated that this kind of technology is too rigid, especially for an interactive system that is the case of the Khresmoi system (at least for the search system).

In the case of the Batch system that will collect, process, and categorize the huge data stream, the process could be interesting. But, those kinds of processes are already integrated in the components provided for the Khresmoi, for example Gate and Talend are tools that allow developing and specifying some processing pipelines to analyse data or integrate data from processing results and according to semantic models.

GATE Developer allows a user to define a pipeline of natural language processing components, or *processing resources*. Each processing resource creates *annotations* in the text being processed: data associated with offsets in the text, such as the boundaries of a sentence, the part-of-speech of a token, or assigned types from semantic models to named entities. Each processing resource may use annotations created by previous steps as input, and build on these, giving an analysis of increasing complexity. Such a pipeline, whilst created in GATE Developer, may be deployed as a batch process with the GATE Embedded libraries. Hosting such a batch process may be as simple as running a single thread from the command line, or as complex as deploying on Amazon EC2 via GATECloud.net.

The nature of the Khresmoi project presents the problem of dealing with dynamic heterogeneous (text-) data originating from many different sources. The relevant information will have to be extracted from the data sources, transformed into a supported format by the system and ultimately semantically disambiguated before being loaded into the system. This is a data insensitive process, which requires high efficiency not only in terms of good performance and scalability, but also in easy maintenance and traceability of each individual step part of the complete data integration process. Talend Open Studio (TOS) is an open-source solution for performing data integration based on the Eclipse platform that offers off the shelf a very good toolkit and infrastructure for composing data processing tasks known as jobs. TOS uses in the background java code generated using Java Emitter Templates (JET), which enable the easy extension of the environment with custom components that extend the system with support for reading and writing from/to RDF, SPARQL and semantic repositories. Such components were developed in the LarKC project to support the Linked Life Data knowledge base release cycles and used to semantically integrate more than 25 data sources or in total over 5 billion facts.

A key advantage of TOS environment compared with the other data integration or mediation technologies is the outstanding performance and scalability due the code-generated jobs produced as an outcome. At the same time the intuitive TOS graphical user interface provides an easy way to debug the data flows and run them in a batch mode or scheduled sequences.

6- Relevant results about SOA approach for the Khresmoi system

Main SOA technologies	**Description**	**Advantages for the Khresmoi**	**Inconvenient for the Khresmoi system**	**Compatibility / impact on the**

		system		**architecture**
SCA Tuscany	Open source infrastructure	Lightweight, efficient, predictable, scalable	Easy deployment for distributed systems	Compliant with several cloud infrastructure
SCA fraSCAti	Open source infrastructure	Lightweight, efficient, predictable, scalable	Easy deployment for distributed systems	Compliant with several cloud infrastructures
Petals ESB	Open source infrastructure	Scalable	More complex configuration, and administration	Compliant with several cloud infrastructure
Apache Service Mix	Open source infrastructure	Scalable	More complex configuration, and administration	Compliant with several cloud infrastructure

Table 13. Relevant results about SOA technologies

ESB and SCA both have very interesting features regarding the system integration, and they can fit well with the Khresmoi requirements. But at the end, our choice is more oriented towards SCA technology because it is more lightweight and less restrictive than ESB technology.

Thus, SCA Tuscany is the best solution:

- It has the most active community
- Easy deployment
- Non-intrusive for development
- Extensible with SCA if new requirements appear
- Compliant with standards of Cloud infrastructure

It is important to choose a flexible solution to test the integration, monitor the different services, and evaluate the resource requirements to refine the technical requirements. The advantage of this solution is that it can be used as the final technology, and if more requirements appear, the specification and work already done for integration will be reused entirely, for example in an ESB solution.

6.3.3 Cloud and virtualization

The Cloud Computing approach allows the IT management to easily provision computing resources, thus reducing the time required to buy the physical computing resources, install, customize and whatever else is needed before being able to give these resources to the users.
To make this possible the Cloud Computing paradigm takes advantage of a well-known technology such as Virtualization. Virtualization has reduced the complexity of having many computers running at the same time by reducing the size of these machines – most of time turning them more specific (executing only one kind of task) – and executing more than one independent machine in a single physical computer. This has reduced the complexity of some IT management operations but also has made it easier to migrate from one computer to another (and obviously from one physical location to another).

But all these features from the Cloud Computing and Virtualization technologies could mean nothing if we don't take care of its limitations and specific requirements.

In fact, from the point of view of the applications providers, these requirements have some consequences in the way that they must install/deploy their tools into the Khresmoi infrastructure. First of all, we need to understand that having everything installed on virtual machines allows us to reproduce exactly the same machine as many times as we want (as long as we made a virtual machine

snapshot with the initial configurations) but, it is possible that these applications must be configured to interact with other machines in a more dynamic environment. For example, it is possible that some applications require a connection with servers to retrieve data from them, and very often, these server addresses are pre-configured on the applications statically. As one could imagine, this kind of behaviour does not fit well on a cloud environment; although it is possible to keep these use cases working, it requires a lot of extra management and adds complexity to the environment. Instead of this, it would be better for everybody to understand how a cloud computing environment works and how applications configuration could be enhanced to be executed in an environment like this.

A cloud computing environment was envisioned to be more dynamic than older computing environments. Because of this, all the partners must agree on doing their best to update their assets to take advantage of the Cloud. It is not only about a larger number of restrictions but sensible advantages too. Migrating our environment to a Cloud allows us to have a clean and ready to work virtual machine in a few minutes. If we are able to properly set up a virtual machine and we take a snapshot at the appropriate moment, we will be able to reproduce the initial conditions as many times as we want regardless of the previous actions.

Sometimes it is necessary to install some extra tools to execute some concrete processes and, later, we realize that this software was not only unnecessary but harmful too. In these cases we will be able to destroy the affected machines and start a brand new environment with clean machines in just some minutes.

To build an environment like this, our aim is to use open source solutions to take advantage of the community support and improvements. But the first step needed to choose which tools best fit our needs is to gain a wide vision of the project and identify special needs for all the partners involved. With this deep knowledge of each partner's environment and technological requirements we will be able to distinguish between a wide range of tools.

The main components of the future cloud environment for Khresmoi are the hypervisor, the software in charge of executing and creating the virtual machines, and the cloud manager, the software responsible for managing the physical machines and their software stack (including the hypervisor).

Relevant results about the cloud infrastructure for the Khresmoi system
As this is a very initial version of the tools analysis, it could contain some errors as long as they haven't been studied in depth. Once the requirements are collected and processed, we will proceed to study the most appropriate ones from this list. This work will be part of the task T6.3 called "System scaling".

Main technologies	**Description**	**Advantages for the Khresmoi system**	**Inconvenient for the Khresmoi system**	**Compatibility / impact on the architecture**
Xen	Open source hypervisor	It has a free license and doesn't have strong hardware requirements.	It could restrict the project to use certain types of operating systems.	If there is no problem with the operating system restriction it is one of the best options for the virtualization stack
KVM	Open source hypervisor	It has a free license. Properly configured it could allow us to run almost all operating systems.	It has strong hardware requirements	It is not as easy to manage as Xen but it is also a good choice for the virtualization stack.
VMware	Commercial hypervisor	Very easy to build VMs and	The open source license could not fit in	It will impact on the cloud manager

		customize. There is no restrictions on the type of OS	the project and the licenses are very expensive. The existing cloud managers has very limited support for the VMware products	selection. It could increase the infrastructure costs (licenses) but it could all
Citrix XenServer	Commercial hypervisor	It is easy to build VMs (not as easy as with VMware but still).	The commercial license is very expensive and cloud managers started to support this hypervisor too recently	It will strongly impact on the cloud manager selection and will increase the necessary budget for infrastructure
OpenNebula	Open source cloud manager	Very easy to install and manage. It has very light requirements and can run with a wide variety of devices.	It was initially designed to work with Xen or KVM, support to other hypervisors has to be tested	Before studying each partner it seems to fit very well on the architecture
Eucalyptus	Open source cloud manager	It is a stable cloud manager	Its installation and configuration is more complicated that the previous one defined. The requirements are higher	It seems to fit on the initial architecture but it could require more management than OpenNebula
OpenStack	Open source cloud manager	Enterprise-ready cloud manager (it comes from RackSpace).	It has as many requirements as Eucalyptus.	Could fit in the Khresmoi architecture but probably it is not the best option.
Emotive	Open source cloud manager	The storage is distributed.	It seems to be developed a little bit slower than the other ones	It has more restrictions on the hypervisor (at this moment it seems like it is only supporting Xen and KVM) but everything else should work.

Table 14. Relevant results about Cloud technologies

As seen in the first part of the state of the art (section 6.2), the scalability is a very important requirement for the search engine, and in particular for the batch system. It was demonstrated that this feature can be addressed by the SCA approach, especially the runtime provided by the Apache Tuscany. Rajarshi Bhose and Kiran C Nair[8] already tested the deployment of Composites Application in the Cloud infrastructure thanks to the Domain manager provided by Tuscany. This ensures also that SCA approach will totally fit with the next step of the project related to the system scalability (T6.3).

6.4 General requirements and integration decisions

[8] Integrating Composite Applications on the Cloud Using SCA: http://drdobbs.com/web-development/223800269

The Khresmoi system should be take into account several constraints that come from the state of the art (about similar systems and integration approaches), from the existing components, and from the user requirements.

All those aspects will be summarized to introduce the integration decisions and their impact on the architecture.

6.4.1 List of the main components to be integrated

As mentioned before, the Khresmoi system is based on several software tools that are provided by the consortium partners. This is an important constraint that affects the architecture design. In fact, some components will be easily reusable because they are already available with Web Service specification and are deployed on public servers. But other components could be a part of the software that is in a production or under licenses restriction. In this case, solutions to extract and integrate the functionalities in the whole system will be more complex.

In the following table (Table 15), all the components provided by the partners are presented. For each one, the relevant functionalities for Khresmoi and their license restriction are listed. The functionalities presented in this table could be deployed directly in the Khresmoi system, but for the majority of them, they will be extended in the corresponding WP to fit with the user requirements.

Partner	**Tool**	**Functionalities**	**Licence**
USFD	GATE TeamWare	- Manual annotation of example documents - Develop automatic extraction and annotation applications - QA and evaluate automatic extraction and annotation applications - Couple automatic and manual annotation in an iterative improvement cycle - Deliver automatic annotation - Index annotations	AGPL
USFD / ONTO	MIMIR	- Mimir – index annotations and text - Mimir – query indices	AGPL
ONTO	Owlim (exposed as Sesame repository)	- Persist RDF data - Evaluate SPARQL query - Disambiguation (autocomplete)	Owlim-Light: LGPL v2 Owlim-Standard: Free for academic use Owlim-Enterprise: Free for academic use
ONTO	Talend	- Extract-Transform-Load workflows - Parse/Serialize RDF - Add data to Owlim - Query SPARQL endpoint - Annotate text-data using GATE applications - Index annotated text-data using Mimir	GPL v2
HES-SO	GIFT	- Image database indexation - Query by example image retrieval - Support user feedback for query	GPL

		refinement	
	GenTool	- GenTool Service – provide gene category	
UDE	ezDL Daffodil	- Docking, tool-based UI framework for search with perspectives o meta search over several sources with result integration, sorting, filtering, term extraction and different methods for query specification o viewing, collect, organize and export results, terms and past queries - Agent-based backend for ezDL clients o wrapper and query framework (includes transformation of query syntax), integration of results from different sources, caching of result metadata o logging of all user and system actions o user administration for authentifi-cation and personalization - strategic support using CBR	GPLv3 (other licenses on request)
CUNI	Czsem Mining Suite	- TectoMT Analysis for GATE: provides TectoMT linguistic analysis of text for GATE documents. - Mímir Index Feeder: Custom component used to fill Mímir index with analyzed Gate documents, under development. - TectoMT (http://ufal.mff.cuni.cz/tectomt/): Set of NLP tools used for linguistic analysis of text.	Open Source
MUW	MedUniVie: Image Learning and Retrieval	- Derive Internal Models given a set of data - Retrieval System Anatomy - Retrieval System Pathology - Semi-supervised feedback - Batch system o Detect Anatomic Regions - Learning o Abnormality Similarity - Learning o Knowledge Persistence Helper (batch) - Search system o Detect Anatomic Regions - Retrieval o Abnormality Similarity – Retrieval	License registration not yet defined.
HON	Wrapin	- Spelling correction - MeSH Terms Extractor	Proprietary tools

Table 15. List of the existing components provided by the consortium

For each component, all the information has been collected (installation, implementation language, interface, references, etc.). Also, a classification has been realized to distinguish the services that will

be dedicated to the search system or the batch system, or eventually both. In section 8.1, two UML diagrams represent all those components with their available interfaces.

The components that should be deployed to be used in the Search engine need to be lightweight to enable the powerful user interactivity. This aspect is not a problematic issue because we use a very flexible approach as provided by SCA, other one would be too restrictive to enable a coherent service framework.

The components that should produce data and specific indexes do not need a full integration. They only should be configured to populate the index repository regularly according to the processed data. Then, in this case, only the repositories and data access is important (RDF repositories and SPARQL endpoints) in terms of integration.

6.4.2 General scenario and user requirements

The discussions about the possible scenario were initiated very early in the project, just after the component description because user requirements are indispensable to identify the functional requirements and to initiate the architecture design.

Thus, at the beginning a fictive scenario was elaborated to describe a possible session of information retrieval. It permits to identify the main functionalities that the system should provide and the possible workflows. Based on this very general scenario, WP8 and WP9 started to specify more concretely the scenarios for the specific Use Cases targeted in the project. As the user requirements are still in progress (in parallel to the architecture for WP9, and later for WP8), the architecture is fully aligned with the current status of this task but will need some refinement after the first year based on requirements analysis and specification activities within WP8 and WP9.

A summary is provided of both scenarios to introduce the main aspects that have an impact on the Khresmoi architecture.

1.1.1.1 User requirements for the General public (WP8)

- **General scenario**

A user searches for e.g. "diabetees". They are the Spanish grandparents (with poor eye sight) of a French child who has just been diagnosed, and they want to know what it is, and how they can help with treatment, etc.
First, spelling correction and disambiguation of "diabetes" - do they mean Type I or Type II? The system has to present both to them with non-technical descriptions, so that they can choose which they mean.
Then, when they have chosen e.g. "Type I Diabetes", it presents them with:
Basic, practical, trusted and readable information, presented from Spanish language primary source, and some translation of other languages, but also French local information translated into Spanish:

- definitions
- treatments
- epidemiological data, probability etc. - in this case maybe life expectancy
- links to images
- general health information
- practical daily life information
- user groups, health care providers
- new technologies
- science and research

Each section shows short extracts, of a few sentences, illustrating these things, and they can click through to sources. When the user chooses to look further at something by clicking links, feedback is collected and used to improve future search.
Information has to be based on geography - e.g. user groups, but also in some cases treatment etc. ("tick bite" means different things in different parts of the world)

- **Main Functionalities / components**

Based on the general scenario, the most important functionalities in the system have been identified:

- detecting locale / IP address
- spelling correction
- disambiguation
- proposal of information type
- trustability and readability categorisation
- section / sentence extraction corresponding to proposed information types
- information selection based on locality
- translation
- linking to source documents, further information etc.
- user interface and presentation

In the Appendix (Section 11.1), different UML diagrams are proposed (Activity diagram, and Sequences diagram) and the list of the main functionalities (Section 11.3).

1.1.1.2 User requirements for the Radiologists (WP9)

- **General scenario**

Below, some typical use case scenarios for WP9 are described:

- Scenario 1 – Clinician

A Spanish clinician is examining a full body CT of a patient on his iPad during a medical visit. He is concerned about the findings of a particular area (e.g. liver) so he selects the region of interest and searches for similar images and similar cases using the image region as a visual query and "anatomía:hígado" as a text query. The system identifies that he is using a mobile browser, translates the query and checks with his authorization rights. Then, searches in the hospital patient records and/or in the Internet depending on the authorization level. It returns the retrieved images and cases (e.g. a summary + a couple of images) on his screen on an appropriate (for the browser) layout.

- Scenario 2 – Teacher

A teacher is preparing the content of a lecture on differential diagnosis using a pathology (e.g. interstitial lung diseases) as an example case. He searches in his local computer and in the literature (internet) for MDCT images showing this pathology, using the text queries "pathology:interstitial lung diseases" and "modality:MDCT". He expects results relevant to his query but of large diversity. The system returns the retrieved images on his screen. He then marks 4 of the retrieved images as relevant and 1 image as irrelevant, and searches again. The system returns a new set of retrieved images using the reformulated query. It proposes similar searches using different modality or similar pathology.

- Scenario 3 – Researcher

A researcher is investigating the efficiency of a new approach in treating a pathology (e.g. Alzheimer's disease). He searches in the literature (Internet) for similar cases that use state-of-the-art approaches in order to compare the approach's results. For comparison reasons he is interested in studies that used images similar to his, so along with the "Alzcheimmer disease" text query, he uses as

a visual query, a set of brain MRIs of patients with Alzheimer's disease that he used in his study. The system corrects the mispelling, extracts metadata from the image (modality:MRI, anatomy:neuro) and retrieves the articles that are most similar to his combined query. It returns the results (e.g. a summary + a couple of images) on his screen.

A retrieval system should have the following functions in terms of queries:

- show me images visually similar to one or several example images
- text-based image retrieval as this is the most frequent (search by modality, anatomy, pathology, view and bio-system under observation)
- search by image region
- search by volume example or 4D
- search for videos or anything that is 2D and time
- search by sketch
- information extraction from images, so image given and system returns anatomic region, abnormal parts, pathology and other information
- search by patient ID to find all documents of a patient (does not need to be a patient record ID, but can be on an anonymous system)
- search for similar patients, potential for a single hospital stay or even over time, based on visual and clinical data available
- query refinement and relevance feedback

In the Appendix (Section 11.2), the following UML diagrams for the WP9 are presented:

- Use Case diagram
- Activity diagram
- Sequences diagram,
- And the list of the required functionalities.

6.4.3 Conclusions about the User requirements

The following are the main initial conclusions about the user requirements. These will be refined based on work currently underway in WP8 and WP9:

- Identification of the sequence of the user activities during a session of information retrieval
- Very interactive system for the information retrieval
- Dynamic UI (for querying, results interaction, etc.)
- Two different systems: the Search engine and the Batch system.
- Multilingual + multimodal system

Khresmoi requires a very lightweight method for integration. This is exactly what the SCA specification and the Apache Tuscany platform offer.

- Generic and simple mechanism of services composition,
- Configuration / re-configuration
- No code refactoring

In Khresmoi, we have to integrate very different components already advanced in terms of developments. So, development should not be intrusive in existing developments.

6.4.4 Integration decisions

ezDL is a key component between the user activities (GUI, search functionalities, result presentation, etc.), and the batch system (data mining). Hence, for the first prototype, the specification will be focused on this aspect and try to connect the relevant functionalities to enable very powerful queries to the end-users.

1. contextualization of the query,
2. user profile use to adapt queries (localization, language),
3. query model sensitive to user profile, (define some pre-defined fields to be filled).

➔ The first prototype should be centred on the ezDL, and it will be the main component to elaborate the integration.

OWLIM is considered as the main knowledge repository: for the first prototype, we should make the hypothesis that this component will be able to respond efficiently to any kind of queries. OWLIM will be useful to retrieve different kinds of facts. So, at the beginning, we can suppose that all the queries will correspond to facts.

1. As the Khresmoi system will have to provide more complex results, not only facts but an information produced/summarized according many facts, the next prototype will take into account more flexible processes to query.

➔ The first prototype should be based on the OWLIM repository

The scenario for the image and the textual search are considered differently.

1. The image and the textual search will be separated at the beginning to simplify the specification. This does not suppose that the first prototype will have the two search modalities not integrated, but it means that every modality will be implemented for its own. Also, the image modality will progressively take into account the textual search thanks to the semantic annotations of the image.
2. The hypothesis consists to say that as soon as possible the image modality should integrate the textual description of the image content. When this would be integrated, the image retrieval would be based on graphic and textual criteria.
3. Also, the multilingual modality will be preferentially applied on the textual search, but as soon as the textual dimension will be associated to images, the multilingual functionality will be applied also to the image search.

➔ The first scenario for image and text retrieval will be basically separated for the first prototype, and the multi-lingual functionality will be preferentially applied to the textual search.

Regarding all the different technologies and components we have to integrate, it is too complex to try to propose a full integration at the end of the first year. For this reason, different simplified scenarios for integration are proposed with only some components. We propose four different “integrated prototypes”:

1. Integrated prototype #1 = Textual search
2. Integrated prototype #2 = Multilingual search
3. Integrated prototype #3 = 3D Image search
4. Integrated prototype #4 = 2D Image search

Those simplified instances of the general scenario will be an interesting step before achieving the first full integration of the Khresmoi system. Those scenarios will be described in section 8.3.2.

7 Concepts of the Khresmoi architecture

The Khresmoi project aims to integrate many different successful components to build a powerful search engine for medical information. As described in the previous section (Section 6.3 about the integration technologies), the SOA approach seems to be the most appropriate to integrate those different components and to build a distributed, flexible, robust, and scalable system.

The architectural aspects are directly derived from the SOA principles, i.e. a set of several loosely coupled services, and supports the composition and re-use of them, where a service is a basic software component that provides a specific functionality. Moreover the method to identify and specify the services, and the structure of the Architecture will be presented. The logical view of the system permits to decompose the whole system with different functional areas. Thus, the system is composed of different layers that are composed of Service Categories. For example, the main services of the system are grouped into the Core Service. Every Service Category is composed of several services that can be deployed and used by all the other services.

The benefits of this approach are very important: the system is very flexible because it can be built from different configuration and services combinations. Also the different combinations can permit to build several applications that are adapted specifically for the Use Cases.

SCA specification permits to add the Service description around the existing components without imposing a component refactoring. The benefit of this approach consists to easily deploy the services and try to test the possible composition between them. For example, the four integrated prototypes will be deployed and tested with this approach. Moreover, the new functionalities can be specified quickly to be in concordance with the basic architecture. Finally, the result of the combination of the different integrated prototypes will be the full prototype.

7.1 Overview of the Khresmoi architecture (main layers)

7.1.1 Architectural overview

The Khresmoi architecture is based on the logical view of the system. This means that as inspired from the SOMA method (Section 6.3.2), the Khresmoi architecture refers to the different logical layers of the system (Application, Core and Persistence) and to the functional decomposition inside every layer. Each element of the logical view will be described separately and in relation with the whole system.

The Khresmoi system will be a powerful search engine for medical information. To define more concretely what we mean as a powerful search engine, the main characteristics that have an impact on the architecture will be described.

Firstly, a search engine implies two systems: the search engine itself that is used by the end users to retrieve relevant information and the batch system that analyses the information and creates the index that is used to compute the information relevancy. The search engine is an interactive application that should permit queries in different modes (text or images) and based on it, it should project the query against an index and retrieve relevant results. Contrary to the search engine, the Batch system is an asynchronous application. Every day it should process new collected data and produce the corresponding index to enrich the whole system.

Even if those two systems have different objectives, they could use specific services to work together, but also similar services to realize their specific tasks. The logical view of the architecture will permit to identify them.

As introduced before, the search engine should provide different modes for querying information. Queries could be defined with keywords, sentences, or conceptual descriptions (the keywords with the concept selection). But queries could be also defined with images. If an image is selected as a query, the system will retrieve similar images according to graphical criteria (e.g. colour histogram, texture similarities, region or form detection, etc.). Finally, the system has to take into account the multilingual aspects. For example, if the system does not return enough results in the query language, the system will translate the query into English to query the English index or the other indexes referring to other languages. For this reason, three main functional blocks of services are required for image, text and multilingual computation.

Two other important aspects should be studied:
The first one is related to the User Interface system. This part should support the interactive process that every user should be supposed to follow. For this, several functionalities to enrich, disambiguate the query, or to browse/organise the different results should be provided by the system. A particular functional block will be in charge to control/manage the different functionalities provided in the end user interface.
The second aspect is also very important because it concerns the required repositories to store the document, the NLP data set, the index and the knowledge that describes the different semantic relations between all those resources. Thus, the semantic repository is very important because this is the kind of information that is in general missing in information retrieval systems. Also, the big challenge of the project consists in integrating the huge amount of information from Linked Data and making it available within the search engine.

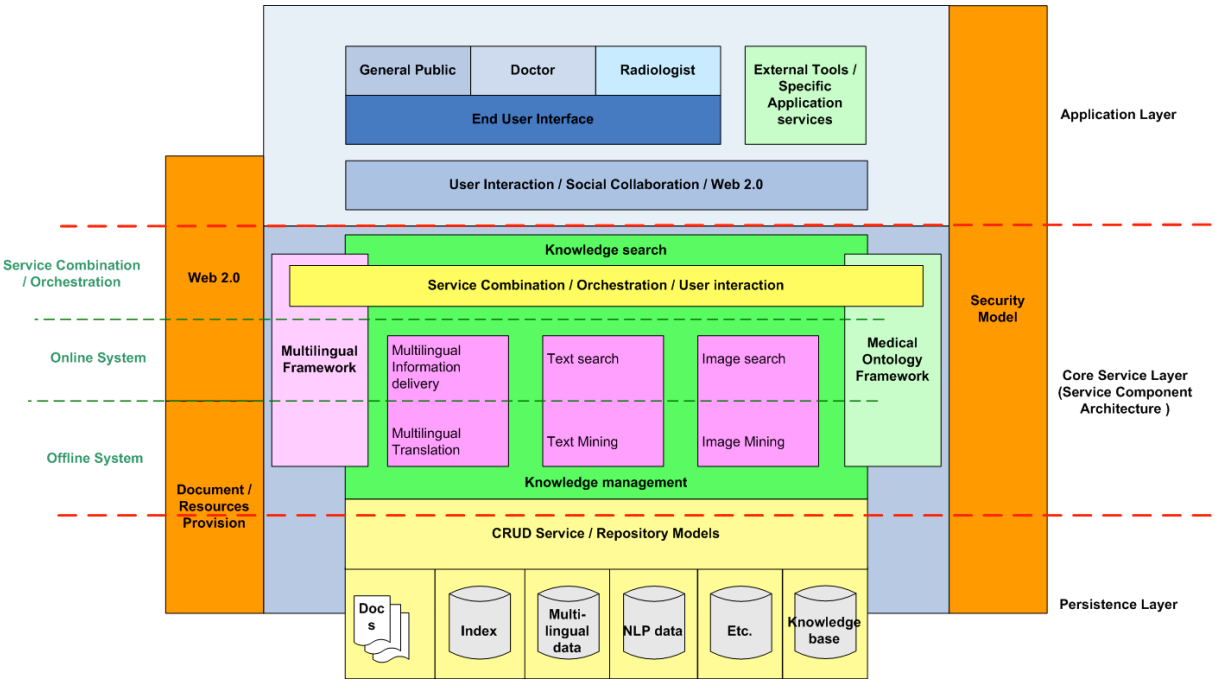


Figure 9. Architecture of the Khresmoi system

Thus, an integration platform is required to enable the efficient communication between all the software components and the scalable service deployment in terms of information retrieval and data processing.

For this, components integration is defined as a higher priority than performance. That is why the choice of SCA is the fundamental decision for the software integration. Based on the first integrated prototypes, first evaluation of the system performance will be done to find important requirements regarding the system flexibility.

7.1.2 The Layered Architecture

The basic architecture of the Khresmoi system is based on common SOA principles. The fundamental part is the logical view that should allow a modular and highly generic structure. For this, a three-tier approach was chosen. This means that the system is decomposed into three different layers which correspond to the following functional blocks: the application, the services, and the persistence of the system.

The **Application Layer** is the application provided to the end users. According the different uses cases defined in the project (WP8 and WP9), different applications can be built to provide adapted user interfaces according to the specific requirements such as those from the radiologists (WP9). The Application Layer should deal with the configuration of the user interface, and the management of the user interaction to dispatch the events towards the internal system (Service Layer). A very generic approach will be to allow the easy implementation of new applications adapted for specific business processes. For example, a new application for surgeons could be designed and implemented.

As suggested before the **Service Layer** is the core of the system. It contains all the main services provided by the system. Those services are called Core Services and as they could be numerous and very different, they are grouped by Service Categories. Those services are atomic functions that could be called whenever by the system. They will be specified with SCA and deployed through the runtime Apache Tuscany.

The last layer is dedicated to the system persistency, which is why it is called the **Persistence Layer**. It has in charge the mechanisms and models to store the information. For each kind of information, a repository is required to store the data. Each repository should provide a basic API to describe its own CRUD (Create, Read, Update and Delete) functionalities to permit easy access to the data. After the first results of integration, a generic approach would be studied to define a standard API for services connexion to the Khresmoi platform. In Khresmoi, the most important repository is related to the storage of the knowledge. As the formalism to represent the knowledge in Khresmoi is based on RDF, RDF repositories will have an important place in the architecture.

The following picture (Figure 10) represents the different layers defined in the Khresmoi architecture.

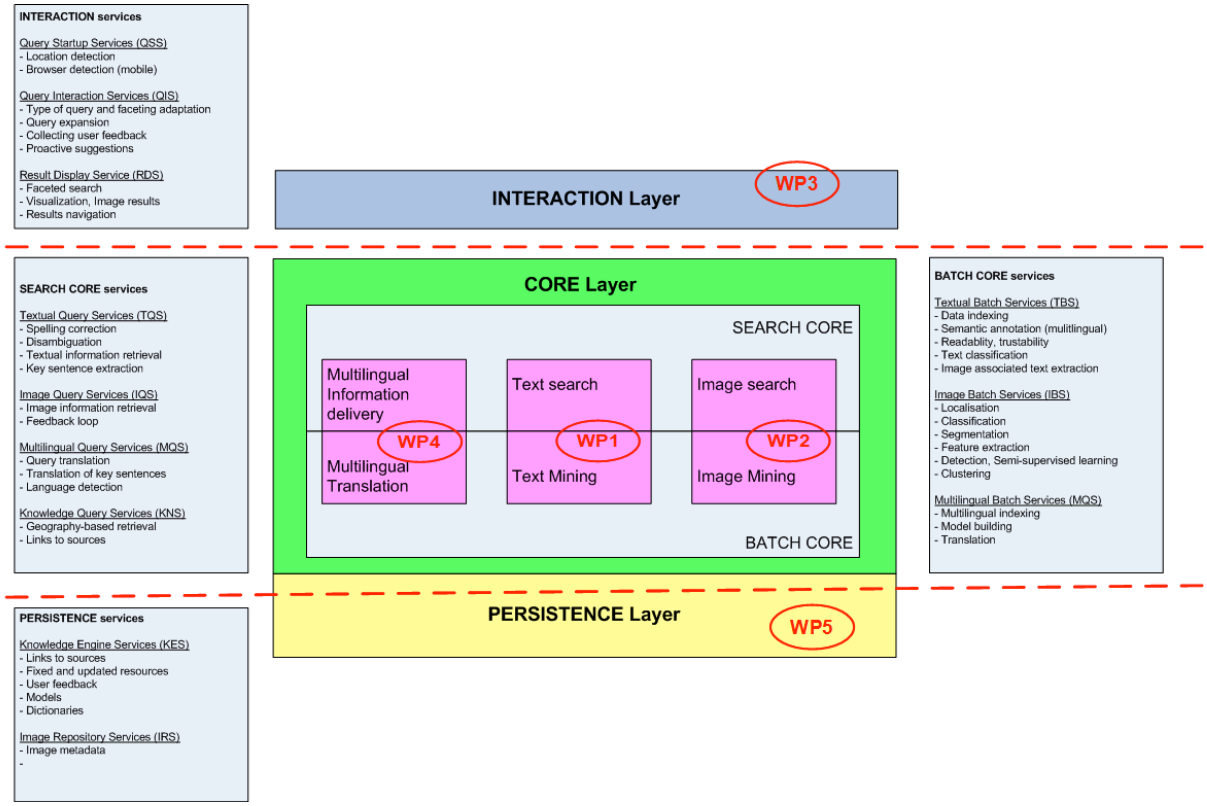


Figure 10. Architectural layers and main services

7.2 Two complementary core systems

7.2.1 The Service Layer

As explained before, the Service Layer contains the main services, but as we identify two complementary systems (Section 7.1.1), this layer should follow this decomposition. Then, the main services will be organised according to those two systems. The services for the Search system are grouped in the category called Search Core, and the services dedicated to the Batch system are grouped in the category called Batch Core.

In addition to this decomposition, a more specific decomposition can be done under the previous one. Effectively, different kind of functionalities will be used by the two systems (search and batch). So, some functional categories can be used to group the main services of the system. These important categories are the textual functionalities, the image functionalities and the multilingual functionalities.

The structure of the Service Layer is represented by the following tree:

- Search Core Services that should provide composition flexibility to support the interactive process of information retrieval:
  - Textual Search services
  - Image Search services
  - Multilingual Search services
- Batch Core Services that should be organized as pipeline to support the processing workflows required to indexed semantically all the resources:
  - Textual Batch services
  - Image Batch services
  - Multilingual Batch services

In the next sections, the description of the different services for every category is provided.

7.2.2 The Search Core services

The Search Core corresponds to the set of services that are dedicated to the information retrieval processes. According to the SCA implementation, those services are atomic services that could be combined to satisfy specific Use Cases.

Inside the Search Core, different categories of functionalities are available. Each functional category fits with the module decomposition discussed previously; one is related to the tactual search, the second to the image search and the last to the multilingual search.

The specification of the Search Core is the most important to start the system design for two reasons. The first reason is related to the software conditions because the Search Core services correspond directly to the functionalities provided by the components.

The second reason is related to the user requirements. In fact, as the system design is oriented by the user requirements, all search functionalities will be essential to define the whole engine. This means that all the user requirements found in each use case will be taken into account to identify the main services that should be implemented in the Core Service. For example, the use case focused on general public (WP8) is focused on textual and multilingual search possibilities:

- Spelling checking,
- Conceptual expansion,
- Query translation, etc.

In contrast, the use case focused on Radiologists (WP9) is more focused on the image search (2D and 3D):

- Image similarities,
- Image and text combination, etc.

The characteristics of the Search Core are:

- Very interactive
- Synchronous processes

7.2.3 The Batch Core services

Similarly to the Search Core, the Batch Core is a set of search services. Functionally, the Batch Core has the same decomposition as the Search Core.

The specification of the Batch Core is not central for the first prototype because it should be driven by the Search Core specification. It means that all specific functionalities used from the Search Core correspond to a data access or data transformation. Then, the kind of data required as a relevant result to fit with the user request will be inferred and specified from the search requirements.

The characteristics of the Batch Core:

- As a pipeline
- Asynchronous processes
- Huge performance for data processing.

7.3 Description of the Khresmoi services

7.3.1 1 - Application Services

At the end, the Application Layer consists of the service used directly by the front end system, i.e. by the user interface implemented to search medical information. Then, it should exploit all the implementation of the business processes that are the result of the composition of Core Search services.

Mainly, this layer is covered by the ezDL tool. This component deals with all the user interactions, it means that it can intercept the different events and dispatch them towards the appropriate services. Thus, the Application Layer is composed of the user interfaces that are based on a Web Application framework such as AJAX and the controller of the user interactions. The controller is ezDL itself that would be extended to fit with the Khresmoi requirements (for example, the logging of the user activities with the semantic annotation). Also, the controller contains a set of configurations to adapt the search engine to different situations (such as adaptation to a specific use case, adaptation to a user profile, selection of particular libraries, etc.). In some cases, it will need to be connected with external tools to exchange interesting information about the users (like the tags, the social networks, etc.), or about the medical resources (like the new resources, the conceptual updates). Then, specific adapters would be implemented to enable those kinds of connections.

Based on the components analysis and the user requirements, different categories of the Application Services have been defined. The first category is called **Query Startup Services (QSS)**; it corresponds to the different services required by the system to configure a user session of information retrieval. The second category is called **Query Interaction Service (QIS)** because it is composed of all the services required to manage the query edition. It should catch the user actions to analyze the content of the query. According to the status of the query, the system will suggest reformulation or information to semantically enhance it. Finally, the last category is called **Result Display Service (RDS)**. Those services are dedicated to present and display the results provided by the search engine.

Service Category	Service name	Functionalities
Application Services	Query Startup Services (QSS)	○ Location detection ○ Browser detection (mobile)
	Query Interaction Service (QIS)	○ Type of query and faceting adaptation ○ Query expansion ○ Collecting user feedback ○ Proactive suggestions
	Result Display Service (RDS)	○ Faceted search ○ Visualization, Image results ○ Results navigation

Table 16. Services and functionalities for the Application Layer

7.3.2 2.1 – Search Core services

The Search Core services are the main services used by the system to analyze the user request and provide relevant results. Three main categories have also been defined for the Search Core.

The first category is the **Textual Query Services (TQS)**. This category contains all the services used to treat textual queries. Of course, the corresponding services to provide results to the textual queries are also grouped in this category. The second category is similar to the previous one but for the image modality. It is called **Image Query Services (IQS)**. In this situation, the query is not a set of keywords but an image taken as example. Finally, the third category is the group of all the services that should

deal with the multi language and is called **Multilingual Query Services (MQS)**. Concretely, it corresponds to the translation functionality that could be applied on a sentence, a set of key words, or on concepts. Before that, one service should detect the language of the query.

Service Category	Service name	Functionalities
Search Core Services	Textual Query Services (TQS)	○ Spelling correction ○ Disambiguation Query type detection and query transformation ○ Query processing ○
	Image Query Services (IQS)	○ Image information retrieval ○ Feedback loop
	Multilingual Query Services (MQS)	○ Query translation ○ Translation of key sentences ○ Language detection
	Knowledge Query Services (KQS)	○ Geography-based retrieval ○ Links to sources

Table 17. Services and functionalities for the Search Core Layer

The Search Core services need significant attention in terms of integration. In fact, they are the first service to be specified to deploy firstly the search engine. Then, the search engine will be presented and tested to end-users. Results of this will imply the refinement of the full system and especially the refinement of the Batch system to improve the data processing and provide the most relevant information.

7.3.3 2.2 – Batch Core services

The Batch Core should produce all the data that would be expected as a relevant result for the end user. For this, many different resources (documents, images, web pages, etc.) should be analyzed and indexed and potentially translated into English.

Thus, each modality (text, image, and multilingual) has its own category of services inside the Batch Core. At first, the services dedicated to textual analysis (or NLP) are grouped in the **Textual Batch Services (TBS)**. Those services are able to analyze a huge quantity of textual information from specific data sources and from the web. The data is extracted and analyzed with linguistic criteria to generate the index that will be used by the search processing. Also, the index can be enriched with semantic information. Linguistic and semantic information will be used for the translation process.
The second category of the Batch Core is called **Image Batch Services (IBS)**. Its services are used to analyze the images (2D, 3D, and 4D). Based on graphical criteria, the Image Batch services produce also an index that permit to retrieve image similarities. On top of the graphical criteria, some textual criteria will be also added. The last Batch category corresponds to the multilingual categories, called the **Multilingual Batch Services (MBS)**. To translate the textual content, those services use the index produced by the Textual Batch Services and some languages resources.

Service Category	Service name	Functionalities
Batch Core Services	Textual Batch Services (TBS)	○ Data indexing ○ Semantic annotation (multilingual)

		○ Readablity, trustability ○ Text classification ○ Image associated text extraction
	Image Batch Services (IBS)	○ Localisation ○ Classification ○ Segmentation ○ Feature extraction ○ Detection, Semi-supervised learning ○ Clustering
	Multilingual Batch Services (MBS)	○ Multilingual indexing ○ Model building ○ Translation

Table 18. Services and functionalities for the Batch Core Layer

Contrary to the Search Core, the service of the Batch Core cannot be specified in details. After the first release of the Search engine, new requirements form the end-users will be collected, the results of this new iteration will lead the specification of the Batch Core services.

7.3.4 3 - Persistence Core services

Owlim is a family of high-performance semantic repositories, or RDF database management systems, with the following characteristics:

- native RDF storage and retrieval engines, implemented in Java and compliant with Sesame
- robust reasoning support for the semantics of RDFS, OWL Horst and OWL 2 RL
- high scalability, loading and query evaluation performance.

The repository can be accessed both locally and remotely via the Sesame API. However, the most generic access is provided by exposing it as a **SPARQL endpoint**. A SPARQL endpoint is a conformant SPARQL protocol service as defined in [12] with HTTP bindings for data transfer. It allows querying the knowledge base by both users and machines using the SPARQL language. Results are returned in one or more machine-friendly formats (e.g. XML, JSON).

Owlim provides outstanding multi-user query support. Moreover, the system can be extended with plugins to enable additional features, e.g. full-text querying integrated with SPARQL.

Service Category	Service name	Functionalities
Persistence Services	Knowledge Engine Services (KES)	○ Links to sources ○ Fixed and updated resources ○ User feedback ○ Models ○ Dictionaries
	Image Repository Services (IRS)	○ Image metadata

Table 19. Services and functionalities for the Persistence layer

7.3.5 4 - Additional modules

1.1.1.3 Authentication module

The identification of the user is very important for the search engine system. It permits to ensure the information privacy about personal information. This topic is particularly sensitive in the health domain.

Moreover, it also permits some personalization of the information. For example, the User Profile will be defined with several ontologies. One will be dedicated to the preferences of the user, another to the activities realized in information search. Other ontologies could be also used to describe the important topics of the user, and his/her annotation activities to share knowledge with peers. Also, geographical ontologies can used to characterize the geographic position of the user, and infer the required information in terms of language translation or the health specificities.

This information will be used to infer relevant search strategies for the user. The specification of this module has already started but is still in discussion between WP3 and WP5.

1.1.1.4 Socialization module

The socialization is an important module of the system, not for the first prototype, but during the second year it will be specified. This module permits to exchange knowledge between the users. This functionality can be very useful in the case of medical experts.

Also, the content annotation with semantic information or tagging activities will be useful to disambiguate the information. This is particularly true in the case of the image description, which is essentially analysed to extract graphic criteria. On top of this, semantics would be added and taken into account by the search engine.

This module will be correlated with the authentication module because some ontologies will also be used to describe the socialization activities. For example, the foaf[9] and sioc[10] ontology that permit to describe blogging and annotating activities will be used in the user profile to enable the connection with the social network, with different kinds of relations such as friends, topics, patient-doctor, etc.

1.1.1.5 Security, Logging, and tracking modules

Many different modules are also required by the system. Progressively, they will be specified, implemented and integrated in the Khresmoi system. This approach is fully compliant with the SCA implementation.

As several organizations (hospitals, companies, and universities) would be able to deploy a particular Composites application, the protection of the internal information is very important. For this, message exchange protocols would have to take care of this issue. Also, the logging functionality of the different tasks triggered by the system would be highly relevant to track the bugs in the two complementary systems, but especially for the Batch system.

7.4 Technical requirements

Here is the table that compiles the different technical requirements collected from the different components providers.

[9] foaf ontology: http://xmlns.com/foaf/spec/

[10] sioc ontology: http://sioc-project.org/ontology

Category	Topic	Detail	Metric / Feature
Integration	Architecture flexibility	JEE Interface	High flexibility
	System integration	GIFT – uses MRML messages for interfacing with clients	
		MATLAB Java Builder environment	Deployment procedures are available
	SOA specification for services deployment	Interface is provided	Java / WSDL
Interoperability	End user access to the system		Restricted/Authorization?
Data processing	Size of processed data	GIFT – the image is converted into 256x256 pixels before feature extraction	
		Data is in DICOM format that contain 2D, 3D, and 4D imaging data	Very large
	Time of processing data	GIFT - indexation of an image takes 0.5 sec	
		Feature extraction, learning, retrieving	Very processing intensive – requires multiple CPUs
	Document format	DICOM standard	
	Document encoding	DICOM standard	
	Priority of documents		
	Structured data providers		
	Incoming data throughput		
Data storage	Data storage formats		Binary, RDF
		PACS Node	
	Data storage maximum capacity		20 x 109
		No limit	Typically PACS Nodes store several gigabyte of data
	Data storage growth rate	High growth rate	Typically > 100GB per day
Data formats	Ontology format		RDF, RDFS. OWL
Deployment	Deployment configuration	Deployment will be achieved using scripting	
System integrity	Resource exhausting	Will return an error	
	Component crash reaction	Calling system needs to react on the error code and may rerun the component	
System specific technical requirements	Ontology Visualization		
	Ontology loading		
	Ontology versioning		
	Language detection quality		
	Semantic Tagger quality		
	Visualization		

Table 20. Full technical requirements for the Khresmoi system

8 Specification of the Khresmoi system

8.1 Specification of the software system

Service Component Architecture (SCA) is a model that permits for the assembly of service components into business solutions. The SCA specification simplified the component programming model for implementing easily services. Moreover this general point, SCA provides many more benefits that really fit with the SOA requirements:

- The components integrate with other components without needing to know how other components are implemented. This is very important to obtain a loose coupling system.
- The components can easily be replaced by other components, and this provides a very high flexibility to the system.
- The services can be easily invoked either synchronously or asynchronously.
- The Composition mechanism is very clearly described.

This kind of model is very useful in the case of the prototyping a complex system that should integrate many different technologies. Also, the easy way to build composite application by composition of components or other composites can increase significantly the productivity. Finally, SCA simplifies development experience for all developers, integrators and application deployers.

Based on the architecture definition (Chapter 7), the entry point to start the specification of the system is to list the different components that will be used in the Khresmoi system. This can be done for the Search and the Batch system.

After that the SCA principles will be described concretely. Then, the integration aspects of Khresmoi will be clearly presented.

8.1.1 Components overview for the Search System

Based on the user requirements (and the general scenario) and the existing components, the main components and functionalities that will be implemented and integrated to provide the first search engine prototype were identified.

For example, we see that the ezDL component is a central component. It will be connected with many other components to enable the end-user interactions.

Also, Owlim permits to store the RDF data. In the case of the Search engine, especially the user interaction (query activities, and content interaction) will be stored in the repository. But the search engine will use also RDF repository as an index repository to provide relevant answer to the users.

For every component, the description of the services provided and the other components dependencies will be represented formally in a UML diagram (Figure 11).

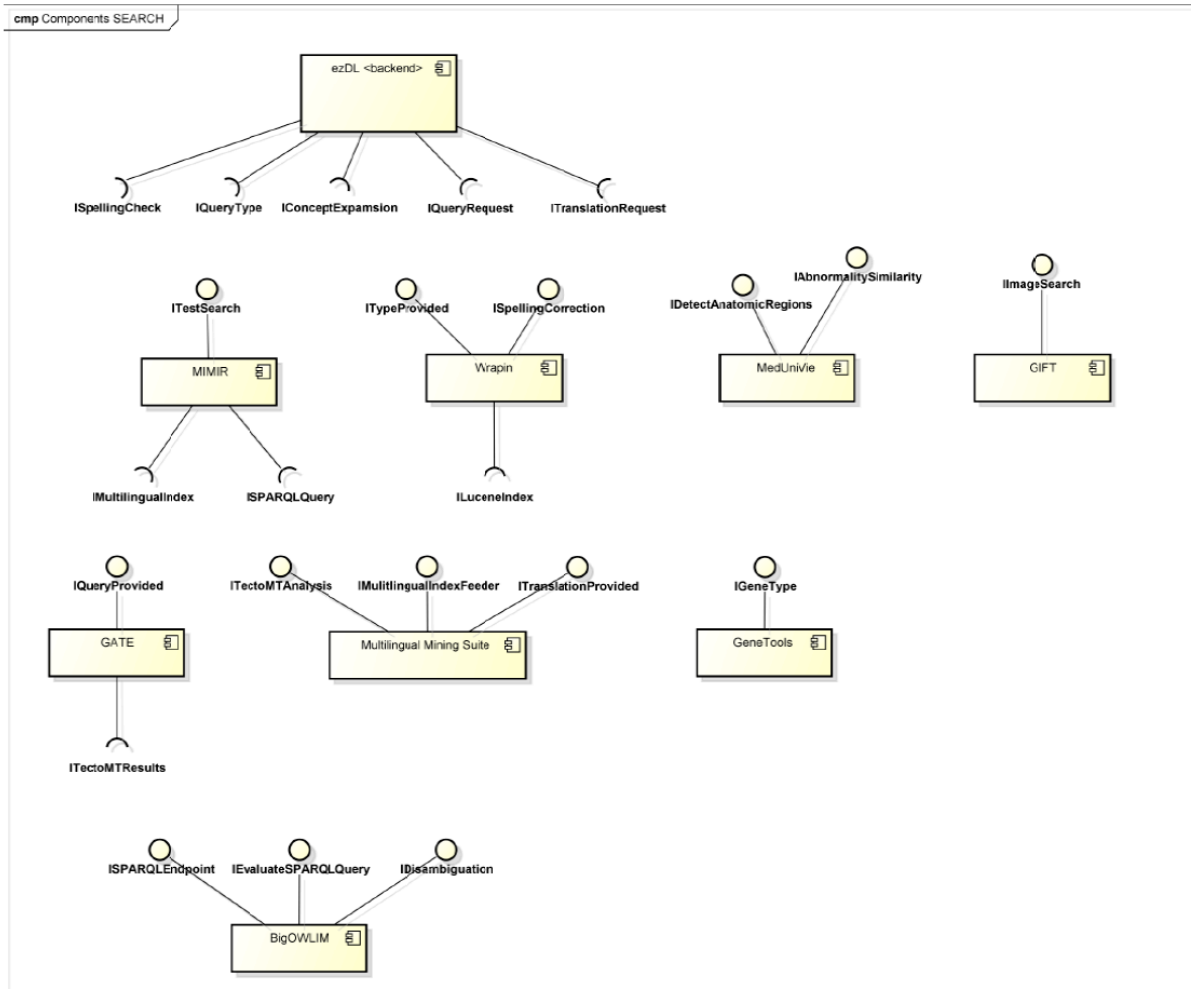


Figure 11. UML diagram for the components overview of the Search system

Also, the details about the component implementation are provided in the appendix (Cf. section 11.4).

8.1.2 Components overview for the Batch System

The main components of the Batch system are those that are dedicated to the data processing. Most of them will be connected with the RDF repository to store the data processing results.

Also, to obtain different RDF results that should be stored in different repositories, the Talend component will be used to organise the workflow of data integration between those different RDF repositories.

Below is the same UML representation as the previous section but for the Batch system:

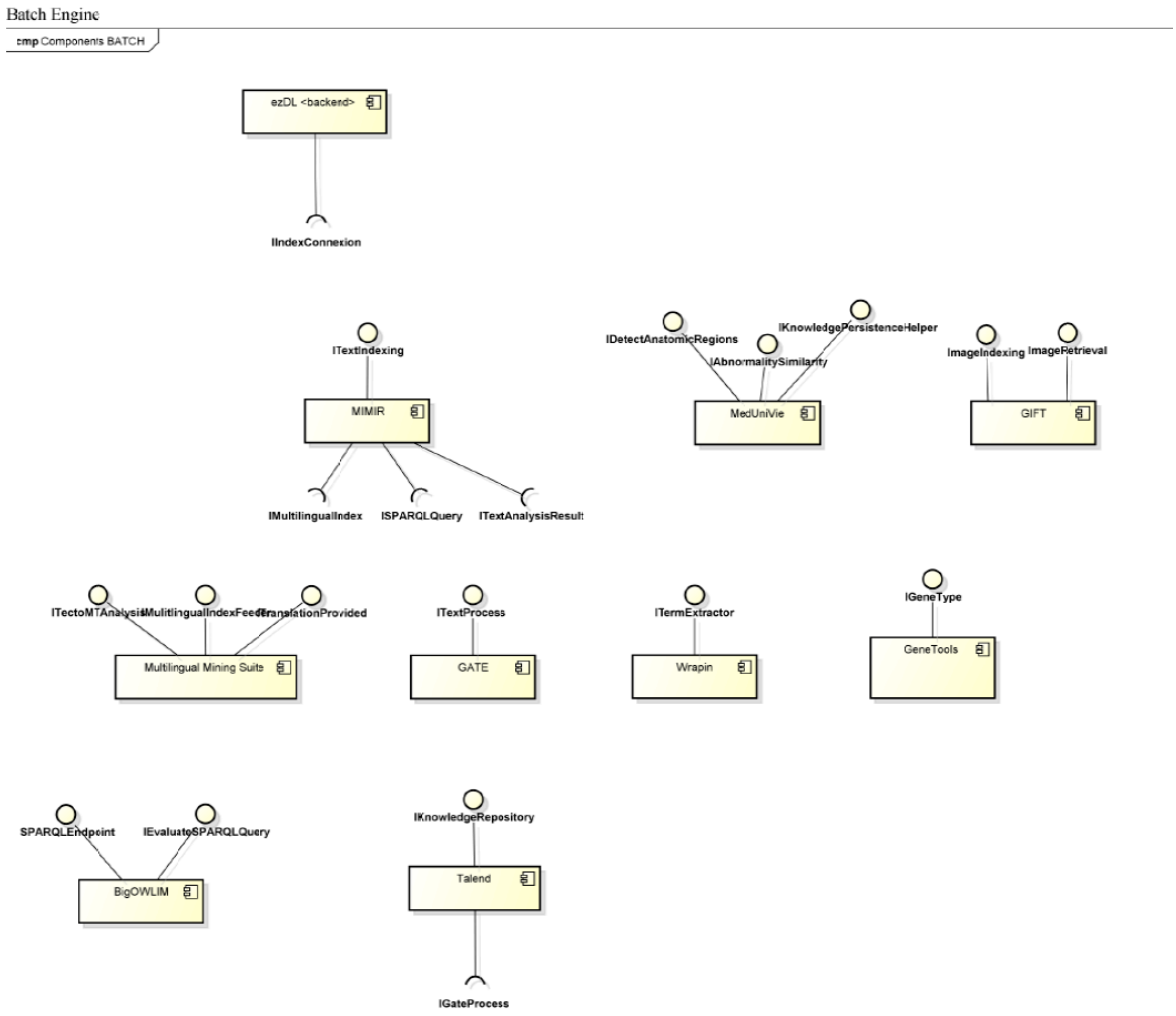


Figure 12. UML diagram for the components overview of the Batch system

More details about the Batch components will be provided after the evaluation of the first release of the search engine.

8.2 SCA principles for the integration

To implementation an SOA system, SCA provides four interesting principles.

1. First, they propose the specification for the Component Implementation. This specification helps to define the services according to the implementation language of the component (i.e. Java, Spring, C++, BPEL, PHP, etc.).

2. Secondly, they provide the Assembly Model that permits to describe the composition of different components. This Assembly Model is recursive, and then it enables the possibility to create a new composite that is the composition of different composites.

3. SCA also proposes the specification of the different bindings. The binding defines how the methods could be accessible and used by other components. For example, the possible bindings could be Web services, JMS, RMI-IIOP, REST, etc.

4. Finally, SCA defines the Policy Framework to facilitate the new infrastructure services in the deployment environment. For example, Security, Transactions, Reliable messaging and other aspects could be integrated.

8.2.1 SCA elements for the system design

To start working with SCA implementation, the main element to understand is the component because this is the basic element that will be used to build the system.

The component is a piece of software that provides different functionalities. It should be described with different elements:

- The services: a service is the declaration of a specific functionality (or method) provided by the component.
- The references: a reference is the definition of the components required to works properly.
- The binding: the binding is the way to use the component.
- And eventually some properties, to express specific data that are used to configure a specific context to use the component.

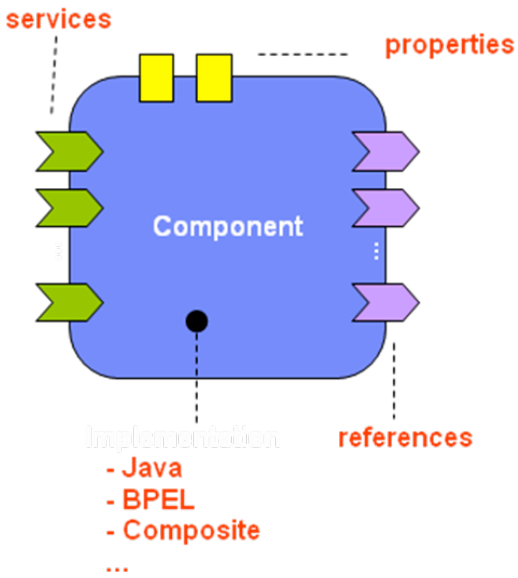


Figure 13. Graphic representation of the SCA component

The composite is a basic unit of composition that permits to assemble the implemented service components and to deploy them together.

After the services deployment, they are considered as public and can be consumed by other components. To define the specific connection between two components inside the composites, a wire is declared. A wire permits to link the service (functionality to be consumed) and the reference of the component (functionality required to achieve the process).

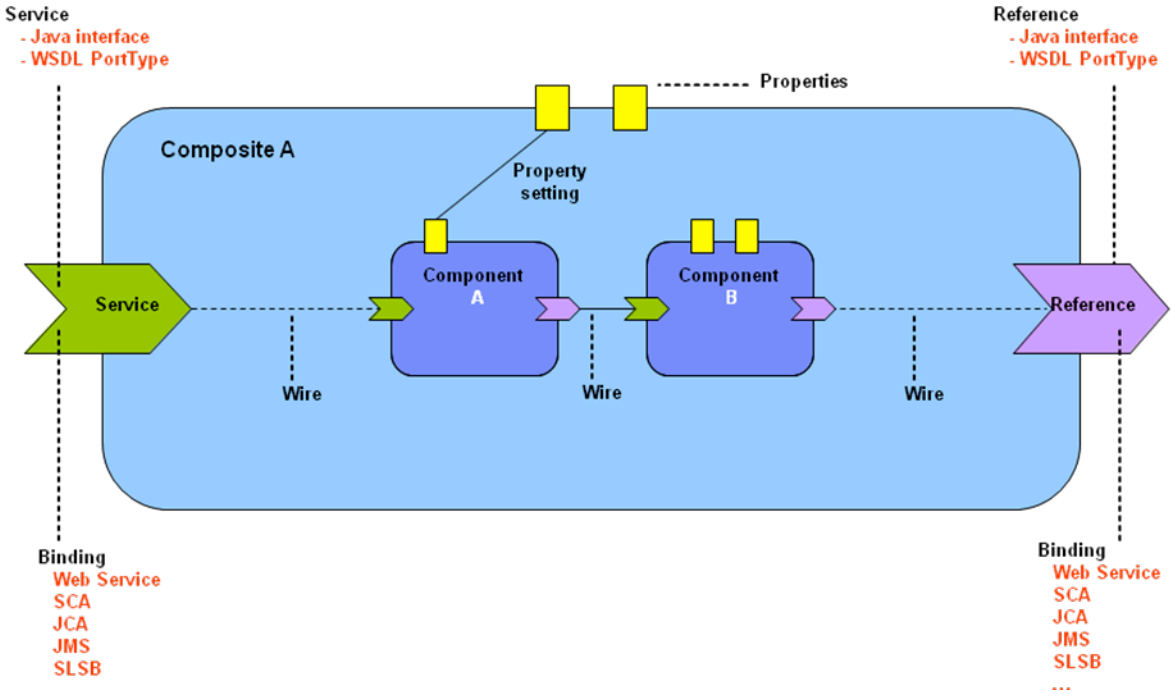


Figure 14. Graphic representation of the SCA composite

The composite represents the composition of different components, but it can also represent the composition of different composites. This composition mechanism may be used as an implementation of components at next higher layer.

8.2.2 Component description with SCA

To define the different configuration of services deployment in the system, SCA provides an XML format. With this formalism, the different components should be declared and their respective relations with the other components, or composites. Also, the binding of the components are declared at the same place.

Below is an example of an XML description of a composite (Figure 15). The components, services, and references are described.

```
<composite xmlns="http://www.osoa.org/xmlns/sca/1.0"
      name="bigbank.accountcomposite">

   <service name="AccountService">
      <interface.java interface="services.account.AccountService"/>
      <binding.ws port="http://www.bigbank.com/AccountService#
      wsdl.endpoint(AccountService/AccountServiceSOAP)"/>
      <reference>AccountServiceComponent</reference>
   </service>

   <component name="AccountServiceComponent">
      <implementation.java class="services.account.AccountServiceImpl"/>
      <property name="currency">EURO</property>
      <reference name="accountDataService" target="AccountDataServiceComponent"/>
      <reference name="stockQuoteService" target="StockQuoteService"/>
   </component>

   <component name="AccountDataServiceComponent">
      <implementation.java class="services.accountdata.AccountDataServiceImpl"/>
   </component>

   <reference name="StockQuoteService">
      <interface.java interface="services.stockquote.StockQuoteService"/>
      <binding.ws port="http://www.quickstockquote.com/StockQuoteService#
      wsdl.endpoint(StockQuoteService/StockQuoteServiceSOAP)"/>
   </reference>
</composite>
```

Figure 15. XML description of the SCA composite

This file contains all the different components that should be deployed in the system. It is used by the runtime environment to deploy the different services.

8.2.3 SCA annotation of the Service Implementation

As described before, the different components are declared in a specific XML file that is used to configure the system deployment. The information from the XML file points towards the component implementation.

Then, in the implementation, it is required to characterize what will be used as an SCA element. For example, in the Java implementation, some annotations may be used. The different Java Annotations serve for generating the corresponding componentType.

Two different kinds of annotations are provided:

- Implementation annotations
  - @Service
  - @Reference
  - @Property
  - @Scope @Init @Destroy @EagerInit
  - @ConversationID @ConversationAttributes
  - @ComponentName
  - @Constructor
- Interface annotations
  - @AllowsPassByReference
  - @Callback
  - @Remotable @Conversational
  - @Oneway

Here, this is an example of the utilization of the annotations used in the Java implementation to describe SCA elements.

```
package services.account;
...
public class AccountServiceImpl implements AccountService {
   @Property
   private String currency = "USD";
   @Reference
   private AccountDataService accountDataService;
   @Reference
   private StockQuoteService stockQuoteService;
   ...
   public AccountReport getAccountReport(String customerID) {
      ...
   }
```

8.3 Approach for the Khresmoi integration

8.3.1 Integration plan

Based on the user requirements and the components analysis, the architecture of the system has been built. For each logical module, the components and the main functionalities have been identified. These different components can be described with the SCA specification to be deployed and integrated in a homogenous environment. The components identification is the first step to build an integrated system. Let's present what will be the following steps to achieve the first integrated prototype:

1 – Description of all the components with SCA (in Section 8.2)
- Define all the SCA component and test their deployment

2- Clarify the scenario of the integrated prototype (in Section 8.3.2)
- Identify the Component interface according to the references
- Specify some composites

3 – Deploy locally the services and tests the different composites (in Section 8.3.3)

8.3.2 Different scenarios for a progressive integration

As decided in Section 6.4.5, the integration work will be parallelized in four different prototypes. This step is a composed of the specification of the different integrated prototype and the implementation of them. The specification is an instantiation of the general specification of the software system. Both are still on progress even if the specification is quite clear now (in Appendix 11.3). A summary of the different integrated prototypes is provided now:

1. Integrated prototype #1 = Textual search: the first integrated prototype consists to implement the search engine focused on text modality. The main components that have to be integrated are the Mimir, Owlim, Wrapin, and ezDL.

2. Integrated prototype #2 = Multilingual search: the second prototype will integrate essentially the translation functionalities with Mimir. For this, the Czsem Mining Suite provides multilingual indexes that be used by Mimir. Then, the components that have to be integrated are the Mimir, the Czsem Mining Suite, and ezDL.

3. Integrated prototype #3 = 3D Image search: the third prototype is focused on the 3D search.

The amin objective consists to make compliant the images index produced by MedUniVie with Mimir. Then, 3D searches will be possible from Mimir. The main components that have to be integrated are the Mimir, and the MedUniVie tools.

4. Integrated prototype #4 = 2D Image search: finally, the last prototype hast to integrate Mimir with the GIFT tool that produces 2D images index. Like the third prototype, the 2D searches will be able from Mimir.

As they use the same integration framework (Apache Tuscany), they will also be compliant with each other. For example, as the Mimir and ezDL will be integrated in the first prototype and Mimir will be integrated with GIFT in the fourth prototype, ezDL will probably requires an adaptation of the user interface, but will be complaint to retrieve text and 2D images.

The SCA mechanism to compose services was presented in Section 8.2.1 (cf. the composite). Here we present two examples of composition.

The first example concerns the Textual search. The following schema represents the SCA composition of the main components required by the Search Core. Every component will be composed of the functionalities provided by the existing components (Section 11.3).

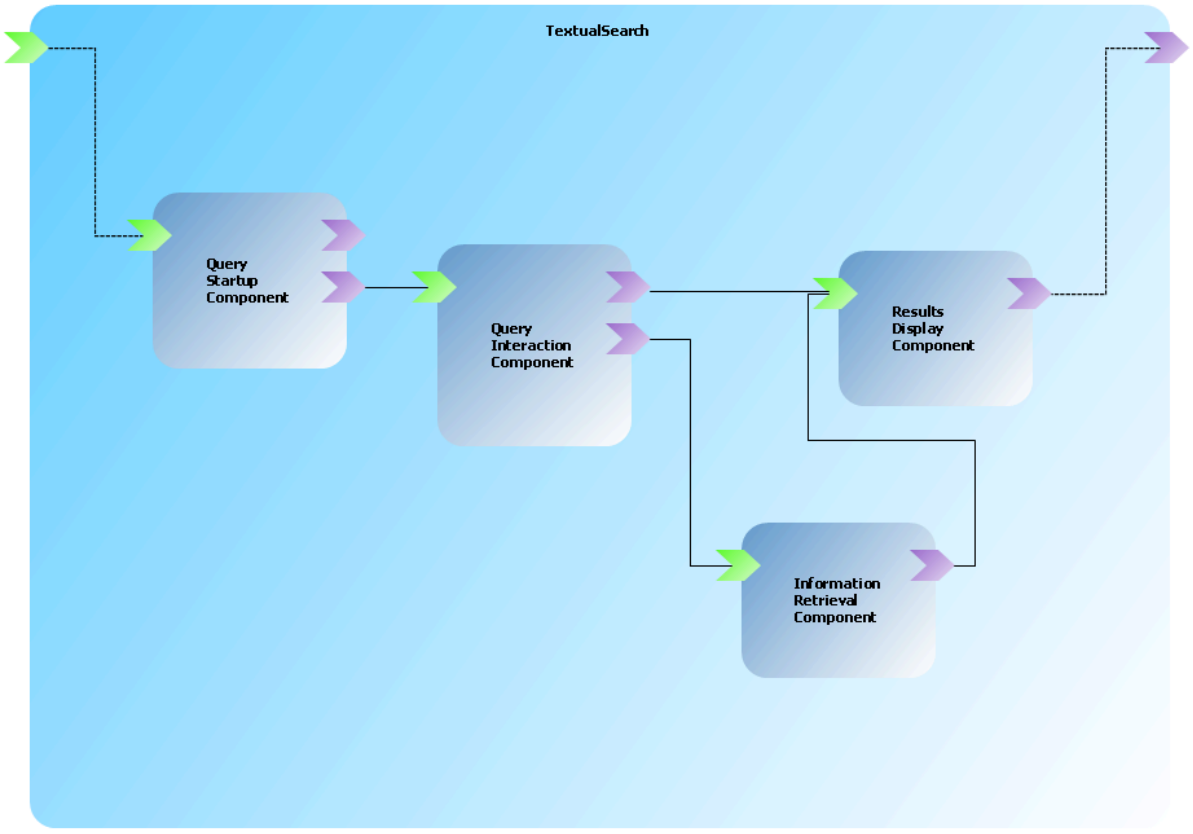


Figure 16. First example of the SCA composite specification

For example, the Query Interaction component is composed of mode detailed components. The different functionalities required to support the query interaction could be represented in the following figure:

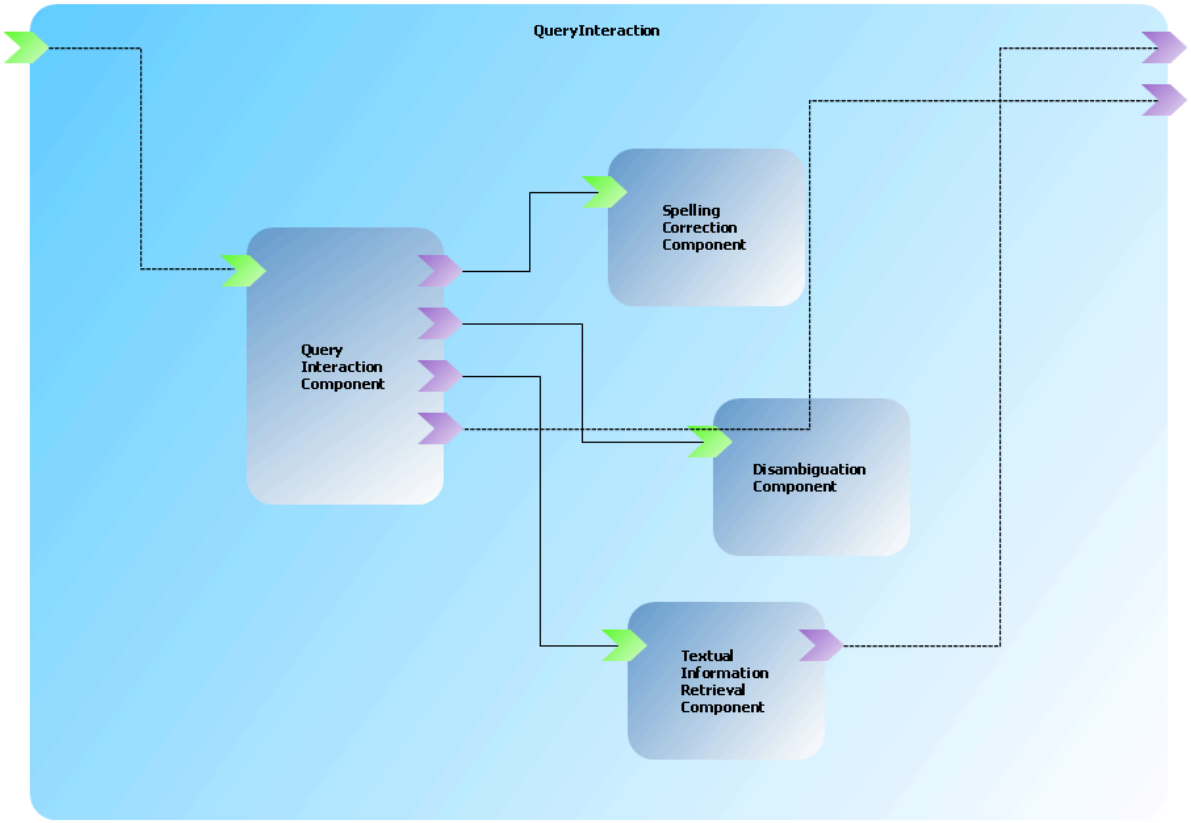


Figure 17. Second example of the SCA composite specification

Both examples show the possibility to built very scalable composition of services with SCA. From atomic services provided by the existing tools, it is possible to build a more complex system. In the appendix, the full description of the component composition is described (Section 11.3).

8.3.3 Deployment of the integrated prototypes

As decided in the state of the art, Apache Tuscany is the most relevant runtime that could deploy easily the SCA implementation. In particular, it provides the Tuscany Domain manager, that permit to configure server nodes where several composites will be deployed.

Then, different configurations of composites can be deployed and tested to evaluate their efficacy. The system can be deployed locally (all composites are deployed on one local server), but also, it can be deployed in a distributable way (several composites deployed on several servers).

SCA really simplifies the composites deployment and is completely compliant with the requirements of any cloud infrastructure.

8.4 Technical specification

8.4.1 Software Development and Integration

Below, is presented the guidelines and the different tools selected to implement the Khresmoi design.

1.1.1.6 Development methodology

The development methodology is based on Agile principles to help software developers and increase their productivity:

- Individuals and interactions over processes and tools,
- Working software over comprehensive documentation,
- Customer collaboration over contract negotiation, and
- Responding to change over following a plan.

Both aspects of each principle are important, but the right side is preferred to the left one. These principles try to simplify the development conditions and reduce the complexity of a very big project. In addition to the development guidelines, a work plan will be elaborated to organize the integration work. This framework will be very useful to support the developer activities, but it will not be rigid. As recommended by the Agile approach, adaptation to changes will be very important too.

To apply the Agile principles, a particular methodology has been selected, called Scrum. As this approach has been defined with very concrete constraints for the day by day support, and face to face interactions, the Scrum methodology could not be applied entirely, only some aspects could be very efficient to increase the developer productivity.

- The development is based on very short iterations. Those iterations are called “sprints” and are focused on very specific tasks.
- Tasks are clarified and agreed before the beginning of the sprint.
- The Scrum team is supervised by two different roles: one to check the development tasks before starting the sprint and will manage the sprint (ScrumMaster), and one that control the WP development, and supervise all the team developments (ProductOwner).
- Although the both team supervisors, the Scrum team is auto-organised: it means that tasks are selected by the developers according interest and priority level. In case of difficulties, solutions are found in group.

1.1.1.7 Development process

The development phase depends on the results obtained from the analysis and requirements activities. So, those preliminary steps should be carefully realized to define very clear specifications. To be productive, the developers should be focused only on development tasks. For that, developments tasks should be very clear, to avoid the refinement of the specification during the development tasks. The mix between specification and development tasks could be very confusing for developers, and imply a severe decrease of productivity. That’s why short and iterative steps called sprints are promoted by the Scrum methodologies.

- Before the sprint: all the analysis and specification activities are deeply realized to provide very clear specifications. Those specifications are translated into very clear tasks for developer.
- To start the Sprint: tasks are clarified; corresponding responsibilities are defined in face to face meeting. To increase productivity, it is better to do not change those conditions during the sprint.
- During the Sprint: the conditions of the sprint could not be modified, and the ScrumMaster has to control the evolution of work.
- Tests and continuous integration can be used to ensure the code validity all along the development process inside the team.
- After the Sprint: results and process are analysed, and eventually refined to start a new sprint.

Communication inside the scrum team is an important key point of success. Then, regular meetings should be planned to follow the development process. Also, some tracking tools will be provided to reveal and share the development status to all the developers.

1.1.1.8 Development team and responsibilities

In Khresmoi, there are two kinds of Scrum teams. One is based on the developments tasks defined by a WP, and the other is dedicated to the integration tasks.

Firstly, the structure and activities organisation of the Scrum team is very relevant for each technical WP to organise the programming work in Khresmoi. Every WP can define a specific software piece composed by very different software components.

- The Scrum team is lead by the ProductOwner that has to supervise all the development tasks. So, for each WP, the WP Leader or his deputy could be in charge to control that specifications are well done, and the corresponding sprints are well accomplished.
- More specifically, the ScrumMaster is responsible to supervise the different sprints. One or several technical experts could be in charge of this responsibility. He has to be close to the developers to identify errors, difficulties, and support them to conclude correctly the sprint.

Secondly, the integration work needs to involve people from the WP6, but also technical experts from each technical WP. Then, some experts involved in the WP work could be grouped to form an integration team. This is the principle of the Scrum of a scrum where members of different scrum team can be grouped to form a new scrum team to achieve more global tasks. That's why the principle can be used to compose the integration team:

- As defined before, the structure of the Scrum team should be applied for each WP. Then, those different development teams can be also reorganised together as a team for the integration tasks. This integration team will be composed of the responsible person of the technical teams and some developers. Together, they will be able to plan/organise specific sprints to realize some integration developments.

8.4.2 Hardware description, server configuration, software tools

1.1.1.9 Best practices for collaborative development

1- SVN to support collaborative development

An important guideline to organise the development is the structure of the project itself. Effectively, the entire project will be shared by all the developers through a subversion project. In a distributed approach, all developers will be able to contribute in a collaborative way. But to avoid confusion between the different tasks, and component specificities, the global project should be organised in different sub-projects. By this way, tasks and roles can be clearly dispatched in every sub-project.

2- Maven to manage the automatic build:

Each sub-project can be organised with an automatic build (i.e. Maven). All required libraries and dependencies are configured to simplify the installation, configuration of the different development project. Also, the build could be a very useful approach to follow the principles of continuous integration. The integration is very hard, and problems could increase exponentially with the number of bugs, of components, etc. So, it is better to replace a big integration step by several small and short ones to minimize the integration efforts. This approach is called "Continuous integration" and can be supported by the tool called Hudson.

To summarise the approach, different sub-projects will be initiated in parallel and shared on the subversion server. For example, one for each WP to facilitate the development of very focused software components. One development project can be focused on the architecture and the integration of the WP components. Finally, some projects focused on the prototype development can be also imagined taking into account the "Top-down" strategy or the "Bottom-up" strategy.

1.1.1.10 Development environment

To facilitate the development homogeneity, programming language and tools should also be shared as possible. As many existing components in Khresmoi are Java-based source code, and Java is a used by a very large community of programmers, it will be selected to develop the core of the Khresmoi system. It means that other languages could be used to develop very specific components, but in that case, developers would have to ensure a compliant interface with the whole system.

In terms of tools, Eclipse is a very powerful IDE for java developers. This tool is fully compliant with all Java technologies, open-source and many plugins are available to support development activities. Thus, Eclipse will be recommended to the Khresmoi developers.

Some other tools would be shared in the development team to organise/manage the development tasks, synchronize them, follow the evolution of the source code, support the bug resolution, etc. For example, Tracks and Mantis are tools to plan and follow the evolution of the developments activities.

1- Development tools

Function	**Software / Language**	**Description**
Programming language	Java	1.6.0_xx http://www.oracle.com/technetwork/java/javase/overview/index.html
IDE	Eclipse	3.5 (Galileo) or 3.6 (Helios) or e4 http://www.eclipse.org/e4/
Version control	Subversion available in ATOS SVN	1.6.12 http://subversion.apache.org/
	Subversion Eclipse Plugin Subversive	0.7.9 http://www.eclipse.org/subversive/
Design tool	EPF, eclipse plugin	http://www.eclipse.org/epf/
	Astah Community	http://astah.change-vision.com/en/index.html
Feature and Bug Tracking	Mantis	http://www.mantisbt.org/documentation.php
	Trac + Trac plugins (i.e. Scrum)	http://trac.edgewall.org/ http://trac.edgewall.org/wiki/PluginList
Team and development management	Trac + Trac plugins (i.e. Scrum)	1.6.0_xx

Project Management and Build Automation	Maven	3.0-beta-3
	Maven Eclipse Plugin m2eclipse	0.10.x
Continuous integration	Hudson	http://maven.apache.org/
Quality assurance	Sonar	http://www.sonarsource.org/

Table 21. List of development tools

2- Components, Libraries, Dependencies

Function	**Component / Library**	**Description**
Runtime Environment / Servlet / JSP Server	Jetty	http://www.eclipse.org/jetty/
Service Execution Environment	Tuscany SCA Java	http://tuscany.apache.org/sca-java.html
POJO-based Persistence	Hibernate	http://www.hibernate.org/
XML (De-)Serialization	Simple Framework	http://simple.sourceforge.net/
User Interaction	ezDL	http://www.ezdl.de/
Web Application Framework	Google WTK	http://code.google.com/intl/es-ES/webtoolkit/
	JSF	http://docs.jquery.com/Main_Page
IDE for testing Web Application	Selenium	http://seleniumhq.org/

Table 22. List of development libraries and components

8.5 Risks / Difficulties

Topic	Risk	Level	Responsibility	Solutions / comments
Architecture design	ezDL backend doesn't scale well enough	medium	UDE	Replace relevant architecture or implementation that allows for better scaling
System development	GIFT is not scalable		HES-SO	GIFT is being replaced by a scalable system
Integration	GIFT - MRML language is old and outdated		HES-SO	Binding schema compatible to be addressed in the new GIFT
	MATLAB Java Builder integration stability		ATOS, MUW	Early testing is recommended
Deployment of the system	GIFT features are old and outdated		HES-SO	State-of-the-art features to be used in developed system
	Restrictive hospital policies hinder deployment of search software	?	?	Hospital workstations are generally closed systems
	Deployment of Java Builder on various platforms require OS dependent code		ATOS, MUW	Testing the deployment

Table 23. List of important risks for the integration

9 Conclusions

In this deliverable, we present a summary of the efforts provided in task T6.2 related to "Architecture design". This document is an introduction into the architecture and integration aspects of the Khresmoi system. The described aspects are the result of a specific approach based on several iterations between top-down (components analysis) and bottom-up (user requirements) approaches. Several states of the art permit to ensure the advanced positioning of Khresmoi and select an integration approach that fits with the requirements of the projected Khresmoi system.

The SOA approach offers the best features in terms of flexibility and scalability to be used in Khresmoi. In the list of SOA approaches, the SCA approach provides the most generic specification of software components and the mechanism to compose services. Thus, SCA provides a useful software infrastructure to integrate and deploy various software components as loosely coupled services.

The software architecture was defined from the classical structure proposed by the SOA approach. It is composed of different logical layers: the Application Layer for end-user interaction management, the Core Layer for the main services, and the Persistence Layer dedicated to the storage and data access functionalities. The Core Layer is separated into two complementary systems: the Search system is the front end used by the end-user for searching information, and the Batch system is the back end that is the data processing system in charge of the index production and update. For each logical layer, several functional categories permit to organise the services to be implemented.

Moreover, as the user requirements concern many tasks still in progress (especially WP8 and WP9), the architecture will undergo further refinements to be taken into account in the future.

If the system architecture describes the previous logical structure, the system specification concretely describes the different services of the system. The SCA approach provides the formalism to annotate the software components as part of the whole system. In particular, it provides an XML formalism used to describe the services of the component (i.e. the functionalities provided) and the references (i.e. the components required). This component specification enables a very generic model of software composition and integration that corresponds to the composites specification in the SCA vocabulary.

We can conclude that the SCA approach provides us a very strong and flexible framework for the prototype design and implementation. Several scenarios for the system integration can be defined, and will be converging towards a full software prototype. Moreover, the design considerations already take into account the scalability aspects with the analysis of some solutions proposed for the cloud infrastructure. So, in parallel to the implementation of the first software prototype, the cloud infrastructure will be studied deeply in task T6.3.

Finally, the development guidelines and the tools used for the collaborative development are described. The team and the development responsibilities are well distributed to ensure the realization of the design and development objectives.

10 Bibliography

[1] Momtchev, Vassil; Peychev, Deyan; Primov, Todor; Georgiev, Georgi; Expanding the Pathway and Interaction Knowledge in Linked Life Data, Semantic Web Challenge 2009

[2] Health Care and Life Sciences Interest Group: A Prototype Knowledge Base for the Life Sciences, http://www.w3.org/TR/hcls-kb/

[3] Belleau, François; Nolin, Marc-Alexandre; Tourigny, Nicole; Rigault, Philippe & Morissette, Jean: Bio2RDF: Towards a mashup to build bioinformatics knowledge systems. In: Journal of Biomedical Informatics, Vol. 41, Nr. 5 (2008) , S. 706-716.

[4] Goel, A. (2008). Enterprise Integration EAI vs. SOA vs. ESB.

[5] William A. Ruh, William J. Brown, and Francis X. Maginnis. 2001. Enterprise Application Integration: A Wiley Tech Brief. John Wiley & Sons, Inc., New York, NY, USA.

[6] OASIS Reference Model for Service Oriented Architecture 1.0. http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf, 2006.

[7] OASIS Open CSA – Service Component Architecture (SCA). http://www.oasis-opencsa.org/sca, 2007

[8] Ali Arsanjani, Francisco Curbera, Nirmal Mukhi: Manners Externalize Semantics for On-demand Composition of Context-aware Services. ICWS 2004: 583-590.

[9] Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures, http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[10] SOAP version 1.2, http://www.w3.org/TR/soap/, 2007.

11 Appendix

11.1 General scenario for WP8

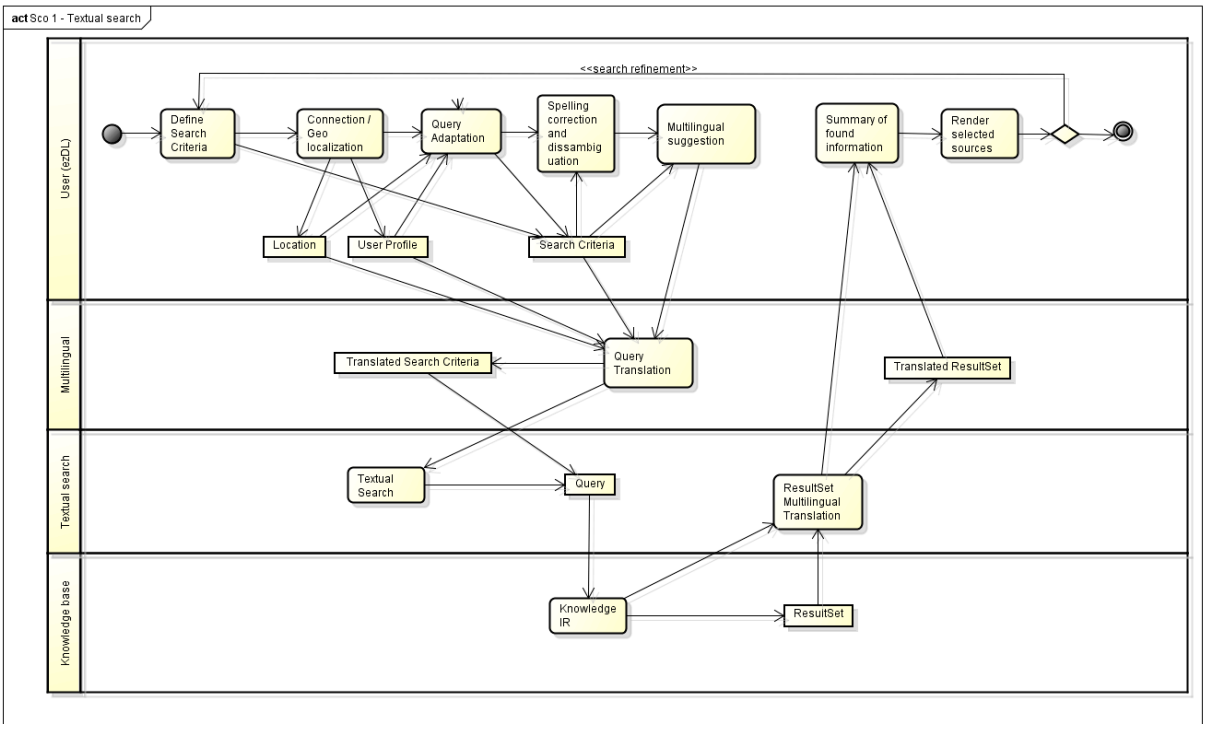
11.2 General scenario for WP9

11.3 Full description of the component involved in the 1st integrated prototype

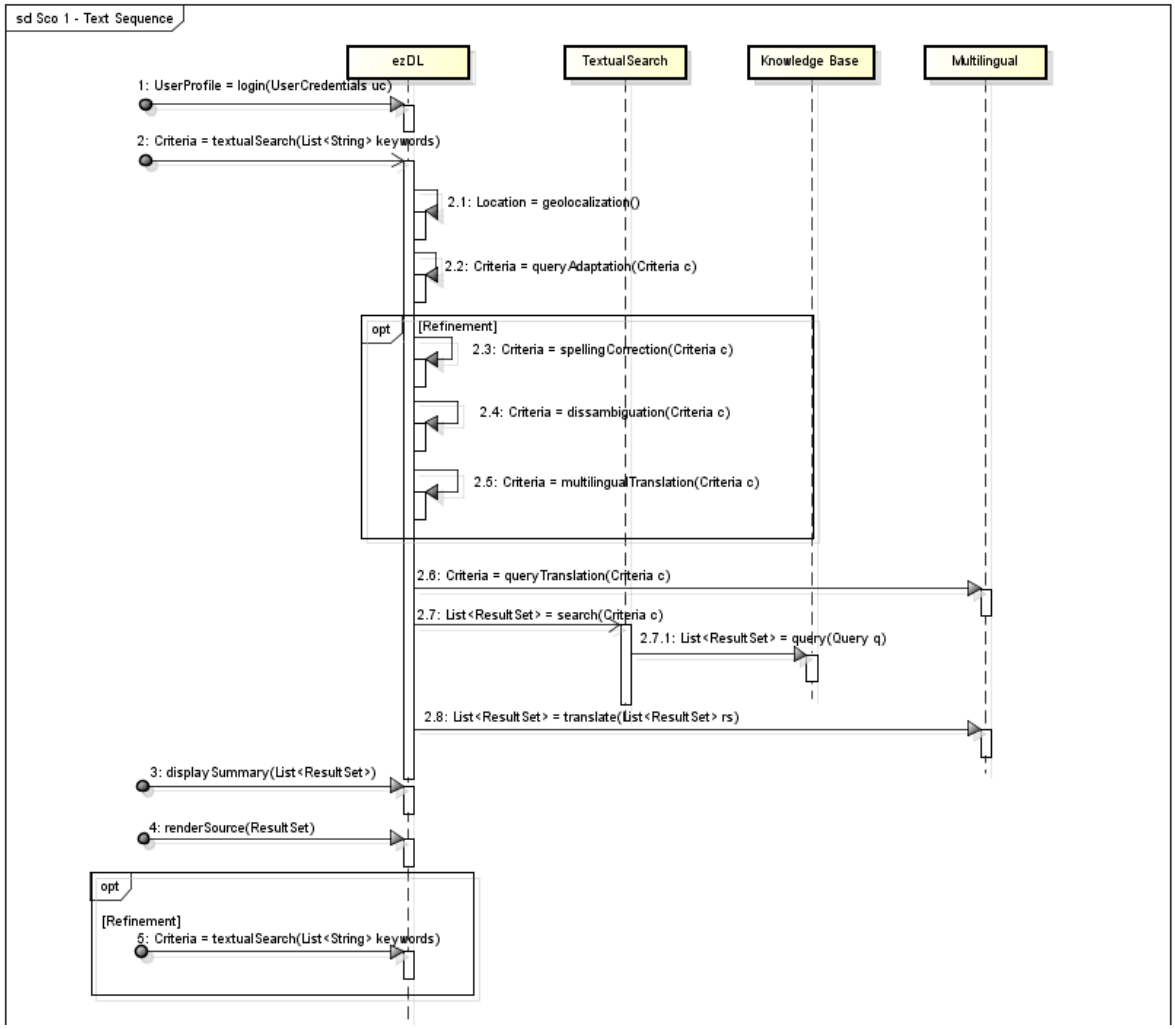
11.1 General scenario for WP8

All relevant templates used for project documentation and reporting are assembled in the wiki. They are updated if in case of changes.

1- Activity Diagram

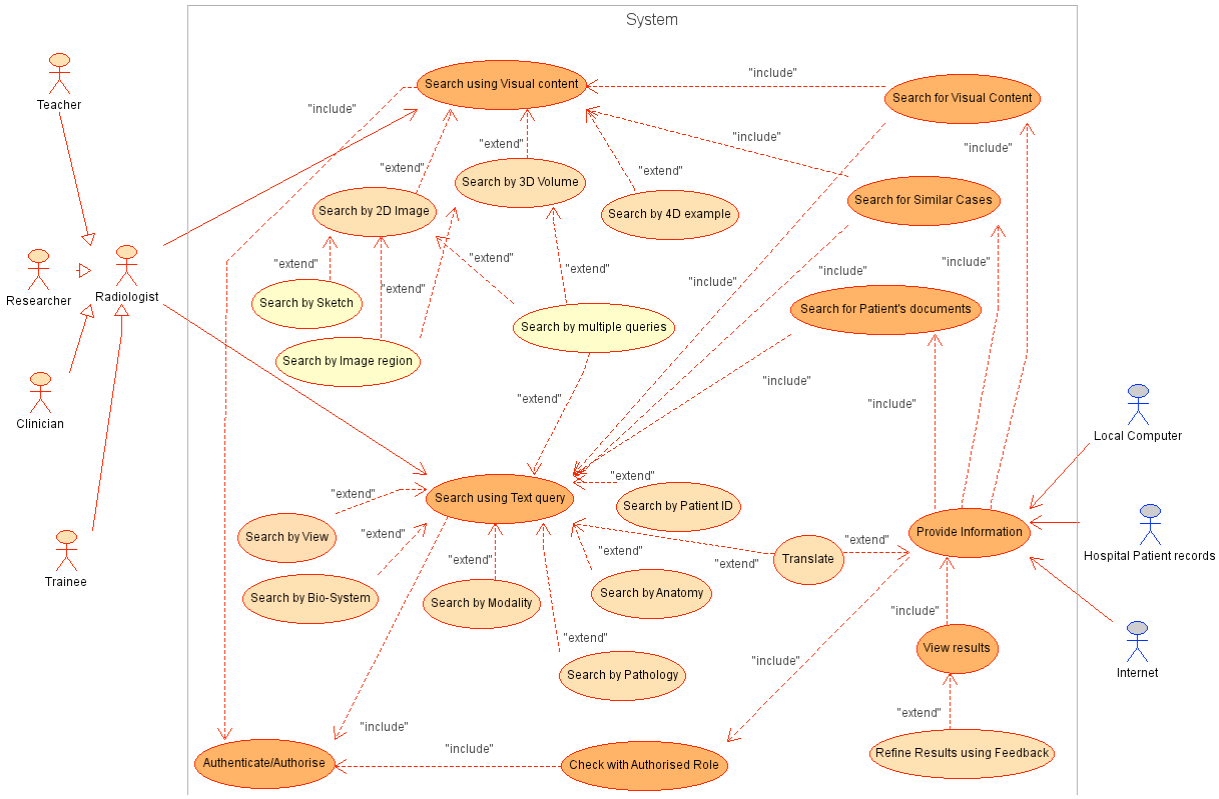


2- Sequences Diagram

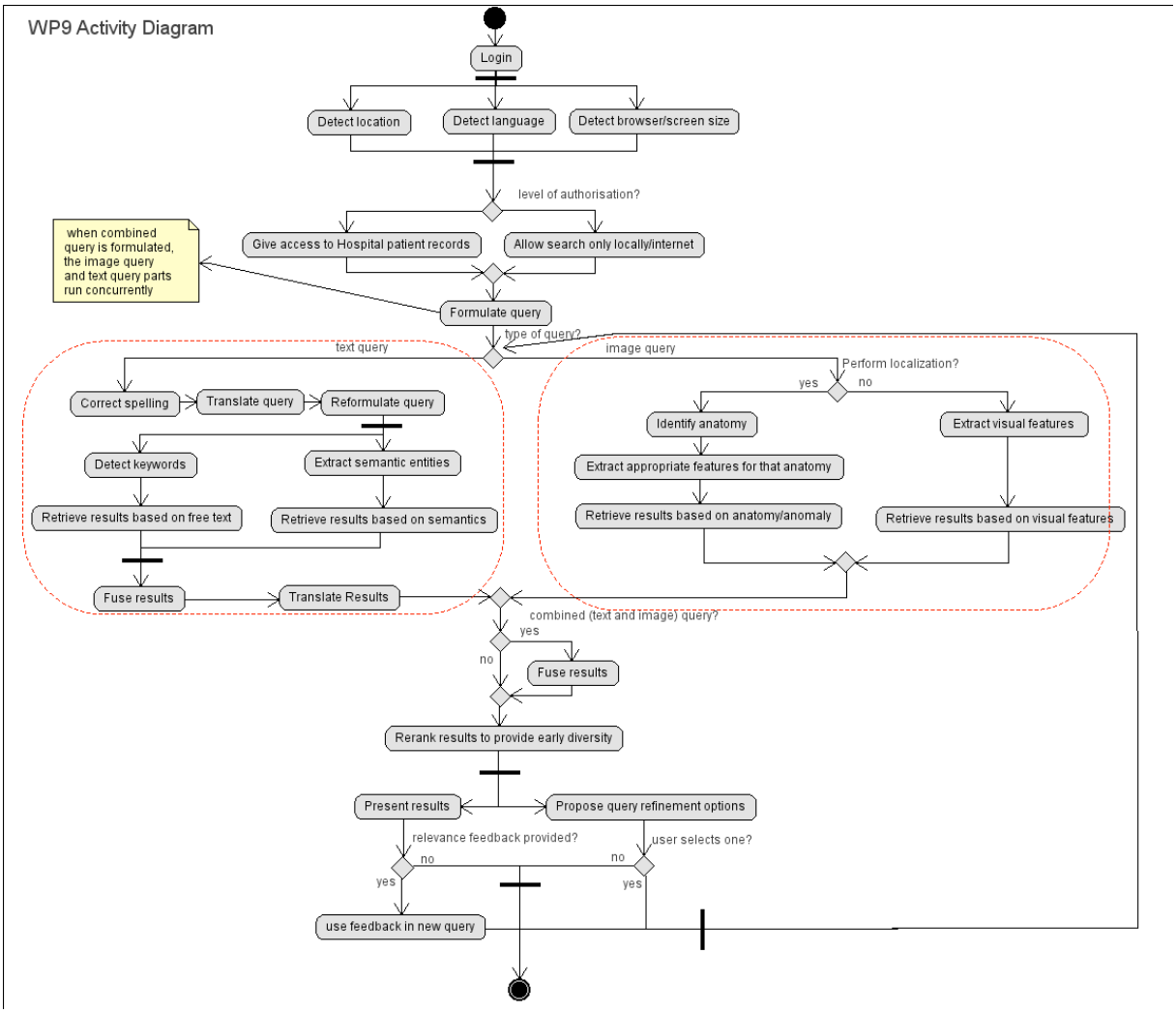


11.2 General scenario for WP9

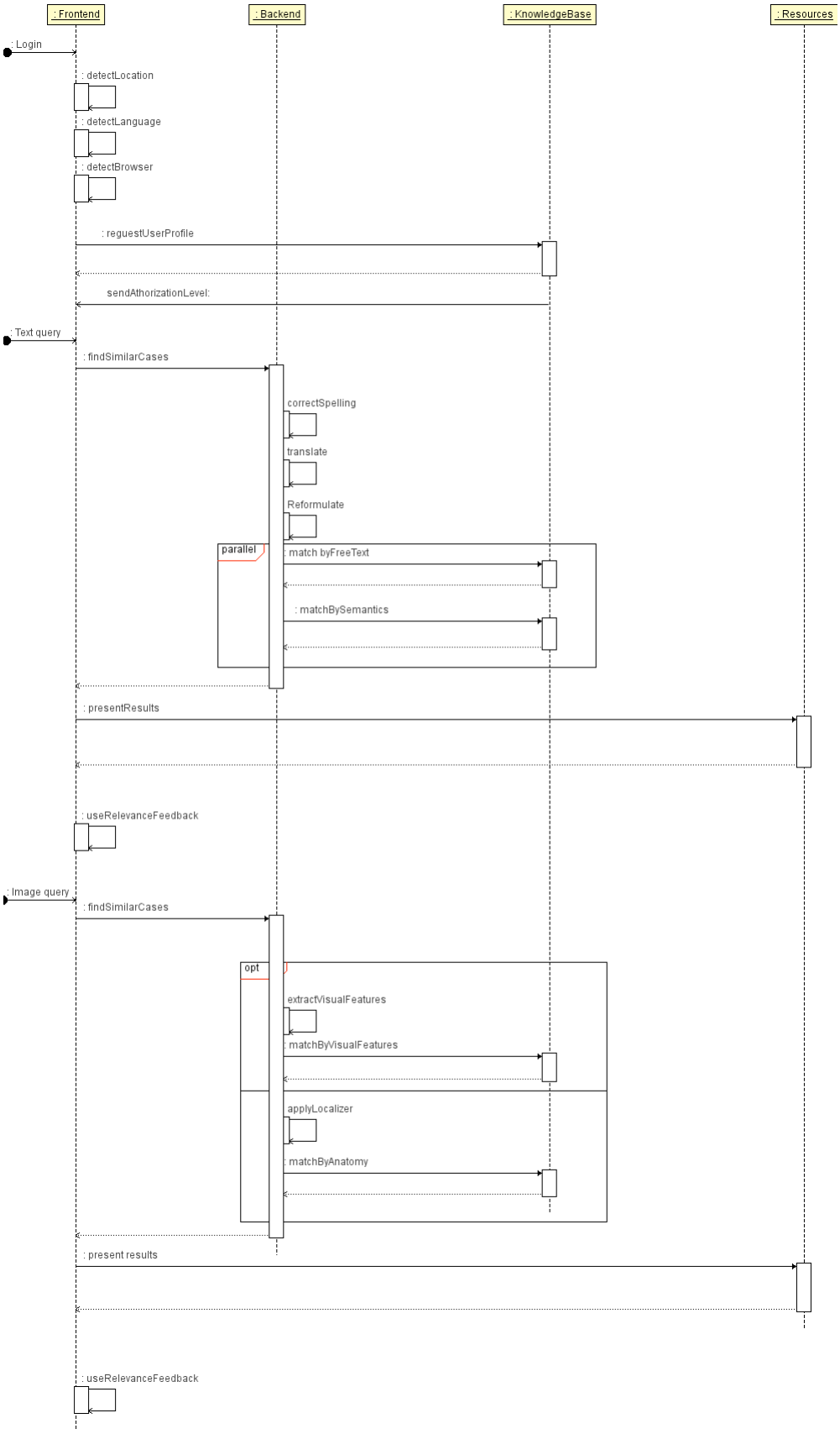
1- Use Case diagram



2- Activity Diagram



3- Sequences Diagram



4- Overview of the required functionalities for WP9

Phase	**Function**	**provided by**
Startup	Location detection	ezDL
	Language detection	ezDL
	Browser detection (i.e. mobile browser)	ezDL
	User authentication (if access to internal data)	
Query entered	spelling correction (if text)	
	obtain image (if visual input)	ezDL
	analysis of mixed data (if document with images and text is submitted)	
	disambiguation (use of terminologies)	HES-SO (SNOMED CT)
	Detect the type of query (looking for disease, anatomic region, symptom, ...)	HES-SO (UMLS semantic types)
	translate query (if text or mixed)	
Retrieval	Adapt faceting to query type	ezDL
	retrieval of similar images	MUW, HES-SO
	retrieve by text	MIMIR, Lucene, HES-SO (PMC, Cochrane and Wikipedia)
	retrieval via a knowledge source	MIMIR
	Anything else?	
	Automatic feedback	HES-SO (Rocchio)
	Thesaurus-driven feedback	HON (e.g. MeSH), HES-SO (e.g. SNOMED CT)
Results display	show images (2D, 3D, ...)	ezDL or external such as YaDiV
	show key images and key text (potentially translated)	?
	filter images without access rights, search in local data	
	allow user to give feedback information (images or keywords or knowledge sources)	ezDL
	Show information to user to allow determining relevance of say an article (like a summary?)	?
	Propose various propositions to refine query	
	Translation of results	
	Extraction of key text passages	HES-SO (n-grams in any input/output and passage retrieval in PMC)
	Multidocument visualization/summarization	HES-SO (GRID-based views)
Batch processing	automatic clustering in very large image spaces	MUW
	return modality based on image example	MUW, HES-SO
	return anatomic region based on an image example	MUW
	text indexation	Lucene, MIMIR
	extract visual features (or visual words)	MUW, HES-SO
	index 3D texture	MUW, HES-SO
	index 2D images	HES
	extract DICOM header information	
	extract other types of metadata of the images	
	storage of all indices and metadata	

11.3 Full description of the components involved in the 1st integrated prototype

No.	Scenario step	Source component	Source interface	Target component	Target interface	Message exchanged	Implementation status
1	Check spelling of user search terms	ezDL	IspellingCheck	Wrapin	IspellingCorrection		Finished
2	Disambiguate query terms	ezDL	IConceptExpansion	BigOWLIM	IDisambiguation		Finished
3.1	Issue spell checked and disambiguated query terms, bound to canned queries containing semantic constraints, so start a new Mimir query	ezDL	IQueryRequest	MIMIR	ITextSearch	postQuery **Parameters:** queryString, the text of the query, using the Mimir query language. **Returns:** An XML message with the ID of the new query, or an error message if there were any problems while parsing the query	Finished
3.2	Obtains the number of hits collected so far. If a query is not com-plete, more hits may be available at later time. If a query has stopped being active before completing, it can be restarted by calling getMoreHits.	ezDL	IQueryRequest	MIMIR	ITextSearch	hitCount **Parameters:** queryId: the ID for the query, as returned by the postQuery action	Finished

No.	Scenario step	Source component	Source interface	Target component	Target interface	Message exchanged	Implementation status
						Returns: An XML message encapsulating a numeric value, or an error message if there were any problems.	
3.3	Obtains the statistics for the documents that have been found so far to contain hits. These include the document IDs and the number of hits for each individual document	ezDL	IQueryRequest	MIMIR	ITextSearch	docStats **Parameters:** queryId: the ID for the query, as returned by the postQuery action. startIndex the first requested document. A value of 0 requests the details for the first document found to contain hits. count the number of documents for which the details are requested. **Returns:** An XML message encapsulating a set of <document> elements, one for each individual document.	Finished
3.4	Obtain a set of hits. Each hit is defined by a document ID, a	ezDL	IQueryRequest	MIMIR	ITextSearch	hits	Finished

No.	Scenario step	Source component	Source interface	Target component	Target interface	Message exchanged	Implementation status
	position and a length, both of which are defined in terms of tokens, not characters					**Parameters:** queryId: the ID for the query, as returned by the postQuery action; startIndex: the first requested hit; count: the number of hits for which the details are requested. **Returns:** An XML message encapsulating a set of <hit> elements, one for each individual hit.	
3.5	Requests a query that has stopped collecting hits before completing to restart collecting hits. If the query has already completed, or is already active, this call will simply be ignored (it will not cause an error).	ezDL	IQueryRequest	MIMIR	ITextSearch	getMoreHits **Parameters:** queryId: the ID for the query, as returned by the postQuery action **Returns:** An XML message reporting success or failure.	Finished
3.6	Check if a query is still working on collecting hits.	ezDL	IQueryRequest	MIMIR	ITextSearch	isActive **Parameters:** queryId: the ID for the query, as returned by the postQuery action	Finished

No.	Scenario step	Source component	Source interface	Target component	Target interface	Message exchanged	Implementation status
						Returns: An XML message encapsulating a boolean value, or an error message if there were any problems.	
3.7	Checks if a query has finished collecting all the hits. If a query is not complete, more hits may be available at a later time. If a query has stopped being active before completing, it can be restarted by calling getMoreHits.	ezDL	IQueryRequest	MIMIR	ITextSearch	isComplete **Parameters:** queryId: the ID for the query, as returned by the postQuery action **Returns:** An XML message encapsulating a boolean value, or an error message if there were any problems.	Finished
3.8	Renders the document text and hits for a given document, in the context of a given query. The HTML of the document is rendered directly to the response stream of the connection.	ezDL	IQueryRequest	MIMIR	ITextSearch	renderDocument **Parameter:** queryId: the ID for the query, as returned by the postQuery action. documentId the ID for the requested document, as returned by e.g. a call to the docStats action.	Finished

No.	Scenario step	Source component	Source interface	Target component	Target interface	Message exchanged	Implementation status
						Returns: HTML content. The hits are rendered as ...	
3.9	Returns the title and URI associated with a document. These values were provided at indexing time.	ezDL	IQueryRequest	MIMIR	ITextSearch	documentMetadata **Parameter:** queryId: the ID for the query that has returned the document ID being used, as returned by the postQuery action. Returns documentId the ID for the requested document, as returned by e.g. a call to the docStats action. **Returns:** An XML message encapsulating the two string values, or an error message if there were any problems.	Finished
3.10	Returns arbitrary metadata associated with a document. These values were provided at indexing time.	ezDL	IQueryRequest	MIMIR	ITextSearch	documentMetadata **Parameter:** queryId: the ID for the query that has returned the document ID	Not finished, to be exposed in Mimir interface

No.	Scenario step	Source component	Source interface	Target component	Target interface	Message exchanged	Implementation status
						being used, as returned by the postQuery action. documentId the ID for the requested document, as returned by e.g. a call to the docStats action. keys: a list of metadata keys for which values are required **Returns:** An XML message encapsulating the values of the parameter keys as given by the document metadata.	
3.11	Obtain a segment of the text of a document.	ezDL	IQueryRequest	MIMIR	ITextSearch	documentText **Parameter:** queryId: the ID for the query that has returned the document ID being used, as returned by the postQuery action. documentId the ID for the requested document, as returned by e.g. a call to the docStats action. position the position of the first returned token.	Not finished

No.	Scenario step	Source component	Source interface	Target component	Target interface	Message exchanged	Implementation status
						This parameter is optional; defaults to 0 is not provided, which means the first token of the document. length the number of tokens to be returned. This parameter is optional, if omitted, all the document tokens will be returned. **Returns:** An XML message containing the text of all the individual tokens and, if available, the spaces between them.	
3.12	Closes a query, releasing all resources allocated for supporting it.	ezDL	IQueryRequest	MIMIR	ITextSearch	close **Parameter:** queryId: the ID for the query, as returned by the postQuery action. **Returns:** An XML message with a success or failure value	Finished
	Split the text query into	Mimir	ISPARQLQuery	BigOWLIM	ISPARQLEndpoint	The SPARQL query as	Finished

No.	Scenario step	Source component	Source interface	Target component	Target interface	Message exchanged	Implementation status
	component parts, and send any semantic constraints to OWLIM as a SPARQL query.					string; optional arguments are LIMIT(Integer), IMPLICIT(Boolean) and JSON(Boolean); Possible credentials required Returns XML or JSON BindingResults	
	Fetch type of query for a given user query, based on statistics about common query types. Used for facetting of search results and and for transferring user query into appropriate Mimir queries.	ezDL	IQueryType	Wrapin	ITypeProvided		Not finished