

PANACEA Project

Grant Agreement no.: 248064

Platform for Automatic, Normalized Annotation and
Cost-Effective Acquisition
of Language Resources for Human Language Technologies

D3.1

Architecture and design of the platform

Dissemination Level: Public
Delivery Date: July 16th 2010
Status – Version: Final
Author(s) and Affiliation: Marc Poch (UPF), Prokopis Prokopidis(ILSP), Gregor Thurmair (Linguattec), Carsten Schnober, (ELDA), Riccardo Del Gratta (ILC-CNR), Núria Bel (UPF), Olivier Hamon (ELDA)

1	Introduction	1
2	Terminology	3
2.1	Definitions	3
2.2	Acronyms	4
3	Goals.....	5
4	Current state of the art. Analysis of existing tendencies, approaches and tools	6
4.1	Frameworks	6
4.1.1	UIMA	6
4.1.1.1	U-compare	8
4.1.2	GATE	9
4.1.3	Concluding remarks. Comparative analysis and recommendations.	10
4.2	Web Services.....	11
4.2.1	WSDL.....	11
4.2.2	SOAP.....	13
4.2.3	REST	16
4.2.4	AXIS.....	17
4.2.5	Common interfaces and tool integration	17
4.2.6	Metadata / Ontology / Closed vocabularies.....	18
4.2.7	Concluding remarks. Comparative analysis and recommendations.	19
4.3	Workflow systems.....	19
4.3.1	Triana.....	20
4.3.2	Kepler	22
4.3.3	Taverna (myGrid).....	25
4.3.4	LoonyBin.....	29
4.3.5	Concluding remarks. Comparative analysis and recommendations.	31
4.4	Grid infrastructure	32
4.4.1	Globus	32
4.4.2	EGEE.....	34
4.4.3	MyGrid	37
4.4.4	TextGrid	40
4.4.5	NorduGrid	41
4.4.6	Concluding remarks. Comparative analysis and recommendations.	42
4.5	The Registry	43
4.5.1	UDDI.....	43
4.5.2	Feta	43
4.5.3	BioCatalogue	44

4.5.4	Concluding remarks. Comparative analysis and recommendations.	45
4.6	Wrappers	46
4.6.1	Soaplab	46
4.6.2	Concluding remarks. Comparative analysis and recommendations.	49
4.7	Sharing research objects (Workflows, ontology, etc.).....	49
4.7.1	myExperiment	49
4.7.2	Concluding remarks. Comparative analysis and recommendations.	50
4.8	Relevant projects	51
4.8.1	KYOTO project	51
4.8.1.1	The Kyoto Annotation Framework (KAF).....	52
4.8.2	ACCURAT	54
5	PANACEA Platform requirements	55
6	PANACEA Platform design.....	55
6.1	Travelling object. Corpus and data format	55
6.1.1	Introduction	55
6.1.2	Crawling and boilerplate removal	56
6.1.3	Text processing.....	61
6.1.3.1	Sentence splitting and tokenization	61
6.1.3.2	POS Tagging and lemmatization.....	62
6.1.3.3	Constituency and/or dependency parsing	64
6.1.4	Alignment.....	67
6.1.5	Revision and distribution metadata	69
6.2	Common interfaces design	69
6.3	PANACEA Platform technologies	71
6.3.1	Options	71
6.3.1.1	Option 1: MyGrid environment.....	71
6.3.1.2	Option 2: Large scale Grids.....	73
6.3.1.3	Option 3: Fallback position: UIMA / GATE.....	73
6.3.2	Primary option, future research and alternatives	73
7	Workplan.....	75
7.1	To-do list	75
7.1.1	Travelling Object.....	75
7.1.2	Web Services	75
7.1.2.1	General	75

7.1.2.2	Common Interfaces	75
7.1.2.3	Soaplab	76
7.1.2.4	AXIS.....	76
7.1.2.5	Alternatives.....	77
7.1.3	Workflow editor and engine.....	77
7.1.3.1	Workflow Editor.....	77
7.1.4	Registry	77
7.1.4.1	General	77
7.1.4.2	MyBioCatalogue > PanaceaCatalogue	78
7.1.5	Portal	78
7.1.6	Tools.....	78
7.1.6.1	Work Package 4 tools.....	78
7.1.6.2	Work Package 5 tools.....	79
7.1.6.3	Work Package 6 tools.....	79
7.1.6.4	Travelling object improvements.....	79
7.1.7	Other technologies and alternatives. Fallback strategies.....	79
7.1.7.1	Surveys	79
7.1.7.2	Tests.....	80
7.1.7.3	Workplan changes	80
7.2	Workplan table	80
7.2.1	Resources.....	82
8	Bibliography	83
9	Appendix	85
9.1	Appendix A	85
9.1.1	Current state of the art analysis schema	85
9.2	Appendix B	86
9.2.1	Emboss Groups.....	86
9.2.2	Soaplab test.....	87
9.2.3	Common Interfaces proposal.....	91
9.2.4	Semantic Service Description (myGrid way).....	96
9.2.5	Biocatalogue web user interface.....	99
9.2.6	myExperiment	100

9.2.7	Common interfaces design	101
9.2.7.1	Sentence Splitting.....	101
9.2.7.2	Crawling	102
9.2.7.3	Tokenization	103
9.2.7.4	Named Entity Recognition	104
9.2.7.5	Lemmatization.....	105
9.2.7.6	PoS tagging.....	106
9.2.7.7	Alignment	107
9.2.7.8	Parsing	108
9.2.7.9	Term Extraction.....	109
9.2.7.10	Topic Identification	111

1 Introduction

Multilingualism in Europe today represents a challenge for Machine Translation (MT) systems which are expected to break language barriers for millions of citizens. These systems require massive amounts of clean parallel data for every pair of languages and for every domain they are to be implemented. Moreover, all these Language Resources (LR) have to be updated regularly since languages change continuously.

Supplying all these resources to the MT systems represents a critical point for their implementation and it is mostly done by hand nowadays. PANACEA aims to create a LR factory which automates the steps to create LR reducing the actual necessary time and cost.

PANACEA can be seen as a factory for the creation of a variety of LRs. In this factory the raw material is captured by corpus creation methods (crawling the web, accessing to archived texts, or local documents of the user computer). Then this material is cleaned, and processed in order to create derivatives: annotated corpus, parallel corpus and annotated parallel corpus. These first derivatives are later used to produce, by means of induction tools, a second order or synthesized derivatives: rich lexica (with morphological, syntactic and lexico-semantic information) and bilingual dictionaries (word and multiword based) and transfer grammars. The factory should also make available repositories to merge new with old resources as well as validation tools for each of the LR's derivatives.

The LR factory will be an interoperability platform of components creating complex workflows which can reproduce the step-by-step process of creating LR. Combining different tools makes interoperability a critical issue for the platform. To this aim standard technologies and protocols had been surveyed and chosen.

The factory is, in sum, an interoperability space where the functional integration of a variety of components can be made for them to work in a chain. Each component (system or tools that perform operations to produce LR's as supplied by WP4, WP5 and WP6) will be integrated into the platform. By integrated we mean that each component has to be wrapped as an independent web service that will later be chained into particular workflows that have as objective the production of the different resources. All components will be deployed as web services using common interfaces as way to ensure interoperability. Specific aspects that must be covered are:

- In order to guarantee interoperability, input sent to a web service must comply with the specifications declared in the description of the web service itself. These descriptions are declared as metadata in xml format that inform possible users about the characteristics both of the input and of the output of every particular web service. In order to maximize interchangeability of components that perform the same function, PANACEA will propose the declaration and use of common interfaces. Thus every component delivered by WP4 to WP6 will be deployed as a web service that will wrap it under a defined common interface.
- A number of components (and middleware) will be involved in an interoperable space, i.e. a platform, in addition to those already mentioned above. The links between components are shown in Figure 1.
- The workflow engine, working as a centralized server that handles the processing chain;
- The user interface(s) and workflow editor, used to configure, run the processing chains and access the results;

sustainability etc. Some technologies are chosen as the primary option to develop the platform while others will be watched closely and used if it's worth.

2 Terminology

2.1 Definitions

AAI [Stanica 2006]

Authentication and Authorization infrastructure

An infrastructure that provides Authentication and Authorization Services. The minimum service components include Identity and Privilege Management with respect to users and resources.

Factory

The set of the platform and the NLP tools used to produce LR.

Metadata [Guenther 2004]

Structured information that describes, explains, locates, and otherwise makes it easier to retrieve and use an information resource.

Metadata registry [Guenther 2004]

A formal system for the documentation of the element sets, descriptions, semantics, and syntax of one or more metadata schemes.

Platform

The set of tools (registry, workflow editor, etc.), software, documentation (closed vocabularies, format definitions, etc.), which combined define the PANACEA interoperability space. The NLP tools used as web services are not considered to be part of the platform.

Provenance data

Information that provides a traceable record of the origin and source of a resource

Registry

Repository focused on the needs of SOA (defined below) environments typically used to publish, search and retrieve a wide variety of technical documents and information as WSDL location, documentation, schemas, service descriptions, business process design models, policy documents and so on.

Resource [Berners-Lee 2005]

The term "resource" is used in a general sense for whatever might be identified by a URI. Familiar examples include an electronic document, an image, a source of information with a consistent purpose (e.g., "today's weather report for Los Angeles"), a service (e.g., an HTTP-to-SMS gateway), and a collection of other resources. A resource is not necessarily accessible via the Internet; e.g., human beings, corporations, and bound books in a library can also be resources. Likewise, abstract concepts can be resources, such as the operators and operands of a mathematical equation, the types of a relationship (e.g., "parent" or "employee"), or numeric values (e.g., zero, one, and infinity).

Repository [CiTER]

Facility that provides reliable access to managed digital resources.

SOA [Mackenzie 2006]

Service Oriented architecture

A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.

SP [Stanica 2006]

Service provider

An entity that provides access to a service.

Web service [Brown 2004]

A web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format.

Workflow [Wulong 2001]

Workflow is a term used to describe the tasks, procedural steps, organizations or people involved, required input and output information, and tools needed for each step in a business process.

2.2 Acronyms

Reference	Abbreviation of	Link
[CERN]	Conseil Européen pour la Recherche Nucléaire	www.cern.ch
[CLARIN]	Common Language and Technology Infrastructure	http://www.clarin.eu
[CLI]	Command Line Interface	
[DAS]	Distributed annotation system	http://www.biodas.org/documents/spec-1.53.html
[ebXML]	e-business XML	http://www.ebxml.org/
[EGEE]	Enabling Grids for E-sciencE	http://www.eu-egee.org/
[EGI]	European Grid Initiative	http://web.eu-egi.eu/
[GATE]	The General Architecture for Text Engineering	http://gate.ac.uk
[GGF]	Global Grid Forum	http://www.gridforum.org/
[GSI]	Grid Security Infrastructure	
[ISocat]		http://www.isocat.org

[LAF]	Linguistic Annotation Framework	
[LHC]	Large Hadron Collider	
[LMF]	Lexical Markup Framework	http://www.lexicalmarkupframework.org
[MAF]	Morphosyntactic Annotation Framework	
[OASIS]	Organization for the Advancement of Structured Information Standards	http://www.oasis-open.org/
[OGSA]	Open Grid Services Architecture	http://www.globus.org/ogsa/
[OWL]	Semantic Markup for Web Services	http://www.w3.org/Submission/OWL-S/
[REST]	Representational State Transfer	http://www.ics.uci.edu/~fielding/pubs/dissertation/ rest_arch_style.htm
[SAWSDL]	Semantic Annotations for WSDL and XML Schema	http://www.w3.org/TR/sawSDL/
[SCHEMAS]		http://www.schemas-forum.org/
[SHIBBOLETH]	Shibboleth	http://shibboleth.internet2.edu/
[SOAP]	Simple Object Access Protocol	
[UDDI]	Universal Description, Discovery and Integration	http://www.oasis-open.org/
[UIMA]	Unstructured Information Management Architecture	http://incubator.apache.org/uima
[WLCG]	Worldwide LHC Computing Grid Project	
[WSDL]	Web Service Description Language	www.w3.org/TR/wsdl
[WSRL]	Web Services Resource Framework	http://www.globus.org/wsrf
[XML]	Extensible Markup Language	

3 Goals

This document aims to establish the requirements and the technological basis and design of the PANACEA platform. These are the main goals of the document:

- Survey the different technological approaches that can be used in PANACEA.
- Specify some guidelines for the metadata.
- Establish the requirements for the platform.

-
- Make a Common Interface proposal for the tools.
 - Propose a format for the data to be exchanged by the tools (Travelling Object).
 - Choose the technologies that will be used to develop the platform.
 - Propose a workplan.

4 Current state of the art. Analysis of existing tendencies, approaches and tools

The following section is devoted to survey the current state of the art in technologies and tools that can be relevant for the development of the PANACEA platform. The section is divided as follows:

- **Frameworks:** A software framework, in computer programming, is an abstraction in which common code or tools providing generic functionality can be selectively specialized by user code providing specific functionality. The overall program's flow of control is not dictated by the user.
- **Web Services:** Web services are typically application programming interfaces (API) or web APIs that are accessed via Hypertext Transfer Protocol and executed on a remote system hosting the requested services. PANACEA platform needs web services to remotely access the NLP tools.
- **Workflow editors / engines:** A workflow editor is a tool used to design workflows. A workflow engine is a program which is able to execute workflows. Usually a workflow editor is an engine too.
- **Grid infrastructures:** A Grid is a new distributed computing infrastructure that interconnects heterogeneous resources using wide area networks like the internet. These are complete solutions with storage systems, web services, workflows, a registry, etc. The whole PANACEA platform could be designed using a Grid.
- **The Registry:** A registry is a repository to list web services. PANACEA platform needs a repository to list the web services that can be used.
- **Wrappers:** A wrapper is a set of tools which let a user with a few or none programming skills easily deploy some tools as a web service. A wrapper could be useful for PANACEA service providers.
- **Sharing Research objects:** This section is about existing portals to facilitate sharing workflows and other relevant documents between users. PANACEA users could share information and files using a portal.
- **Relevant projects:** This section is devoted to talk about projects which used some technologies or developed some formats that could be helpful for PANACEA.
- **Travelling object. Corpus and data format:** This section aims to describe existing data formats that could be used in PANACEA.

4.1 Frameworks

4.1.1 UIMA

The Unstructured Information Management Architecture or UIMA is an open, scalable and extensible platform for the development, integration and deployment of applications that aim at analyzing large

volumes of unstructured information contained in text, video and audio¹. Although UIMA begun as an IBM project, an open source reference implementation of the UIMA specification is currently available as an open source project² under the Apache Software Foundation family of projects. Apache UIMA is offered under an Apache License, thus allowing its use for both proprietary and open/free applications.

UIMA applications usually integrate in a chain one or more components for specific tasks like, for example, "sentence and token boundary detection" => "POS tagging" => "lemmatization" => "named entity detection". The components implement interfaces defined by the framework and are described in XML descriptor files, while the UIMA framework manages the flow of (annotated) data between the components. The frameworks are available for both Java and C++, with the Java Framework supporting running both Java and non-Java components (using the C++ framework). Another framework, the UIMA Asynchronous Scaleout Framework provides scale out capabilities to the Java framework via JMS (Java Messaging Services).

A UIMA component that analyzes artifacts (e.g. documents) and generates annotations is called an Analysis Engine (AE). Analysis results from an AE produce are represented by typed Feature Structures which refer to a span of the text under analysis. For example, an annotation over the span of text "Haiti" can have the type Location. A Dependency annotation for the same span can be accompanied by the value Subject for the attribute Label, and by an integer value for the attribute Head.

An XML file called a Type System Descriptor defines the Feature Structure types that can be generated by an AE. UIMA utilities will automatically generate Java classes corresponding to the types that are defined in the Type System Descriptor. In the example feature structure above, one would use the *getHead()* method of the Dependency class to get the integer representing the token's head. The annotations are stored in the Common Analysis Structure (CAS) which is used for communication of annotations between UIMA AEs and/or applications. Special AEs called CAS Consumers can be used to serialize CAS's to different formats.

Although the Apache-UIMA site provides information on making analysis results available as REST web services, the main documentation efforts focus on deployment solutions using the UIMA Asynchronous Scaleout Client - Service Architecture, in which a custom UIMA client application accesses one or more instances of a component or a processing pipeline service.

A relatively small number of tools have been ported to UIMA by the project's community. These include a Tokenizer tool, a HMM POS tagger and a Lucene CAS Indexer. Other organizations like the BioNLP UIMA Component Repository³ and the JULIE Lab⁴ offer open source and extensible NLP tools and Type Systems for several annotation tasks. U-Compare is a closed-source system that allows on-line creation of workflows based on existing UIMA components (see 4.1.1.1 for a description of U-Compare).

¹ In this document, we focus on UIMA applications for analysis of text.

² <http://incubator.apache.org/uima>

³ <http://bionlp-uima.sourceforge.net/>

⁴ http://www.julielab.de/Resources/Software/NLP_Tools.html

UIMA has been approved by OASIS, the international open standards consortium, as an OASIS Standard. The Apache UIMA implementation has reached its 2.3 version and has recently (18 March 2010) graduated from an incubation phase to become a top level Apache project. Documentation is provided in the form of a set of well-written technical documents, while support and advice by the main contributors and the community in the related mailing lists is generally quick and informative.

Depending on the point of view, weaknesses may include

- large, often daunting, learning curve and time investment needed to get accustomed to the framework
- bias towards the Java framework in community discussions and documentation
- prior knowledge of Eclipse and Java a must
- GUIs for editing XML descriptors for components available only for Eclipse; no support for other IDEs
- not that many tools already ported as Analysis Engines
- lack of a framework like GATE's JAPE for developing rule-based analysis engines

4.1.1.1 U-compare

U-Compare⁵ (Kano et al., 2009a) is a system based on UIMA but aims to provide a platform that is easier to use and allows users to make visualizations and comparison between the tools. It uses the UIMA tools for text mining and natural language processing. U-Compare contains its own graphical user interface and workflow editor which can be used to create workflows to be used with UIMA and shared with other users. Several components can be run in parallel and their respective results can be compared with each other and evaluated.

U-Compare is a joint project between the University of Tokyo, the Center for Computational Pharmacology (CCP) at the University of Colorado Health Science Center, and the National Centre for Text Mining (NaCTeM) at the University of Manchester. The platform itself, i.e. the U-Compare type system, is released under the Apache license and is free for research purposes, but cannot be re-distributed or re-used if any changes have been done to its code. The comparison generator, the GUI and the other shared parts can be used freely for research, too, but these components must not be re-distributed or re-used.

U-Compare provides a number of corpus readers (Biological Entity Annotated Corpora, Biological Event Annotated Corpora, BIO, XMI, and plain text) and writers (XMI, Inline XML, Annotation Printer, BIO), syntactic tools (sentence splitters, tokenizers, POS taggers, lemmatizers, CFG parsers, dependency parsers, and deep parsers), semantic tools (named entity recognizers, biological event recognizers, an abbreviation detector). These are UIMA tools that have been integrated into U-Compare. Additionally, the U-Compare Annotation Viewer to visualise annotation instances, MoriV to visualise HPSG feature structures and CFG tree structures, and Annotation Comparator are integrated into the system.

Additional UIMA compatible components can be used with U-Compare with little effort. Other components can be wrapped as well by making them compatible with the U-Compare type system (Kano

5 <http://u-compare.org>

et al., 2009b). In order to create compatible interfaces, a new UIMA Aggregate Analysis Engine component needs to be created that comprises three components: a type systems converter to convert the U-Compare output to the input format required by the tool, the original component, and another converter that converts the new component's output into the U-Compare type system. It can then be deployed as a UIMA Soap web service, thus providing a UIMA Soap service description file which can be used to create a component comprising the web service only. This will then be compatible to the U-Compare type system. This is the only possible way to make non-Java components U-Compare type system compatible, while Java components can also be deployed as .jar files.

Version 1.1.4 has been released in March 2010. Guides for both using and developing the system are available on the project homepage. The first version was published in 2008; active development was done in 2009 for the participation in BioNLP 2009.

Potential weaknesses:

- closed source system
- not under active development
- Development documentation is not complete

4.1.2 GATE

The General Architecture for Text Engineering or GATE⁶ is an open source, extensible Java platform for the development and integration of NLP applications. The GATE project started in 1995 and is being maintained by a core team of developers associated with the NLP Research Group of the University of Sheffield. The core GATE software is licensed under the Gnu Library General Public License (LGPL), which basically allows the use of a library for the development of both open/free and/or proprietary applications, as long as the source of the library is distributed along with these applications.

The main concepts in GATE terminology are LRs, Processing Resources, Data Stores, and Applications. LRs are data-only resources like corpora, lexicons, thesauri or ontologies. PRs are software resources which often include LRs and whose typical purpose is to process documents and create annotations. A DS is where (annotated) corpora are stored for efficient reuse and reprocessing. An Application is a saved state of a configuration including PR and LRs.

The core GATE software includes

- the GATE Developer, which is an IDE for aggregating PRs, running applications on document collections. The Developer also offers facilities for benchmarking and evaluation of results produced by PRs, vs. gold annotations that can be created manually with the IDE
- the GATE Embedded library, which allows any Java application to process documents using GATE Applications

The GATE platform is distributed with a large set of NLP components for several languages, while, for English, a ready-to-be-used information extraction system called ANNIE is also included. Several rule-based GATE components take advantage of Java Annotation Pattern Engine or JAPE, which allows development of finite-state transducers over annotations based on regular expressions.

⁶ <http://gate.ac.uk>

As of today, GATE is at its 5.1 release. The core documentation is the User Guide, a book providing information for different GATE users (component developers, grammar writers, annotators) on all aspects of the framework. A set of tutorials and screen casts is also available. Support and advice by the main contributors in the related mailing lists is quick and informative.

Depending on the point of view, weaknesses may include:

- no automatic code generation for accessing Annotation attribute values
- no GUI for editing descriptor files for components and aggregated applications
- not enough documentation on robust, error-tolerant deployment of components as web services

4.1.3 Concluding remarks. Comparative analysis and recommendations.

The following table shows the main differences between the UIMA and GATE frameworks.

Table 1: UIMA / GATE comparison

	UIMA	GATE
Functionalities	Annotation type system; automatic class generation for all annotation types; GUI tools for creating and editing descriptors for configuration, integration and deployment of components; robust, error-tolerant, distributed deployment of components;	Typeless annotation schema; GUI for manual annotation of documents and evaluation of automatic annotation; large set of NLP components already available
Integration	Java/OS independent, C++ version of the framework available	Java/OS independent
Maturity	Mature, actively maintained	Mature, actively maintained
Support and plans for the future	Active developer and user community; bugs and issues can be reported in the project's tracker;	Active developer and user community; bugs and issues can be reported in the project's tracker; training seminars regularly organized. Evolving technologies include GATE Teamware, a web-based management platform for collaborative annotation; and GATE Cloud, a parallel distributed processing engine that combines GATE embedded with a heavily optimized service infrastructure running on supercomputer hardware
Availability	Open-source	Open-source

These two solutions are both valid options that could be used as fallback positions for the PANACEA project. Considering that some partners have experience with UIMA that would be the chosen solution.

4.2 Web Services

Web services (sometimes called *application services*) are services (usually including some combination of programming and data, but possibly including human resources as well) that are made available from a Web Server for Web users or other Web-connected programs.

Regarding the PANACEA project, web services have to be the basic processing element inside the factory. Web services technology will allow the remote execution of tools in the PANACEA platform. Web services can be invoked alone or combined in workflows.

This section is devoted to study some existing web service technologies, formats and protocols:

- Web Service Description Language [WSDL]: it's an XML format used to describe a web service that can be understood by computers.
- Simple Object Access Protocol [SOAP]: SOAP is a protocol used to access web services.
- REST: is a development style for distributes media based on HTTP. RESTful web services are developed following the REST style.
- AXIS: a web service framework for SOAP.
- Common interfaces and tool integration.
- Metadata / Ontology / Closed Vocabularies.

4.2.1 WSDL

Web Services Description Language [WSDL] is an XML format that is used to describe web service interfaces. It describes the functions and data formats (messages) in an abstract manner. This results in a set of reusable bindings which are subsequently bound to concrete network protocols and message formats. SOAP (Simple Object Access Protocol) and XML schema are usually used to define web services over the web.

Every function of the web service is described using XML in the WSDL file. Each function description contains the types of its arguments and return values formalized with XML Schemas. These types can be simple or complex. The end-point (the physical address of the web server hosting the service) and the URI of the service which is a unique identifier with which the service is associated (the HTTP request is formed by issuing a POST or GET HTTP operation to the end-point asking for a function along with the URI) are included in the service description inside the WSDL.

In the following example (taken from the WSDL 2.0 primer of W3C, <http://www.w3.org/TR/2007/REC-wsdl20-primer-20070626/#basic-example>)

```
<types>
  <xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://greath.example.com/2004/schemas/resSvc"
```

```

    xmlns="http://greath.example.com/2004/schemas/resSvc">
    <xs:element name="checkAvailability" type="tCheckAvailability"/>
    <xs:complexType name="tCheckAvailability">
      <xs:sequence>
        <xs:element name="checkInDate" type="xs:date"/>
        <xs:element name="checkOutDate" type="xs:date"/>
        <xs:element name="roomType" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
    <xs:element name="checkAvailabilityResponse" type="xs:double"/>
    <xs:element name="invalidDataError" type="xs:string"/>
  </xs:schema>
</types>
<interface name="reservationInterface">
  <fault name="invalidDataFault"
    element="ghns:invalidDataError"/>
  <operation name="opCheckAvailability"
    pattern="http://www.w3.org/ns/wsd/in-out"
    style="http://www.w3.org/ns/wsd/style/iri"
    wsdlx:safe="true">
    <input messageLabel="In"
      element="ghns:checkAvailability" />
    <output messageLabel="Out"
      element="ghns:checkAvailabilityResponse" />
    <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
  </operation>
</interface>

```

the function *opCheckAvailability* has as parameters the XML Schema complex type *tCheckAvailability* that is composed of two dates and one string and that it returns one double real number.

WSDL does not specify a semantic of the operation(s) it describes. The user of a web service must know the significance of the result one function returns. For instance, a WSDL file describes two functions *f1* and *f2*, both of which require two float numbers as arguments and are returning float. Now,

let's suppose that the first one is taking the sum of its arguments and the second one divides its arguments. If the user does not recognize the type of operation either by reading the function name or by reading the documentation the creator of the function has been kind to offer, he will never know what makes the *f1* and *f2* different.

One other underspecified element of WSDL is that the particular format of a return value is not known. In other words, the user of a web service must know in advance what form the input parameters must have (for instance if a function is expecting a string, that string has a specific format like the format of a date for instance and will not work with a different format) and what format the output of the function adopts.

4.2.2 SOAP

SOAP (Simple Object Access Protocol) is a XML-based communication protocol for accessing a web service, created to communicate over HTTP (which is today supported by all internet browsers and services). SOAP is platform and language independent, simple and extensible. SOAP may also be used over HTTPS (which is the same protocol as HTTP at the application level, but uses an encrypted transport protocol underneath) with either simple or mutual authentication.

A SOAP message is an XML document containing:

- an *Envelope* element that identifies the XML document as a SOAP message and constitutes the root element; it contains the *namespace* attribute (which defines the envelope as a SOAP envelope) and the *encodingStyle* attribute (defining the data types used in the document);
- an optional *Header* element, containing application-specific information (like authentication, payment, etc) about the SOAP message;
- a *Body* element, that contains the actual SOAP message (call and response information);
- an optional *Fault* element containing errors and status information; it must be a child of the *Body* element and it can appear only once in a SOAP message.

Although using SOAP over HTTP allows for easier communication through proxies and firewalls than previous remote execution technology, the technique has the disadvantage of using an application level protocol (HTTP) as a transport protocol (critics have argued that abusing a protocol by using it in a different purpose may conduct in sub-optimal behaviour).

The following table shows a short summary of the advantages and disadvantages of SOAP protocol.

Table 2: Pros and Cons of SOAP

Pros and Cons SOAP	
Pros	Cons
Language, platform, and transport independent	Conceptually more difficult, more "heavy-weight" than REST
Designed to handle distributed computing environments	More verbose
Is the prevailing standard for web services, and hence has better support from other standards (WSDL, WS-*) and tooling from vendors	Harder to develop, requires tools

Built-in error handling (faults)	
Extensibility	

WSDL SOAP binding

As seen in [Butek 2005]⁷ the binding style (affects the way in which the body of a SOAP message is constructed) can be “RPC or Document”. A SOAP binding can also have an encoded use or a literal use. Table 3 aims to summarize the different possibilities.

Table 3: SOAP WSDL binding style/use

Binding Style	RPC: Remote procedure call. Needs to comply with conventions.
Affects the way in which the body of a SOAP message is constructed.	Document: No need to follow conventions. Soap message is sent as one document inside soap body element.
Use	Literal: rules to encode soap body with xml schema.
Specifies the encoding rules of the soap message.	Encoded: rules in a URL (defined in encodingStyle attribute).

[Butek 2005] presents five different options for the WSDL SOAP binding and it concludes that while each style has its place, under most situations the best style is **document/literal wrapped**. These are the basic characteristics of the document/literal wrapped pattern:

- The input message has a single part.
- The part is an element.
- The element has the same name as the operation.
- The element's complex type has no attributes.

Here are the strengths and weaknesses of this approach:

Strengths:

- There is no type encoding info.
- Everything that appears in the soap:body is defined by the schema, so you can easily validate this message.
- Once again, you have the method name in the SOAP message.
- Document/literal is WS-I compliant, and the wrapped pattern meets the WS-I restriction that the SOAP message's soap:body has only one child.

Weaknesses:

⁷ <http://www.ibm.com/developerworks/webservices/library/ws-whichwsdl/>

- The WSDL is even more complicated.

An example is presented in Table 4.

Table 4: SOAP WSDL document/literal wrapped example

<p>Java method</p> <pre>public void myMethod(int x, float y);</pre>
<p>Document/literal wrapped WSDL for myMethod</p> <pre><types> <schema> <element name="myMethod"> <complexType> <sequence> <element name="x" type="xsd:int"/> <element name="y" type="xsd:float"/> </sequence> </complexType> </element> <element name="myMethodResponse"> <complexType/> </element> </schema> </types> <message name="myMethodRequest"> <part name="parameters" element="myMethod"/> </message> <message name="empty"> <part name="parameters" element="myMethodResponse"/> </message> <portType name="PT"> <operation name="myMethod"> <input message="myMethodRequest"/> <output message="empty"/> </operation></pre>

```

</portType>
<binding .../>
<!-- I won't bother with the details, just assume it's document/literal. -->
Document/literal wrapped SOAP message for myMethod
<soap:envelope>
  <soap:body>
    <myMethod>
      <x>5</x>
      <y>5.0</y>
    </myMethod>
  </soap:body>
</soap:envelope>

```

4.2.3 REST

REST is an architectural style for distributed hypermedia systems such as the World Wide Web. The term was introduced in 2000 in the doctoral dissertation of Roy Fielding [Fielding 2000], who also participated in the IETF [Internet Engineering Task Force] working groups on URI, HTTP and HTML. The systems which follow REST principles are called RESTful.

Shortly, the basic REST principles are:

- Application state and functionality are abstracted into resources; all types of documents can be used as representations for resources (XML, XHTML, HTML, PNG, ...)
- Every resource is uniquely addressable using a universal syntax for use in hypermedia links (URI –Uniform Resource Identifier)
- All resources share a uniform interface for the transfer of state between client and resource, consisting of a constrained set of well-defined operations (represented by the GET, POST, PUT and DELETE methods) and a constrained set of content types (optionally supporting code on demand);
- The transfer protocol is client-server, stateless, cacheable and layered.

A RESTful web service is a simple web service implemented using HTTP and the principles of REST. Some advantages and disadvantages of REST are listed in the following table.

Table 5: Pros and Cons of REST

Pros and Cons of REST	
Pros	Cons

Language and platform independent	Assumes a point-to-point communication model--not usable for distributed computing environment where messages may go through one or more intermediaries
Much simpler to develop than SOAP	Lack of standards support for security, policy, reliable messaging, etc., so services that have more sophisticated requirements are harder to develop ("roll your own")
Small learning curve, less reliance on tools	Tied to the HTTP transport model
Concise, no need for additional messaging layer	
Closer in design and philosophy to the Web	

4.2.4 AXIS

Axis⁸ is an open source web service framework developed and maintained by the Apache Software Foundation⁹. Developers can use Java or C++ to encode applications and deploy them as web services. Axis is based on XML and SOAP protocol.

Latest release is Axis2¹⁰ from 23th October 2009 which supports SOAP 1.1 and SOAP 1.2, but it also has integrated support for REST style of Web services. It is carefully designed to support the easy addition of plug-in "modules" that extend their functionality for features such as security and reliability. Axis2 not only provides the capability to add Web services interfaces to Web applications, but can also function as a standalone server application.

Features:

- **Hot Deployment:** Axis2 can deploy Web services without having to shut down the server.
- **Flexibility:** The Axis2 architecture gives the developer complete freedom to insert extensions into the engine for custom header processing.
- **WSDL support:** Axis2 supports the Web Service Description Language, version 1.1 and 2.0, which allows you to easily build stubs to access remote services, and also to automatically export machine-readable descriptions of your deployed services from Axis2.

4.2.5 Common interfaces and tool integration

The main goal when deploying NLP tools (like Freeling or a tagger) as Web services was to create a scenario where services are easy to invoke and interoperable between one another if necessary. The user can operate those services at will and interconnect them to create complex execution chains. In computer science, interoperability is achieved by separating interfaces from implementations. Those interfaces can be defined specifically for each tool. In the end, the desired scenario is one in which several tools with the same functionality can be easily exchanged.

⁸ <http://ws.apache.org/axis/>

⁹ <http://www.apache.org/>

¹⁰ <http://ws.apache.org/axis2/index.html>

These requirements led us to explore the possibility to define common interfaces. If similar tools (with the same functionality) can be described with a common interface then it's easy to change them and to learn how to call them.

In WSDL, the operations are the basic unit with a specific functionality that can be invoked. Each operation can only have one *input message* and one *output message*. These messages define the information the service receives and returns when the operation is invoked. In the end, every operation will be implemented by a software *method* that needs all its necessary parameters to work. Those parameters must be represented in the message. The quickest solution, often, is to map all implementation parameters into the corresponding message parts.

Part messages can be defined case by case or use type definition via XML schemas (enabling type sharing and reusing). Example tools like FreeLing and TreeTagger show that the attempt to define a common set of typed input parameters is not feasible when dealing with implementation parameters. Rather, it is suggested an approach where complexity derived from concrete implementations has no significant consequences on interfaces. This can be achieved by avoiding the proliferation of implementation particular parameters in WSDL messages.

4.2.6 Metadata / Ontology / Closed vocabularies

This section is devoted to semantic annotations of services as a way of easing service interoperability. These annotations are eventually used for classifying, discovering, matching, composing and invoking Web services.

There are different approaches to semantic annotation of services (OWL-S¹¹, SAWSDL¹², etc.). They all agree on using some sort of semantic model that is used to annotate their services descriptions.

OWL-S is an ontology that covers everything from service description to service grounding (linking between the semantic model and the WSDL). OWL-S relates ontological concepts to real implementations.

SAWSDL is a W3C standard for semantic annotation of WSDL. SAWSDL provides standard means to relate WSDL documents to semantic descriptions.

There can be found some lightweight approaches that should be noticed. The myGrid semantic model (Wolstencroft et al., 2007) from the bioinformatics field presents an ontology divided in Grid Service Ontology (used to describe technical issues like WS, etc.) and Grid Domain Ontology (used to describe the bioinformatics field). This approach avoids much of the complexity of OWL-S based descriptions because invocation details are not included.

Using myGrid ontology input and output parameters can be defined in terms of their semantic types and the format. The format describes how data is represented and the semantic type describes the domain specific information. The operations are defined regarding the task they perform and the resources they use.

All these metadata are published in a registry (Feta or Biocatalogue for myGrid environment) which allow the user to make queries (Lord, et al, 2005).

11 OWL-S, Semantic Markup for Web Services. <http://www.w3.org/Submission/OWL-S/>.

12 SAWSDL, Semantic Annotations for WSDL and XML Schema. <http://www.w3.org/TR/sawSDL/>.

4.2.7 Concluding remarks. Comparative analysis and recommendations.

The PANACEA platform function is to integrate different functionalities deployed as web services. These web services could be developed using SOAP or REST. The small learning curve and the simplicity of REST technology are good advantages of this technology. However, the lack of standards and tool support makes it a possible solution for the future but not an adequate for first versions of PANACEA platform. The best option for PANACEA is to start developing using the SOAP protocol but keeping an eye on how the REST technology evolves. First versions of the platform could be based on SOAP only web services. Later versions could use some REST web services too.

The SOAP protocol uses WSDL to describe its web services interfaces. WSDL is the first point to start working with interoperability. Using the *document literal wrapped* flavour of WSDL will help PANACEA developers to create and use common interfaces and to reuse already made types.

To populate all these parameters, operations, etc. metadata and closed vocabularies could be crucial. After all, the web services are not developed for an in-house use, they are supposed to be discovered and used by users outside. Using an ontology will help users to find, use and combine web services. The OWL-S ontology seems to be a very complex solution. However, myGrid ontology presents an interesting division between service ontology (no need of further development) and domain ontology. The first one could be reused and PANACEA would only need the development of the domain ontology.

Table 6: Web Services and Ontologies comparative analysis

Web Services	Cost / learning curve	Functionalities	Integrability	maturity	Support and plans for the future	Availability
SOAP	😊	😊😊	😊😊	😊😊	😊	😊😊
REST	😊😊	😊	😊	😊	😊😊	😊😊
Ontologies	Cost / learning curve	Functionalities	Integrability	maturity	Support and plans for the future	Availability
OWL-S	😞	😊😊	😊	😊😊	😊😊	😊
MyGrid ontology	😊	😊	😊	😊😊	😊😊	😊

4.3 Workflow systems

The PANACEA platform will allow combining different components to create complex workflows. Users will need a user-friendly managing tool for the composition of their desired workflows. This section is devoted to the analysis of three existing workflow editors that could be used for the PANACEA platform.

The following sections present a survey developed by UPF to test the feasibility of using existing workflow editors to work with existing language technology tools as foreseen in PANACEA.

4.3.1 Triana

Triana¹³ is one of the two test bed applications developed for GridLab, a large EU funded project. The aim of GridLab is to develop a simple and robust grid application toolkit (GAT) enabling applications to exploit the power of the GRID.

Table 7: Triana Technical overview

Paradigm	Data flow based
Language	Proprietary language.
Concurrency	It has a good level of concurrency control with merge and block components.
Parallelism	Being a data flow driven software Triana has a very good level of parallelism.
Loops	Triana has a very good level of loop control. It can be done in several ways depending on the user decision and it accepts dynamic variables to control the exit condition. Any logic operation with variables can be used for the exit condition.
Conditions	The conditional components are not as powerful as the loop components and there is only a very basic IF clause
exception handling	Exception handling is good enough. Every component may have an error output (called error node). Actions can be programmed in the workflow when an error appears.
Other	Triana does not include a way to insert small algorithms in the workflow (this must be done by a web service or a local compiled component).

Table 8: Triana Descriptive information

Developers	Developed by Cardiff University
Domain	Astronomy and life sciences.
Maturity	Projects using Triana: GridOneD, GEO 600, BiodiversityWorld, DIPSO, FAEHIM, GEMMS, GEN-IUS Grid portal, Data-Mining Grid The last publication reported on the Triana's web page is 2009.
future plans	-- no information available --

¹³ <http://www.gridlab.org/WorkPackages/wp-3/index.html>

Community	Information taken from NesCForge: Public mailing lists: 6 lists with no activity Public Forums: 2 public forums with no activity First release: beta version 3.0 date: 2004-08-12 Last release: version 3.2.2 date: 2007-04-24
eHumanities	The DataMiningGrid project is a shared cost Strategic Targeted Research Project (STREP) granted by the European Commission (grant no. IST-2004-004475). It was part of the Sixth Framework Programme of the Information Society Technologies Programme (IST) Based on Triana 170 downloads from sourceforge
Integration	Triana runs on windows and linux.
documentation	Enough documentation when starting. However most of the functionalities and tools are not documented.

The GUI

a) Installation & documentation:

- Easy installation and well documented.
- The GUI includes help and tutorial (with some missing images and not completed).
- Problems when looking for help/support on the web.
- Not easy to start working with it.

b) Editing workflows:

The task of editing workflows is a bit verbose. Triana includes *Local workflows* (a large library of “local tasks”) and *Distributed workflows*. Distributed components within Triana include grid-oriented components (GRAM¹⁴ & GRMS¹⁵) and service-oriented components (web services and P2P). Service-oriented components use a GAP(Grid Application Prototype Interface) Interface which provides job submission and file-transfer operations within Triana.

Once a web service is imported, it appears as a tool in the *tool tree* alongside the other tools and can be connected into a Triana workflow in exactly the same manner as other 'local' tools.

Triana includes a graphical editor where selected processors (used in TRIANA to denote a component perform a unit of work) are dragged into the editor window. Input/output information is displayed in a pop-up window when the mouse is over the processor. As soon as processors are added to the editor, the system establishes the links between them.

c) Input / output:

¹⁴Grid Resource Allocation Management del Globus project

¹⁵Grid(Lab) Resource Management del GridLab project

In Triana input renderers are 'tools' and they are listed in the tool tree for 'Triana Tools' together with any other processor. The user needs to choose the relevant input/output tool in order to read/see inputs/outputs. As Triana was created to support astronomy and live sciences, most of the input/output tools are irrelevant to humanities. In order to deal with complex inputs, the user needs to generate static type classes and create custom tools.

d) Searching:

Triana includes a searching facility. However we were not able to search in the UDDI repository.

Table 9

Analysis of main characteristics	
Functionalities	workflow editor. good level of loop control.
Integration	Windows and Linux. Proprietary language.
Maturity	The last publication reported on the Triana's web page is 2009.
Support and plans for the future	Most of the functionalities and tools are not documented. Not documented plans for the future.
Availability	Open-source

4.3.2 Kepler

Kepler¹⁶ is a cross-project collaboration led by the Kepler/CORE team (UC Davis, UC Santa Barbara, and UC San Diego). The software builds upon the Ptolemy II framework, developed at the University of California, Berkeley. Ptolemy II is a software framework designed for modelling, design, and simulation of concurrent, real-time, embedded systems.

Table 10: Kepler Technical overview

paradigm	Data flow based
language	Proprietary language. Workflows can only be created and executed using Kepler tool. Kepler uses the proprietary Modelling Markup Language (MoML).
concurrency	The concurrency control is based on Merge as well that is enough for many workflows but not for some more sophisticated.
parallelism	Parallelism is full executed as normal in Kepler.
loops	Kepler has a lot in common with Taverna, it has no loops since it is data flow driven but in Kepler is not used the internal iteration system that Taverna has. In this way, any kind of iteration should be described using tricky ways. In addition, dynamic variables cannot be used as loop exit conditions.

¹⁶ <https://kepler-project.org/>

conditions	Good conditional components. We find select, switch, comparator, logic function, equal, isPresent and some more.
exception handling	Exception handling is not available in Kepler. Although the user can program raising exceptions in the required situations, there is not a way of catching exceptions when a web service is missing, timeouts, etc...
	Kepler allows to encapsulate algorithms easily in components (in Kepler are called actors). This is very helpful for connecting components that need small tidy up in the data before get connected.

Table 11: Kepler Descriptive information

developers	Developed by the members of the Ptolemy project at UC Berkeley.
domain	Molecular biology, ecology, geosciences, chemistry and oceanography.
maturity	First Kepler alpha 6 version released on April 29, 2005 Last release (Kepler 1.0.0) May 2008 Kepler is nightly updated.
future plans	
community	kepler-users mailing lists with moderate usage. kepler-dev mailing list or technical discussions with active usage Kepler includes a kepler repository which can be accessed from the web. The tool allows searching the repository. The organization of the repository and search capabilities is not developed.
eHumanities	Text-mining. MultiChek (the Multivalent – Chesire- Kepler VRE project) aims at developing a collaborative engineer environment able to provide new methods of creating, sharing, disseminating and reusing scholarly information.
integration	
documentation	Good documentation for both users and developers. Help manuals are good. It includes flash demos and examples.

The GUI

Easy installation and well documented. The Workstation includes user manual, examples and a ‘Kepler Actor’ reference file.

The graphical tool is good and has nice functionalities; however we faced some refreshing problems. (When editing and instantiating Java components, the application had to be shut down and restart in order to refresh the whole workflow).

Editing workflows

In Kepler components are called *Actors*. The system includes a large standard library of local actors. The way 'remote actors' are included in workflows differs from that of Taverna and Triana. In this case, the user needs to choose the correct actor 'type' from the local library. Once the relevant actor is dragged into the editor window, the user populates it with information concerning the WSDL uri and the method name.

Thus, for example, Kepler includes *WebServiceActor* and *WSWithComplexTypes*. These actors invoke the Web service and broadcast the response through their output ports. But whereas the first deals with web service operation with simple types, the later deals with web services with complex types. This means that the user needs to know whether the remote service requires simple or complex input types.

Note that in Triana and Taverna the way to 'load' remote processors from web services is quite different: the user enters the WSDL URI and the system populates the local library with the external processors. Once the remote processors are in the local library, the user 'drag&drops' them into the editor. In Kepler, the user selects the appropriate *Actor* and, once this is 'dropped' in the editor area, populates the *Actor* with WSDL information.

Again, the way GUIs deal with processors requiring complex input types is different. In Triana, the user needs to select the appropriate input renderer. In Taverna, the system automatically offers the possibility to include the local tool XMLsplitter which deals with complex input types.

Kepler allows adding local Java Actors (the equivalent Java Beans in Taverna). Java Actors are edited and compiled outside Kepler before they can be 'imported' and used in the workstation. For non-expert users, this is not an easy scenario.

In Kepler, every workflow requires a director. The user selects the relevant Director which directs the execution of the workflow. The list of directors includes: Synchronous Dataflow (SDF), Process Networks (PN), Dynamic Dataflow (DDF), Continuous Time (CT) and Discrete Events (DE). For a non expert user, choosing the appropriate Director is a non-trivial task.

Table 12

Analysis of main characteristics	
Functionalities	workflow editor. Good conditional components. No loops.
Integration	Windows, Mac and Linux. Workflows can only be created and executed using Kepler tool.
Maturity	The current version of Kepler is 1.0.0, released on May 12, 2008.
Support and plans for the future	Good documentation. https://kepler-project.org/
Availability	Kepler is freely available under the BSD License.

4.3.3 Taverna (myGrid)

Taverna¹⁷ is a free software workbench for designing and executing workflows, created by the myGrid project, and funded through OMII-UK (see later section 4.4.3).

Table 13: Taverna Technical overview

paradigm	Data flow based
language	Proprietary language, the Simple Conceptual Unified Flow Language or SCUFL. (Taverna2 no longer uses ScufL.)
concurrency	The concurrency offered by Taverna is basic and based on merge 2 branches when both are finished. It is enough for many workflows but some advanced ones are outside this approach.
parallelism	Parallelism is a primary concept in Taverna since it is Data Flow driven and almost all tasks are performed in parallel when possible.
loops	Taverna has not a <i>while</i> component that allows for iterations but there is the possibility to use list of items as input in the operations and set the option “iterate”. It will execute the operation once per item. It is allowed as well to set several input lists and the input will be taken one item of each list per processing iteration or as a Cartesian product per iteration. In most of the cases this system of iterations is enough but should have more powerful loop control like <i>while</i> statement.
exception handling	Taverna has a basic exception handling mechanism included. It supports retry of invocation with configurable timeout and number of retries, and user-defined alternatives for processors failing constantly.
other	All operations in Taverna must be encapsulated as a web services or local services programmed in java. There is no possibility of simple data manipulation in the workflow. It is very useful for connection between processes. Sometimes it is required to manipulate a little bit an output of a web service for being the input of another one, just a simple mathematical operation for instance.

Table 14: Taverna Descriptive information

developers	Created by the myGrid project, and funded through OMII-UK Original myGrid Partners : EMBL-EBI University of Manchester University of Newcastle University of Nottingham University of Sheffield University of Southampton IT Innovation Centre
domain	e-biology

¹⁷ <http://www.taverna.org.uk/>

maturity	<p>Taverna 2.1 Workbench is the latest version of the Taverna Workbench. It is highly recommended that you use this version if you are new to Taverna, or to migrate to this version if you have used Taverna Workbench before.</p> <p>Taverna 1.7.2 Workbench, the latest version the Taverna 1.x Workbench series, is still available to download. Users are recommended to switch to Taverna 2.1 Workbench where possible.</p> <p>Taverna Server is the remote workflow execution service that enables you to set up a dedicated server for executing workflows remotely. Taverna Server uses the Taverna 1.7.x API.</p> <p>Many 'e-biology' projects can be accessed using Taverna: Seqhound, BioMoby, BioMart, BioTeam iNquiry, Utopis, BioMart, EMBOSS(Soaplab).</p>
future plans	<p>April 2010 – Taverna Server 2.x Beta</p> <p>May 2010 – Taverna Workbench 2.x Beta</p> <p>(Roadmap)</p>
community	<p>Mailing Lists: Different mailing lists with 2580 users, moderate usage.</p> <p>Taverna has a social web site named myExperiment with over 964 users, 82 groups, 301 workflows, 101 files and 15 packs. MyExperiment 'makes it really easy to find, use and share scientific workflows and other files, and to build communities'.</p>
eHumanities	text-mining
integration	Taverna runs on any modern PC or Mac, running any recent version of Windows, Linux, OSX and most UNIX like operating systems as long as Java version 5
documentation	<p>Good documentation. However, some examples in the documentation are not in the distribution.</p> <p>Although the GUI is really friendly and 'easy' to use, it does not include any help.</p>

The GUI:

a) Installation and documentation

Taverna GUI is nice and friendly. It is easy to install, execute and start with. Taverna is in general well documented.

b) Workflow editor

The editor is friendly and nice and includes drag-and-drop facilities and visual facilities. Taverna has two spaces: the *editor* (Advanced Model Explorer) and the visualization. It also includes and interactive graphical (experimental) editor. The *editor* panel contains a tree with five initial branches for processors, inputs, outputs, data connections and coordination links. Basically, the process of editing workflows goes as follows:

- The user selects the desired processors. When processors are drag-and-dropped into the *editor* panel they are automatically placed under the processors node.
- The user has to define the inputs and outputs of the workflow and link the processors.

- The graphical panel displays the diagram as we are editing the workflow. The system allows for different visualizations and the diagram can be saved.

c) Inputs

The edition of inputs (and outputs) in Taverna is different from that of Triana and Kepler. In Taverna, inputs are not longer listed together with *processors*. They have a different status. The user creates as many inputs as needed under the branch *inputs* in the *editor* panel. The *input* branch is populated with the created inputs. The user names the input and edits the metadata assigned to it. Metadata includes (i) free text description (used to give instructions to other users about how to populate the inputs) and (ii) tagging of the input (or output) with MIME types. In case of outputs, MIME types are crucial as they determine the selection of renderers within the result browser (see below).

Taverna includes tools to deal with complex inputs in the workflow. When a web service has a complex input (an xml document) it is hard for the user to provide the required data without having previous knowledge of the structure. Taverna includes an *XML splitter* processor which deals with complex inputs. XML splitters show the child inputs to the user. Child inputs can be *edited* as simple inputs and the values are assigned in the standard manner. When a processor has a complex input, the system automatically offers the possibility to include an XMLsplitter to deal with.

d) Assigning values.

Contrary to Triana and Kepler, in Taverna parameters do not have to be pre-specified. That is, the assignation of values to inputs is not done during workflow construction but is done during workflow execution. When a workflow is invoked, a pop-up window is displayed to assign input values. The user can enter values in different ways: typing, reading from a file, reading from a directory or from a previously saved *file input definition*.

The assignation of values in Taverna is *independent* from the workflow definition. In Triana, for example, if we want a workflow to read from a file, we need to select a specific *read from file* tool. Similarly, if an xml output is produced, Taverna includes an xml viewer which automatically displays the data and allows the user to save it. In Triana, the user has to choose beforehand whether she/he wants to see the data (no syntax displayed) or save them.

The system allows specifying default values. In this case, default values are assigned during the edition process. When running workflows requiring inputs, these can be saved. This allows to re-run the workflow using the same inputs.

e) Outputs

Taverna includes different renderers to display results (results may have different formats). Provided the correct MIME types are specified for the workflow output, the renderer selection mechanism will select, by default, an appropriate renderer component. Thus, the user can see plotters, graphics, xml files and text files without having to specify anything.

Taverna includes third party visualisation tools. This is the case of SeqVISTA graphical tool (which displays some chemical/...' types) and Jmol (an open-source Java viewer for chemical structures in 3D). Taverna allows the user to inspect and save intermediate inputs and outputs of individual processors both during and after a workflow invocation.

Outputs can be saved in different manners.

- Iterations: When a processor expects single sequence and receives a list of sequences iteration takes place automatically. Specific iterations behaviours can be defined.
- Fault tolerance: Taverna includes the following fault tolerance settings:
 - The ability to retry after failures
 - User can set the number of retries
 - User can set the time between retries
 - The ability to define alternative processors when all retries have been exceeded
 - The ability to define a processor as critical, in this case the workflow stops
- Discovery, metadata-management and grimoires registry: that is the component within myGrid responsible for semantic service search. Feta is composed of two components, namely Feta Client and Feta Engine. The Feta Client is a GUI-plug-in to Taverna which is used to search for services descriptions of which are provided by the Feta Engine.

Taverna includes a good discovery tool, user can search by: name & description and, most interesting, by tasks, method and resources used, input and output, and type. Multiple queries can be issued at the same time to save the communication time. Multiple searching criteria can be combined in one query

Grimoires is an UDDIv2 compliant service registry. It was originally developed for the myGrid Project (www.mygrid.org.uk). Currently, it is a managed program project of Open Middleware Infrastructure Institute www.omii.ac.uk. Grimoires provides metadata annotation/discovery and WSDL registration/discovery functions that are not supported by UDDI.

f) Interesting features:

- Taverna includes the *Resource usage report* functionality which shows the various external resources used by the current workflow. Such functionality is useful for documentation purposes.
- The workflow diagram includes different visualization options.
- For large workflows some non-interesting parts can be marked as *boring*. Boring processors are hidden from diagram.
- myExperiment plug-in allows access to workflows in myExperiment. This allows for both browsing and direct invocation.
- Provenance tools: The Taverna Log Book is a plug-in for Taverna that allows users to automatically log their experiments in a database and browse, reload, rerun and maintain past workflows.
- LSID (Live Science Identifiers) for editable metadata associated to a workflow. Any time the user edits a definition, the system connects to whatever LSID authority is configured in the `mygrid.properties` file and asks for a new LSID suitable for a workflow definition. This then provides a globally unique identifier for the workflow definition.
- Taverna offers the ability to save *input configuration* to re-run the experiment with the same data input.
- Ability to define alternate processors in case of failure

- Taverna allows the user to inspect and save intermediate inputs and outputs of individual processors both during and after a workflow invocation.
- Good discovery tool (services can be discovered by exploiting their semantic descriptions. This is done by Tavern Feta Plug-in.
- Taverna includes a BeanShell editor tool (this is easier than working with Java and does not need to compile and use external tools).
- Breakpoints allow users to stop the workflow and edit data before going on.

Taverna Server

Taverna 1.7.x Server combines a Remote Execution Service with the Remote Execution [plug-in](#) for Taverna 1.7.x Workbench to give you the possibility to set up a dedicated server for executing workflows remotely. The user can submit a workflow for execution from Taverna 1.7.x Workbench, detach, and check the workflow execution status later either from the Workbench or from a Web page.

Taverna future work ([Roadmap](#))

This is a list of some future work and improvements to be developed in 2010 for Taverna that may be interesting for the PANACEA project:

- Enhanced security support for Web services
- myExperiment integration
- A better option for long running workflows
- Taverna Server 2.x Beta

Table 15

Analysis of main characteristics	
Functionalities	workflow editor. Multiple useful features.
Integration	Windows and Linux. myExperiment integration.
Maturity	Taverna 2.1 Workbench is the latest version of the Taverna Workbench. Taverna is continuously growing.
Support and plans for the future	Good documentation. (Roadmap)
Availability	Free and under the Lesser General Public License (LGPL) Version 2.1 .

4.3.4 LoonyBin

LoonyBin (Clark and Lavie, 2010) has been developed by Jonathan Clark (CMU) in order to address specific issues arising within the machine translation group. The MT group has developed a complex workflow using LoonyBin. There is a machine translation tool pack (Clark et al., 2010) available developed by members of the CMU machine translation group and others.

Table 16: LoonyBin technical overview

Paradigm	Data flow based Unix shell script generation
Language	Python/Bash
Concurrency	LoonyBin lets the user define different machine configurations (remote and local) and uses external schedulers (Torque, Sun Grid Engine (experimental support), Condor) and SSH to organize the workflow execution.
Parallelism	All vertices in a workflow will be run in parallel when their dependencies are satisfied.
Loops	Iterations are not supported. The only way to realise loops is to adapt the respective tools being executed which makes it impossible to use one tool's output as a looping condition.
Exception handling	A basic system is implemented that sends a notice via e-mail in case an exception occurs. In case of a failure in the workflow, successfully performed steps will automatically be recovered.
Other	LoonyBin does not provide web services.

Table 17: LoonyBin Descriptive information

Developers	Jonathan Clark (CMU)
Domain	Language Technology (Machine Translation)
Maturity	V0.5 has been released in 04/2010,
future plans	No information available.
Community	There does not seem to be any active community. The public mailing list has been used for announcements by Jonathan Clark only.
Integration	The Bash scripts generated by LoonyBin require a Unix-based system to run. The workflow editor runs on Windows, too.
Documentation	A comprehensive tutorial including an example workflow covers most relevant issues, but there is no complete reference.
Availability	LoonyBin is published under the LGPL.

The GUI:

a) Installation and documentation

An installation is not necessary; the Java GUI can be executed straight after download on any operating system providing a Java runtime environment. The setup will still require basic Bash/Unix shell knowledge. There is a GUI for the workflow editor only which generates the command line shell scripts. It is mostly intuitive and the tutorial features a full description for the editor. However, the documentation does not cover the development of custom tool packs.

b) Workflow Editor

The editor works all by drag and drop; any component is presented in a list and can be added to the workflow as a vertex. The vertices are connected by dragging a line; subsequently the interface is configured in a simple dialogue window providing the available inputs and outputs of the components. It can automatically connect corresponding inputs and outputs if they have appropriate names, e.g. *nBestout* and *nBestIn*. The components' specific parameters are defined in a side panel accessible through a click on the corresponding vertex. Different workflow paths can be defined that are conditioned on certain parameter values, e.g. produced by certain components or configure by the user.

After a workflow has been created, the editor generates a Bash script including the commands to call and path specifications (where to find the applied tools, where to write the output, which e-mail address to notify in case of failure etc.). This script will then be executed manually in the Unix shell. Different machine configurations can be set and assigned to vertices which allows for automatic execution on remote Unix machines including scheduling. LoonyBin provides a web server on the *home machine* that can be used to monitor the tasks.

c) Inputs

The user creates as many inputs as needed. The editor provides readers for input files stored in the local or in a Hadoopi file system. Alternatively, a *parameter box* lets the user define any input parameter. It can be configured such that it produces one or multiple outputs in any format as entered by the user. However, the configuration interface is not suitable for entering large data directly.

The input readers will be connected to a vertex that is capable of reading the respective format, i.e. a self-defined converter or a processing tool that reads the input file's format. LoonyBin does not come with any format converters; the machine translation components provided by the package are expecting data that is pre-formatted such that they suit the given tools.

As LoonyBin has been developed in a language technology context, there are several relevant tools already included, such as tools dealing with monolingual and parallel corpora, the Stanford English syntax parser, the Berkeley word aligner, language modelling tools etc.

d) Outputs

The output of each vertex, if any, will be written into the directory specified when generating the script, as well as the log files. This enables a detailed inspection of the single components in a workflow and the recovering of the components' outputs in case of failure.

Regarding the output format, appropriate converters need to be implemented if desired and necessary. The output will eventually be written into a file; LoonyBin does not come with any tool for visualization or other analysis of the output.

e) Including custom components

In order to make additional tools available for the LoonyBin workflow editor, a simple descriptor needs to be written in Python. It has to provide a couple of pre-defined functions that call the respective tool with the appropriate arguments.

4.3.5 Concluding remarks. Comparative analysis and recommendations.

The test conducted by UPF showed that for the test scenario (a simple chain with some conditional clauses) all tested workflow editors had several problems: with Triana it was impossible to test the whole workflow because there were several server problems that could not be solved. Kepler could not deal with some loop problems. However, in some cases, Taverna presented some feasible alternatives to those problems. Another advantage of using Taverna is that myExperiment portal (it will be ex-

plained later) allows executing its workflows directly from a web browser. LoonyBin is rejected because it is not web service oriented.

Table 18: Workflow systems comparative analysis

	Functionalities	Cost / learning curve	Integrability	Maturity	Support and plans for the future	Availability
Triana	😊	😊	😊😊	😊	😐	😐
Kepler	😊	😊	😊😊	😊	😊😊	😊
Taverna	😊😊	😊😊	😊	😊	😊😊	😊😊
LoonyBin	😐	😊😊	😊	😐	😊	😊😊

4.4 Grid infrastructure

Grid infrastructure is a technology that allows the coordination of all kind of computing resources. These resources are not under a centralized control and can be of different nature (processors, data storage, applications, etc.). From this point of view Grid is a new distributed computing infrastructure that interconnects heterogeneous resources using wide area networks like the internet that could be useful as a complete technical solution (web services, workflows, managing tools, etc) for PANACEA.

4.4.1 Globus

The Globus¹⁸ Alliance is an international collaboration group working on research and development of Grid technologies. This Grid aims to allow users to share databases, on-line tools and computing resources securely using some networks.

The project has changed the way science is conducted. High-energy physicists designing the Large Hadron Collider at CERN are developing Globus-based technologies through the European Data Grid, and the U.S. efforts like the Grid Physics Network (GriPhyN) and Particle Physics Data Grid. Other large-scale e-science projects relying on the Globus Toolkit include the Network for Earthquake Engineering and Simulation (NEES), FusionGrid, the Earth System Grid (ESG), the NSF Middleware Initiative and its GRIDS Center, and the National Virtual Observatory.

The [Globus Toolkit](#)[®] is a set of tools designed to provide distributed security, resource management, monitoring and discovery, and data management. GT5 is the latest version of the toolkit and its components and libraries are compliant with the [Web Services Resource Framework \(WSRF\)](#), a set of standards in development in [OASIS](#). The Globus Alliance is a leading member of the [Global Grid Forum \(GGF\)](#). This forum has defined a framework named [Open Grid Services Architecture \(OGSA\)](#) which Globus toolkit tools are compliant.

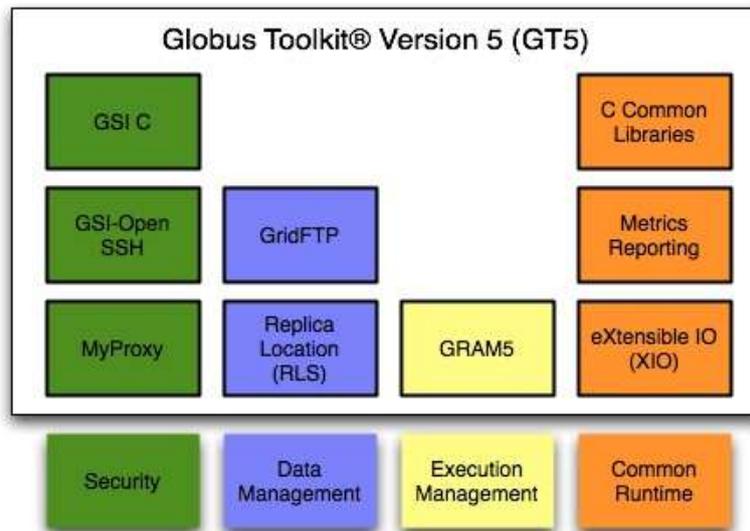
Globus Toolkit ®

¹⁸ <http://www.globus.org>

The toolkit includes components that can be used either **independently or together** for resource monitoring, discovery, and management, plus security and file management. Using internet combined with these tools a user can access remote resources seamlessly but keeping the local access control.

The Globus Toolkit has grown through an open-source strategy. Figure 2 shows the different components classified regarding its functionality.

Figure 2: Globus Toolkit software services



Security

The Globus toolkit provides WS and non-WS authentication and authorization capabilities. All security infrastructure in Globus grid is based on the standard X.509 end-entity and proxy certificates. Certificates are used to identify persistent entities such as users and servers. Certificates are used too to delegate temporary privileges to other entities.

Globus security tools aim to identify users and services (authentication), to protect the integrity and privacy of communications, authorization, and provide logs to verify that all the security architecture works as planned.

Security Key Concepts

- Grid Security Infrastructure (GSI)
 - [GSI C](#)
- Security Services
 - [MyProxy](#)
- Run your own Certificate Authority (CA)
 - [SimpleCA](#)
- Utilities
 - [GSI-OpenSSH](#)

Data Management

GridFTP is a high-performance, secure, reliable data transfer protocol optimized for high-bandwidth wide-area networks. The GridFTP protocol is based on FTP, the highly-popular Internet file transfer protocol.

Execution Management

Execution management tools objective is the initiation, monitoring, management, scheduling, and/or coordination of remote computations (or jobs). GT5 includes the Grid Resource Allocation and Management (**GRAM**) interface as a basic mechanism for these purposes.

Table 19

Functionalities	software toolkit used for building grids. Different modules that can be used independently. Security and massive data.
Integration	Linux.
Maturity	Latest Stable Release: 5.0.1 (23-3-2010).
Support and plans for the future	Good documentation. Each module has a lot of documentation.
Availability	Open source software.

4.4.2 EGEE

Enabling Grids for E-sciencE (EGEE¹⁹) is a European project that provides computing support for over 13,000 researchers working on different fields.

All resources are being coordinated by EGEE now. However, by the end of April 2010, that responsibility will be transferred to the European Grid Infrastructure (EGI). That means each country's grid infrastructure will be run by National Grid Initiatives. EGI's main goal is to ensure abundant and quality computing support for the research community for the future.

The Worldwide LHC Computing Grid Project (WLCG) was created to prepare the computing infrastructure for the simulation, processing and analysis of the data of the Large Hadron Collider (LHC) experiments. The LHC is the world's largest and most powerful particle accelerator. The WLCG and the EGEE projects share a large part of their infrastructure and operate it in conjunction.

WLCG/EGEE operates a production Grid distributed over more than 200 sites around the world, with more than 30,000 CPUs and 20 PB of data storage. The status of the Grid can be seen from the various monitoring pages linked from the Grid Operations Centre (GOC²⁰) monitoring page.

Middleware

The EGEE infrastructure is build upon middleware software named gLite²¹. Using gLite facilitates accessing shared storage resources across the internet and sharing computing resources as well.

¹⁹ <http://www.eu-egee.org/>

²⁰ <http://goc.grid-support.ac.uk/gridsite/monitoring/>

²¹ <http://www.glite.org/>

The available services in the gLite distribution can be broadly classified in two categories:

- *Grid Foundation Middleware*, covering the security infrastructure, information, monitoring and accounting systems, access to computing and storage resources, providing the basis for a consistent and dependable production infrastructure;
- *Higher-level Grid Middleware*, including services for job management, data catalogs and data replication, providing applications with end-to-end solutions.

The gLite middleware aims to ensure easy installation and configuration on the chosen platforms (currently Scientific Linux versions 4 and 5, and also Debian 4).

Security

To access WLCG/EGEE a user must be registered. The Grid Security Infrastructure (GSI) in WLCG/EGEE provides secure authentication and communication. The system is based on X.509 certificates, public key encryption and the Secure Sockets Layer (SSL) communication protocol. GSI allows single sign-on and delegation.

User interface

A user can access the WLCG/EGEE from any machine where users have a personal account and where their user certificate is installed.

The WLCG/EGEE user interface (UI) provides CLI²² tools to perform some basic Grid operations:

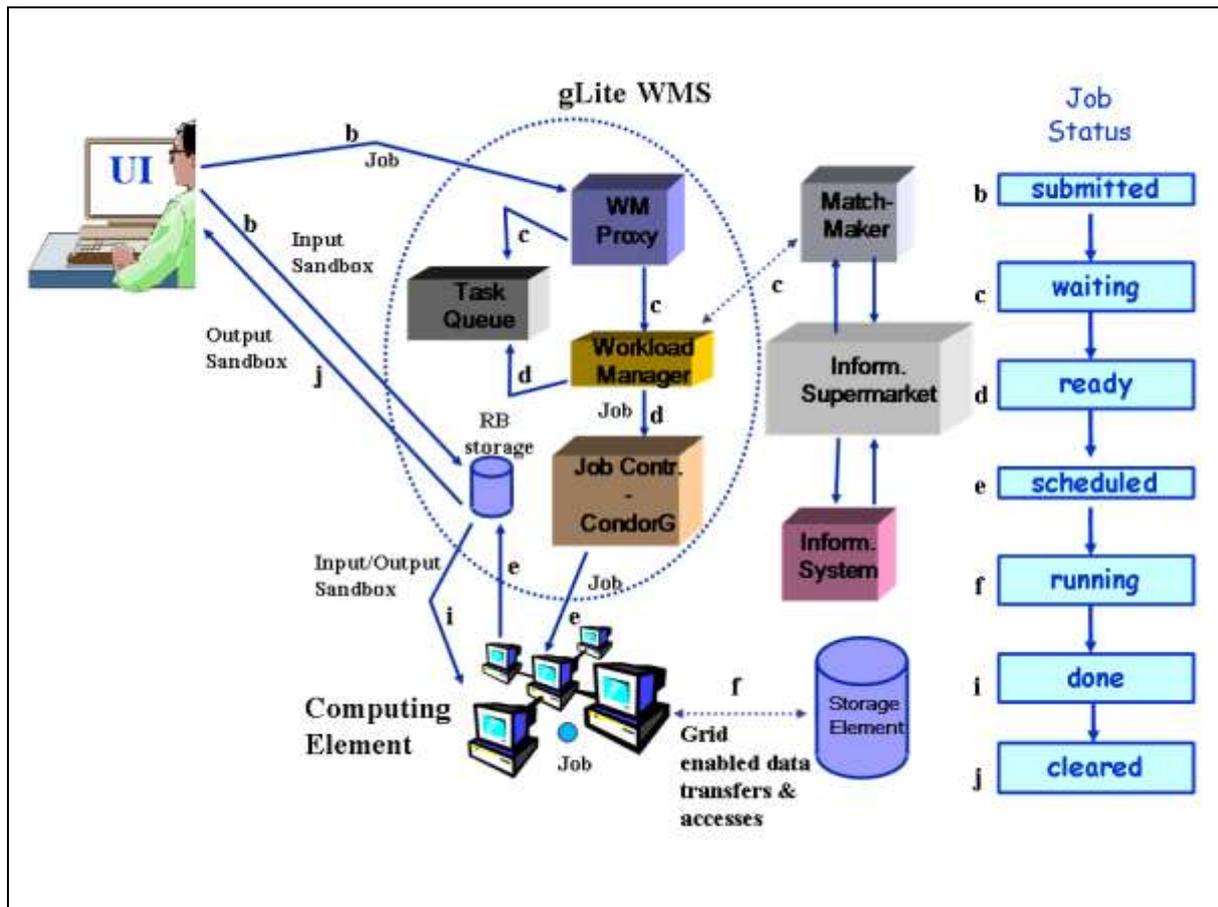
- list all the resources suitable to execute a given job;
- submit jobs for execution;
- cancel jobs;
- query the status of jobs and retrieve their output;
- copy, replicate and delete files from the Grid;
- submit and manage file transfer jobs
- retrieve the status of different resources from the Information System.

Job flow

Figure 3; **Error! No se encuentra el origen de la referencia.** illustrates the process that takes place when a job is submitted to the Grid. The individual steps are as follows:

²² CLI: Command Line Interface

Figure 3



a. After obtaining a digital certificate from a trusted Certification Authority, registering and obtaining an account on a User Interface, the user is ready to use the WLCG/EGEE Grid. He logs in to the UI and creates a proxy certificate to authenticate him in subsequent secure interactions.

b. The user submits a job from the UI to the gLite WMS (Workload Management System). In the job description one or more files to be copied from the UI to the WN (Worker Node) can be specified, and these are initially copied to the gLite WMS. This set of files is called the Input Sandbox. An event is logged in the LB (Logging and Bookkeeping Service) and the status of the job is SUBMITTED.

c. The WMS looks for the best available CE (Computing element) to execute the job. To do so, it interrogates the Information Supermarket (ISM), an internal cache of information which in the current system is read from the BDII, to CERN-LCG-GDEIS-722398 Manuals Series Page 30 determines the status of computational and storage resources, and the File Catalogue to find the location of any required input files. Another event is logged in the LB and the status of the job is WAITING.

d. The gLite WMS prepares the job for submission, creating a wrapper script that will be passed, together with other parameters, to the selected CE. An event is logged in the LB and the status of the job is READY.

e. The CE receives the request and sends the job for execution to the local LRMS (Local Resource Management System). An event is logged in the LB and the status of the job is SCHEDULED.

f. The LRMS handles the execution of jobs on the local Worker Nodes. The Input Sandbox files are copied from the gLite WMS to an available WN where the job is executed. An event is logged in the LB and the status of the job is RUNNING.

- g. While the job runs, Grid files can be directly accessed from a SE or after copying them to the local file system on the WN with the Data Management tools.
- h. The job can produce new output files which can be uploaded to the Grid and made available for other Grid users to use. This can be achieved using the Data Management tools described later. Uploading a file to the Grid means copying it to a Storage Element and registering it in a file catalogue.
- i. If the job ends without errors, the output (not large data files, but just small output files specified by the user in the so called Output Sandbox) is transferred back to the gLite WMS node. An event is logged in the LB and the status of the job is DONE.
- j. At this point, the user can retrieve the output of his job to the UI. An event is logged in the LB and the status of the job is CLEARED.
- k. Queries for the job status can be addressed to the LB from the UI. Also, from the UI it is possible to query the BDII for the status of the resources.
- l. If the site to which the job is sent is unable to accept or run it, the job may be automatically resubmitted to another CE that satisfies the user requirements. After a maximum allowed number of resubmissions is reached, the job will be marked as aborted. Users can get information about the history of a job by querying the LB service.

Table 20

Functionalities	gLite middleware framework to create grid applications. Security and massive data.
Integration	Scientific Linux versions 4 and 5, and also Debian 4.
Maturity	08.02.2010 - gLITE 3.2 Update 08
Support and plans for the future	Good documentation. Long term funding and support.
Availability	Open source software. Apache LICENSE-2.0

4.4.3 MyGrid

The myGrid²³ team, led by Professor Carole Goble²⁴ of the School of Computer Science at the University of Manchester²⁵, UK. is a research group focusing on e-Science. The team is formed with different institutions and people from different disciplines together in an international environment.

The myGrid team work to develop a suite of tools designed to help scientists with the creation of e-laboratories and have been used in domains as diverse as systems biology, social science, music, astronomy, multimedia and chemistry. The tools have been adopted by a large number of projects and institutions.

²³ <http://www.mygrid.org.uk/>

²⁴ <http://www.mygrid.org.uk/about-us/people/core-mygrid-team/carole-goble/>

²⁵ <http://www.manchester.ac.uk/>

Tools

These tools and infrastructure allow:

- Design, edit and execution of workflows in [Taverna](#)
- Sharing of workflows and related data by [myExperiment](#)
- Cataloguing and annotation of services in [BioCatalogue](#) and [Feta](#)
- Creation of user-friendly rich clients such as [UTOPIA](#)

The myGrid e-Laboratory includes a suite of components for the creation, cataloguing, annotation, discovery and monitoring of services:

- [SoapLab](#) - creation of services (wrapper).
- [BioCatalogue](#) and [Feta](#) - cataloguing, annotation and discovery of services.
- [BioCatalogue](#), [Feta](#) and [Find-O-Matic](#) - discovery of services.
- [Workflow Monitor](#) - monitoring of workflows and the services in them.
- [QuASAR](#) - validation and inference of annotations.

MyGrid and the MOBY²⁶ consortium have developed a [myGrid Ontology](#) that is used for service annotation. This ontology is separated into two parts, the *service ontology* and the *domain ontology*. The service ontology aims to describe the web services from a technical point of view: it describes inputs, outputs, operations, etc. The domain ontology, on the other hand, describes the bioinformatics research and acts as an annotation closed vocabulary for bioinformatics data types.

The scope of the ontology is limited to support service discovery. Each hierarchy contains abstract concepts to describe the bioinformatics domain at a high level of abstraction. By combining the terms from the ontology, descriptions of services are constructed to detail:

1. What the service does.
2. What data sources it accesses.
3. What each of the inputs and outputs should be.
4. Which domain specific methods the analysis involves.

By describing the domain of interest in this way, users should be able to find appropriate services for their experiments from a high level view of the biological processes they wish to perform on their data.

Scope of the myGrid ontology:

- **Informatics:** captures the key concepts of data, data structures, databases and metadata. The data and metadata hierarchies in the ontology contain this information

²⁶ <http://www.biomoby.org/>

- **Bioinformatics:** This builds on informatics. As well as data and metadata, there are domain-specific data sources (e.g. the model organism sequencing databases), and domain-specific algorithms for searching and analyzing data (e.g. the sequence alignment algorithm). The algorithm and data resource hierarchies contain this information.
- **Molecular biology:** This includes the higher level concepts used to describe the bioinformatics data types used as inputs and outputs in services. These concepts include examples such as, protein sequence, and nucleic acid sequence.
- **Tasks:** A hierarchy describing the generic tasks a service operation can perform. Examples include retrieving, displaying, and aligning.
- **Services:** The concepts required to describe the function of web services and their parameters. The service ontology is described in more detail below.

The myGrid ontology can be downloaded in OWL and RDFS.

MyGrid uses a special XML file, called Semantic Service Descriptor, to provide user with much richer details and information about a WS. Every WS is described by its WSDL and its Semantic Service Descriptor using the Semantic Model (the ontologies). All these rich data can be afterwards used by the users for web services discovery, to create workflows, etc.

Figure 19: WSDL / Semantic service description shows the relation between the WSDL and the Semantic Service Descriptor.

Web Service developer guidelines

MyGrid aims to offer a [guideline](#) to WS developers to create WS-I²⁷ compliant Web Services. However, this is a long-term goal. By now they basically want service providers to create web services that are compatible with Taverna applying the following guidelines:

- The preferred binding style is **Document/literal wrapped**.
- Avoid untested situations like: multiple WSDL imports, multiple service endpoints, ambiguous type names (identically named types that belong to different namespaces).
- Avoid situations that are known to fail in Taverna: Cyclic references, Overloaded operations, using anyType type.

Table 21

Functionalities	Tools and ontologies.
Integration	Linux and windows (depends on the tool). Very good integration between tools.
Maturity	Soaplab 2.2.0 (26-06-2009), Taverna 2.1.2 (2010), Biocatalogue (latest release 26-05-2010)

²⁷ Web Services Interoperability. <http://www.ws-i.org/Profiles/BasicProfile-1.1.html>

Support and plans for the future	Good documentation. All tools have continuous support and development.
Availability	Free and open-source.

4.4.4 TextGrid

TextGrid²⁸, is based on the Globus²⁹ grid infrastructure and benefits from its security tools and integration capabilities to interconnect a group of web services designed to work with texts. The user interface is build with the Eclipse RichClient Platform.

The project is part of the D-Grid initiative³⁰ in Germany and its main goal is to create a community for the collaborative editing, annotation and analysis of texts.

Most tools in the TextGrid are implemented as web services. They represent simple functionalities as tokenization and lemmatizing and be executed alone or combined in workflows. The services can be registered into a topic map-based service registry³¹.

To promote integration and interoperability there are different levels of integration in the system. For example, authentication is necessary from the third step on. Each initiative can choose how far they advance into the TextGrid service network.

User interface

The first user environment is an Eclipse-based client. Some web interfaces could be developed in the future.

Grid access

The service carries all the necessary information to be allowed into the grid. This includes information for authentication and logging. TextGrid is Shibboleth-enabled [SHIBBOLETH³²] and will interconnect with the national scientific Shibboleth federation DFN-AAI [DFN] once productive. For the time being, users can register at TextGrid directly to be granted access. Grid access entails the possibilities to join projects, deposit and share (private) digital objects, and similar activities. In the future, licensing policies may build on authentication information.

Table 22

Functionalities	Globus based grid for text processing. Depends on Eclipse platform. Security. Limited amount of tools.
Integration	Services need to be Textgrid compliant (there are different levels of integra-

²⁸ <http://www.textgrid.de/>

²⁹ <http://www.globus.org/>

³⁰ <http://www.d-grid.de>

³¹ To be implemented in TextGrid II.

³² <http://shibboleth.internet2.edu/>

	tion)
Maturity	New and under development.
Support and plans for the future	The second phase of the project has as goals the conversion of the virtual research ambit into a sustainable operation, as well as achieving a wide base usability.
Availability	TextGridLab - Beta Version is free to be downloaded and open source project.

4.4.5 NorduGrid

A project called "[Nordic Testbed for Wide Area Computing and Data Handling](#)" was launched in 2001. Its goal was to create a Grid infrastructure which could handle research tasks at production level. As a result of that project a new middleware was developed. That middleware, called [Advance Resource Connector \(ARC\)](#), has been continuously improved to deliver a robust, scalable, portable and fully featured solution for a global computational and data Grid system.

Grid technologies are not new and its architectures are widely used. However, most of the Grid software is complex and not portable (a few OS are compatible). Installation is complex and maintenance costs are high. ARC middleware has been developed and improved giving special attention to reliability, performance, ease of use and maintenance.

NorduGrid architecture was designed following some basic principles:

- Start with simple things that work and proceed from there
- Avoid architectural single points of failure
- Should be scalable
- Resource owners retain full control of their resources
- As few site requirements as possible:
 - No dictation of cluster configuration or install method
 - No dependence on a particular operating system or version
- Reuse existing system installations as much as possible
- The NorduGrid middleware is only required on a front-end machine
- Compute nodes are not required to be on the public network
- Clusters need not be dedicated to Grid jobs

ARC provides a set of fundamental Grid services, such as information services, resource discovery and monitoring, job submission and management, brokering and data management and resource management.

The middleware builds upon standard Open Source solutions like the [OpenLDAP](#), [OpenSSL](#), [SASL](#) and [Globus Toolkit](#)[®] (GT) libraries. Most services are build on GSI security layer.

Services:

- Grid Manager
- gridftpd (the ARC/NorduGrid GridFTP server)
- the information model and providers (NorduGrid schema)
- User Interface and broker (a "personal" broker integrated into the user interface)
- Extended Resource Specification Language (xRSL)
- Replica Catalog
- the monitoring system

The *standalone client* is available for a dozen of platforms and can be installed in a few minutes. The server installation does not require a full site reconfiguration. The middleware can be built on any platform where the external software packages (like GT libraries) are available.

ARC middleware is distributed under the [Apache v2.0 license](#).

4.4.6 Concluding remarks. Comparative analysis and recommendations.

Globus and EGEE infrastructures are very similar solutions: they are solutions for very large scale grids providing security and massive data handling. Some drawbacks of these solutions are the big complexity of the infrastructure, the large learning curve for users, developers and system administrators, the non-user-friendly user interfaces and the software compatibility limitations. On the other hand, MyGrid infrastructure offers tools with more user-friendly interfaces and a smaller learning curve. However, security is not implemented by default and massive data handling is not as well managed as in the other solutions.

The recommendation for PANACEA is to use MyGrid infrastructure and tools and benefit from its small learning curve and watch its roadmap plans that could solve part of its problems. However it is recommended to keep watching closely the big grids technological evolutions. It should be taken into account that Globus toolkit claims to be modular and some of its tools could be used for some concrete purposes.

From the TextGrid experience it could be interesting to make a deeper analysis on the Shibboleth web single sign-on security system which could be useful for PANACEA.

NorduGrid architecture can be considered a large scale Grid in which usability, costs and compatibility have been taken as key issues. However, it doesn't have the advantages of the myGrid approach and needs a further analysis and test. It could be a very interesting starting point for testing large scale Grids in PANACEA.

Table 23: Grid comparative analysis

	Functionalities	Cost / learning curve	Integrability	maturity	Support and plans for the future	Availability
Globus	😊	😬	😬	😊😊	😊😊	😊
EGEE	😊	😬	😬	😊😊	😊😊	😊
NorduGrid	😊	😊	😊	😊	😊	😊
MyGrid	😊	😊😊	😊	😊😊	😊😊	😊
TextGrid	😊	😊	😊	😊	😊	😬

4.5 The Registry

In the distributed computing environment, applications that are available as web services are numerous and are growing in number. Users find it more and more difficult to find interesting web services to create their workflows. There is a need to have a list of existing web services. However, a list is not enough, web services need to be discovered and searched by users not only by their names, but for their descriptions, inputs, outputs, versions, etc. All this information encoded as metadata should be used to allow complex search queries. To this end, registries are developed to help users in their research for web services.

4.5.1 UDDI

The need for a global registry for discovering and cataloguing web services brought Arriba, Microsoft and IBM together to work on a project called the Universal Description, Discovery and Integration [UDDI³³].

UDDI is conceptually divided in three parts: first part stores the basic information about a service and the organization providing the service. The second part stores information using a taxonomy classification and the third part is used to store technical information about the service such as its location and binding.

All the information of the UDDI can be searched and retrieved using a web service API.

Both UDDI and ebXML are maintained by OASIS (Organization for the Advancement of Structured Information Standards). However, it was reported in a CLARIN web services workshop that both suggestions are not widely used in the research community.

4.5.2 Feta

Feta is a semantic discovery tool developed by myGrid team that can be used to search available services and find those that best match the requirements of the user. It is used combined with the standard web services registry, namely UDDI.

³³ <http://uddi.xml.org/>

Feta search system allows the user to discover services based on metadata regarding the name of the application involved, operations, inputs, outputs, etc. It helps to find other web services with similar functionality so the user can choose or even replace a non available web service for another one.

Feta can be downloaded as a plug-in to the Taverna 1.7.x Workbench and used to search over services that have been annotated with the myGrid ontology. There is no Feta plug-in for Taverna 2.1 Workbench, where the similar functionality will be provided by the BioCatalogue plug-in.

4.5.3 BioCatalogue

The main deliverable of the [Web Services for Life Sciences](#) project was a registry of curated biological Web Services where users, researchers and curators can register, annotate and monitor Web Services. The registry, called BioCatalogue³⁴, is a BBSRC funded project and has been running since 1st June 2008. The project is a joint venture between the EMBL-EBI (led by Rodrigo Lopez) and the myGrid project at the University of Manchester.

The BioCatalogue wants to be the single registration point for Web Service providers and the place where researchers look for services. All this community of experts, users, providers, etc. can meet, contact, and discuss using the BioCatalogue portal. Putting together expert curators, the web services and the users will provide monitor and catalogue and high quality annotations.

Features

[Web service discovery](#)

- Handles service discovery by keywords or browsing the latest services submitted;
- Handles service filtering by tags on services, operations, inputs, and outputs, as well as filtering by providers, submitters, and locations.
- Exposes an OpenSearch description document.
- Possibility to search the BioCatalogue with your data.

[Web service annotation](#)

- Supports annotation of services by tags, user comments and text description.
- Annotated services or their parts (operations, inputs, outputs, etc.) are searchable.
- Ability for users to provide many more types of annotations (like examples, name aliases, etc.).
- Harvests all annotations and services from Feta.
- Annotation can take the form of **free text**, tags, **terms from selected ontology** and examples values.

[Web service submission](#)

- Supports registration for SOAP, REST and Soaplab services.

[Web service monitoring](#)

³⁴ <http://www.biocatalogue.org/>

- Monitor the WSDL document and endpoints of services (BioCatalogue checks that the service endpoint responds to simple requests).

Users

- Supports user registration and user profiles for both regular users and service providers.
- Supports Service rating.
- Provides a user-friendly web 2.0 interface showed in Figure 20 (appendix 9.2.5).

Source code

The BioCatalogue source code is also free under the **BSD License**³⁵.

BioCatalogue is a **Ruby on Rails** application.

Future work

These are some of the new features still to come for BioCatalogue:

- **Support for DAS services** registration
- **Integration with myExperiment:** on one hand services from BioCatalogue will be displayed in myExperiment so users of myExperiment can browse them. On the other hand services within workflows in myExperiment will reference the appropriate services in BioCatalogue so users can click through to the information in BioCatalogue.
- **Integration to Taverna:** users will be able to import Web Services and annotations to and from BioCatalogue and Taverna.
- **Public APIs:** RESTful APIs to BioCatalogue.
- **myBioCatalogue:** setup your private BioCatalogue for your project or organization.

More detailed information can be found in the [Biocatalogue 2010 roadmap](#).

4.5.4 Concluding remarks. Comparative analysis and recommendations.

Biocatalogue is clearly a good option to be used as a registry (specially combined with the rest of the myGrid tools). It's the one with the best user interface (a well designed and user-friendly web portal) and much more features.

Table 24: registry comparative analysis

	Func-tionali-ties	Usability	Cost / learning curve	Integrability	maturity	Support and plans for the future	Availability
UDDI							
FETA							

³⁵ Terms of use: <http://beta.biocatalogue.org/termsfuse>

BioCatalogue	 	 				 	
--------------	---	---	---	---	---	---	---

4.6 Wrappers

In the NLP world there are many existing tools already developed that could be useful to be accessed as a web service. These tools, usually called using the command line, need that someone with web services programming skills develops the necessary code (in JAVA or others) to deploy that tool as a web service.

Wrappers are tools designed to solve this situation. A wrapper is a set of tools which let a user with a few or none programming skills easily deploy some tools as a web service.

4.6.1 Soaplab

[SoapLab](#) and its new version [SoapLab2](#) are a set of tools to deploy as web services already existing command line tools. Soaplab was designed for sets of similar tools such as the European Molecular Biology Open Software Suite (EMBOSS).

Every EMBOSS program has different inputs, parameters and outputs to carry out its specific functions. Soaplab makes use of a special description file called ACD (Ajax Command Definitions) which describe the command line for every program. It can specify the inputs, outputs (using files or not), mandatory or optional parameters, etc. ACD file can even store some help information of every parameter that can be recovered later for Soaplab clients to help users. Using the necessary metadata most command line tools can be described using an ACD file.

In the end, Soaplab services share a common API disregarding all specific tool idiosyncrasies like programming language, operating system, command line syntax, etc.

- To this aim Soaplab is built on metadata, API and a distributed architecture. **Metadata:** Soaplab uses metadata in order to describe individual tools in detail.
 - description, type, and provider of the given analysis tool.
 - names and types of the input data and command-line parameters.
 - names and types of the resulting output data.
- The main Soaplab **API** allows the client
 - To determine the analysis type, category and all its metadata.
 - To send input data and parameters to the analysis.
 - To run the analysis.
 - Synchronously (by blocking the request until the analysis finishes)
 - Asynchronously (by creating a session identifier that can be later used by polling the server for the status and results).

- To retrieve current analysis status, including various notification messages (if implemented).
- To retrieve data results.

Metadata is the basis of Soaplab. A Soaplab2 service provider does not need to program anything but it needs to know how to describe its services by using metadata.

Supported protocols

The Web Services can be (unfortunately) accessed using several various protocols, or several flavours of the same SOAP protocol. The chosen protocol also determines what Java toolkit can be used (not every toolkit has support for every protocol).

- At the moment, the main protocol is the **SOAP** protocol, using the *document/literal* flavour (jax or jaxws java toolkit).

- Soaplab1: Soaplab2 also supports the SOAP protocol, using the *rpc/encoded* flavour. (axis1 java toolkit).

Soaplab developers claim that if there is a demand for it, Restful interfaces could be added to Soaplab:

“It would be relatively easy to add also the REST protocol (thus removing the SOAP layer completely). If there is a demand for it, let us know please.”

Deployment

Once Soaplab2 and Tomcat server are installed there are only three basic steps to deploy web services using Soaplab:

- 1- create your ACD files
- 2- generate³⁶ XML by calling: `ant gen`
- 3- deploy calling: `ant jaxdeploy` (or `ant axis1deploy`, depending of the protocol of your choice).

Workflows

[Taverna](#) workflow editor is an important tool for accessing Soaplab services. It uses a plug-in to extract metadata from the Soaplab service providing the user with extra information and capabilities. For example the polling support to adjust the timeouts of every Soaplab service. Only Taverna-1.7 is supported, Taverna-1.6 and Taverna-2.x are not supported.

License

Soaplab is free software under an [Apache License, Version 2.0](#).

Soaplab Typed Interface

³⁶ Because there are some doubts about executing this process under Windows, some research on this point is foreseen in the workplan.

Soaplab's generic interface (only one WSDL for all services) makes it possible to access any Soaplab web service regardless of the command line interface of underlying programs. This is very interesting for client developers. However it is not possible to specify input/output data types as part of Soaplab generic WSDL; instead the interface uses special methods to query this information from the XML. This difference from common WSDL interfaces, for example, doesn't allow standard Web Services clients to check/validate input data before sending a request or output data after a response has been received.

Bioinformatics Web Services community thought that Soaplab should include input/output type descriptions at WSDL level. To this end the Soaplab typed interface was developed. It can be easily activated and it provides a WSDL and type descriptions for every single web service. However, Soaplab clients and plug-in can still be used and benefit from all XML metadata.

EMBOSS Standard Groups

When developing an Emboss Soaplab web service the developer is strongly encouraged to fill a special attribute named *groups*. This attribute must be filled with an EMBOSS Standard Groups possible value. Table 30 (appendix 9.2.1) shows the different possibilities to classify an EMBOSS tool or web service. These metadata, like all the ACD file data, is stored in the XML metadata file. Later, for example, it can be used by FETA discovery plug-in for Taverna to find web services.

UPF Soaplab2 example

Some experiments have been developed here at UPF to try Soaplab2 functionalities and costs. The objective was to deploy a few web services and create a workflow with them. A step by step roadmap is presented:

- Soaplab2 and necessary software installation
- ACD metadata definition (for all tools)
- Server deployment
- Web service test using Soaplab web client Spinet
- Taverna workflow editor installation
- Soaplab2 plug-in for Taverna installation
- Workflow development and test

Soaplab helps the developer to deploy web services without having a deep knowledge about web services. Learning to use ACD metadata files is easy and there are very simple examples to test. Spinet web client allows the developer to test the deployed web services. *Figure 11: splitter WS* and *Figure 12: Freeling WS* show the Spinet web client used to test the Soaplab implementation of the splitter and Freeling tools.

- ▶ *Figure 13: Test workflow* shows the developed chain:
 - **Get_p**: extracts plain text from JR-Acquis corpus (TEI)
 - **Splitter**: simple sentence splitter

- **Words_per_line_filter**: max allowed words per sentence filter
- **Freeling_file**: Freeling WS.

Figure 14: detailed test workflow shows all the ports (inputs, outputs and parameters) and their connections.

Soaplab is a user-friendly and easy to install set of tools which can be very helpful to deploy web services. The small learning curve combined with the small amount of work needed to deploy a web service make this wrapper a really interesting option to help non skilled service providers.

4.6.2 Concluding remarks. Comparative analysis and recommendations.

Wrappers can be very useful tools combined with the usual way of deploying web services (developing JAVA code for example). From the PANACEA project point of view, having two different options to deploy tools as web services is very interesting. Service providers can then decide what option is better for them considering their resources.

4.7 Sharing research objects (Workflows, ontology, etc.)

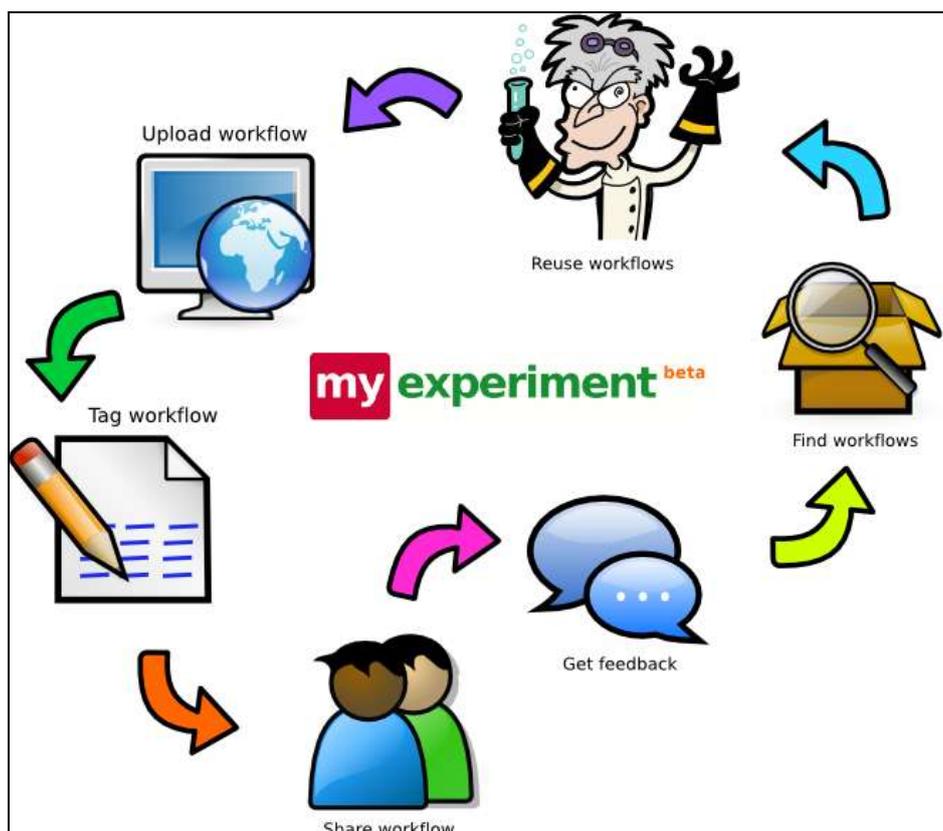
Deploying tools as web services can be very useful for others researchers. With a growing community of users, researchers, service providers there is a growing need to share information and research files. Sharing already developed workflows can be a considerable reduction of duplicate work. Ontologies, vocabularies, schemas, etc. are all research objects that need a place to be shared and discussed. Having a unique ‘point of entry’ where users can easily find all necessary documents and files to start their work will help to create a bigger community.

4.7.1 myExperiment

MyExperiment is collaborative web portal where researchers can create groups, exchange information, discuss about topics and share workflows. All shared objects are sorted and searched easily using user-friendly web 2.0 interfaces (*Figure 21: myExperiment main page*).

MyExperiment is developed by a [joint team](#) from the University of Southampton and The University of Manchester in the UK, led by [David De Roure](#) and [Carole Goble](#), and is funded by [JISC](#) and supported by EPSRC as part of the [myGrid](#) and [e-Research South](#) consortia and by Microsoft's [Technical Computing Initiative](#).

Figure 4



The myExperiment services are accessible through simple RESTful programming interfaces so they can easily be invoked from other web sites, wikis, etc.

Users can execute workflows directly from the myExperiment portal. To do so users need access to a remote Taverna server.

MyExperiment is downloadable for free and any institution, research group, etc. can run their own myExperiment instance.

The [source code](#) is maintained on RubyForge and is available under the **BSD licence**.

4.7.2 Concluding remarks. Comparative analysis and recommendations.

Considering the relatively small amount of partners and users inside the PANACEA project developing a new very sophisticated portal or tool to share workflows could be not necessary. However, it could be very interesting for the PANACEA factory future to have a good portal (PANACEAexperiment for example) where users could create communities, share workflows and discuss about research. From this point of view, adopting the myExperiment model could be a low cost solution that could provide PANACEA users with very mature and already developed solution.

Another interesting aspect of the myExperiment regarding PANACEA project is the possibility to execute workflows directly from the website. This means a new or non-expert user could test a workflow without having to install anything on his/her computer. On the other hand, an expert user could develop very complex workflows using Taverna and get feedback from others users so the workflow can be improved.

4.8 Relevant projects

4.8.1 KYOTO project

The goal³⁷ of the KYOTO project (ICT-211423) is to develop an information and knowledge sharing system that relates text in various languages to a shared ontology in such a way that it enables the extraction of deep semantic relations and facts from text in a specific domain and for a closed set of languages: English, Dutch, Italian, Spanish, Basque, Mandarin Chinese and Japanese³⁸. We surveyed KYOTO because this distributed architecture and its data formats could be interesting for PANACEA design when working with the second version of the travelling object.

The KYOTO system

The KYOTO system aims at establishing communication and interpretation across languages and cultures and (at) supporting the building and maintaining the system by groups of people in a shared domain and area of interest. The system consists of 4 main components³⁹:

- WikiPlanet: a semantic media Wiki for collecting and sharing textual information in a community;
- KyotoCore: pipeline architecture of modules for processing text documents for term and concept extraction and for text mining;
- Wikyoto: Wiki platform for editing domain terms and concepts across different languages and cultures;
- DebVisDic platform: database system for storing the Wordnets and the central ontology;

The goal of KYOTO is a system that allows people in communities to define the meaning of their words and terms in a shared Wiki platform so that it becomes anchored across languages and cultures but also so that a computer can use this knowledge to detect knowledge and facts in text. KYOTO will represent this knowledge so that a computer can understand it.

The KYOTO system in 6 steps⁴⁰

1. People from a domain specify the locations of diverse and distributed sources of knowledge in different languages. They can do this through Wikiplanet.
2. The text in various languages is captured from the sources and offered to the KYOTO system
3. Term yielding robots (so-called Tybots) automatically extract all the important terms and possible semantic relations and relate these to existing semantic networks (Wordnets) in each language.

37 See Deliverable D10.1 in www.kyoto-project.eu

38 KYOTO is focused to the domain of the environment and specifically to the topic of ecosystem services, but the system is designed to be used for any language and to be applied to any domain.

39 See Deliverable D10.2 in www.kyoto-project.eu

40 Annual report 2009, www.kyoto-project.eu

4. The Wikyoto (wiki-environment) allows the domain people to maintain the terms and concepts and agree on their meaning within the community and across languages.⁴¹
5. Kybots use the terms and knowledge to detect factual data in the text in various languages.
6. The factual data is indexed and can be accessed by anybody through semantic search, again in various languages

Some useful KYOTO tools

The following tools are the ones that can be useful at the beginning of PANACEA project, just to have an idea of how similar problems (for instance multi word expressions) have been addressed. The Tybot has been cited because it represents an example of simple parser of the KAF structure (see 4.8.1.1). Indeed, having a database is useful in the PANACEA platform for managing, storing large amounts of data in a little response time.

- The Kyoto Multiword Tagger detects multi word expressions in the output of the parser and restructures the KAF according to the token span. Multi word expressions are thus seen as groups of token IDs.

- The Kyoto Tybot extracts terms from the KAF representations and stores these in a term database.

- The Kyoto Ontotagger inserts ontological information extracted from a knowledge base into the document's KAF structure. Ontotagger has been cited as it can be transformed into a web service with Soaplab.

4.8.1.1 The Kyoto Annotation Framework (KAF)

In the project, information is encoded according to the Kyoto Annotation Framework (KAF). The format defined by KAF is compatible with the Linguistic Annotation Framework (LAF) (Ide and Romary, 2003) but imposes additional, more specific annotation standardisations due to the specific domain KYOTO deals with. KAF stores morphosyntactic and semantic annotations. It is realised by implementing dialects of the existing ISO standards MAF (morphosyntactic annotation) (Clément and de la Clergerie 2005) and SynAF (syntactic annotation) (Declerck 2006).

The root element of a KAF file is `<KAF>` which has one element `xml:lang` defining the document language:

```
<KAF xml:lang="en"> ... </KAF>
```

The document header is marked by the optional, but recommended `<kafHeader>` tag. It can contain the elements `<fileDesc>`, `<public>` and `<linguisticProcessors>`. The latter has one or more `<lp>` elements describing the linguistic processors used to produce the document. The header elements' possible attributes are shown in this example extracted from (Agirre et al., 2009):

```
<kafHeader>
  <fileDesc title="3_3012" author="WWF" filename="KYOTO_3_3012" file-
type="PDF" pages="19"/>
```

41 The meanings are formalized in a domain ontology which can be used by computer programs.

```

<public publicId="3_3012" uri="http://kyoto.org/docs/KY0T0_3_3012.pdf" />
  <linguisticProcessors layer="text">
    <lp name="Freeling" version="2.1" timestamp="2009-06-25T10:05:00Z"/>
  </linguisticProcessors>
</kafHeader>

```

The document's actual content is tagged by the `<text>` element. Within that, the word forms are tagged by the `<wf>` element with an obligatory *wid* attribute (word id) and some optional attributes: *sent* (sentence id), *para* (paragraph id), *page* (page id), and *xpath* (XML xpath expression). An example from (Agirre et al., 2009):

```

<text>
  <wf wid="w1" sent="s1" para="p1">John</wf>
  ...
</text>

```

Terms (`<term>` elements) group word forms and provide information about type (open category, closed category, or entity), named entity type if the term is a named entity (type *netype*), lemma, part of speech (attribute *pos*), case and the head word's id if the term is a compound. All the term definitions are enclosed by a `<terms>` tag and followed by a `` tag that defines the word forms that are part of the given term.

Additionally, a term can be assigned to external resources such as knowledge bases or ontologies in the `<externalReferences>` context. Each resource is declared by the `<externalRef>` tag comprising the attributes *resource* (the resource id), *reference* (the resource code), and *confidence* (a confidence weight between 0 and 1).

The `<component>` element assigns a compound to a term. It provides information about identifier (attribute *id*), lemma (attribute *lemma*), part of speech (attribute *pos*), and case (attribute *case*). An example adapted from (Agirre et al., 2009) (note that the content given here as an example does not necessarily make sense):

```

<terms>
  <term tid="t1" type="entity" lemma="John" pos="R" netype="person">
    <span>
      <target id="w1"/>
    </span>
    <externalReferences>
      <externalRef resource="WN-1.7" reference="ENG-17-00861095-v" confidence="0.80"/>
      <externalRef resource="WN-1.7" reference="ENG-17-00859568-v" confidence="0.20"/>
      ...
    </externalReferences>
  </term>

```

```

</externalReferences>

</term>

...

</terms>

```

Additional relations can be defined in similar fashions: Dependency relations (<deps>), Chunks (<chunk>), events (<events>), quantifiers (<quantifiers>), and time expressions (<timexs>) according to the Timex specification (Lee et al., 2007). The definitions work essentially in the same ways as in the elements presented in the examples, e.g. each dependency relation within a <deps> environment is defined by a <dep> tag. The full KAF reference is given by (Agirre et al., 2009).

4.8.2 ACCURAT

ACCURAT is a Collaborative project funded within FP7-ICT-2009-4 call and action ICT-2009.2.2: **Language-based interaction** under Grant agreement no. 248347, cf. www accurat-project.eu

The aim of the ACCURAT project is to research methods and techniques to overcome the lack of linguistic resources for under-resourced areas of machine translation. The main goal is to find, analyze and evaluate novel methods that exploit comparable corpora in order to compensate for the shortage of linguistic resources, and ultimately to significantly improve MT quality for under-resourced languages and narrow domains.

One focus in this project is to define methods and tools to represent comparable corpus data, and provide measures for comparability. In this context, the project proposed an extension to the CES standard in two ways:

One deals with the representation of the single texts, and includes an extended source description to cover information about genre, domain, encoding, and cleaning-techniques to clean the texts, as well as the original HTML document (as the HTML structure may be relevant to determine comparability).

Figure 5: ACCURAT proposal for extension of CES header

```

<sourceces>
  <biblstruct>
    <monogr>
      <h title>Title of the text</h title>
      <h author>Author of the text</h author>
      <edition></edition>
      <imprint></imprint>
      <bibnote></bibnote>
    </monogr>
  </biblstruct>
</sourceces>
<extendedsourceces>
  <genre>newswires</genre>
  <domain>international news</domain>
  <publicationresource>http://news.kathimerini.gr/4dcgi/_w_articles_world_2_19/12/2007_253016</publicationresource>
  <encoding>utf-8</encoding>
  <publicationdate>19/12/2007</publicationdate>
  <textcleaningnote>short description about the process used to clean the text and any other relevant information from the HTML source
</textcleaningnote>
</extendedsourceces>
</filedesc>
<cesheader>
  <text>
    <body>Cleaned text</body>
  </text>
  <htmlsource>Html source with entities encoded</htmlsource>
</cesdoc>

```

The other extension focuses on an extension of the alignment, and proposes an extension of attributes in the linkGrp tag such that the alignment level (values: *parallel* / *strongly comparable* / *weakly comparable*) and the alignment decision could be expressed:

Figure 6: ACCURAT proposal for extension of the linkGrp tag

```
<?xml version="1.0" encoding="UTF-8"?>
<cesalign version="1">
  <linklist>
    <linkgrp targtype="doc" alignmentlevel="strongly comparable" alignmentdecision="based on
    MTurk evaluation">
      <link xtargets="el-en_newswires_international-news_3C321_el.xml; el-
      en_newswires_international-news_3C321_en.xml"></link>
    </linkgrp>
  </linklist>
</cesalign>
```

Once an automatically computable metric for the comparability of corpora / paragraphs / sentences will be available, this may form another attribute in the linkGrp tag.

Both proposals are extensions to the existing CES standard, required to deal with comparable corpora. The CES standard and these extensions could be useful or at least be used as a guideline for PANACEA corpus representation.

5 PANACEA Platform requirements

The aim of this section is to list the requirements for the Platform and tools that deal with the technology needed to deploy the factory, its usability and interoperability to determine if the PANACEA service portfolio could meet the users' needs.

As agreed on the technical meeting in Athens (April 2010), the PANACEA platform user requirements will be adopted directly from the work developed in Workpackage 8 (Evaluation in industrial environments) in its first deliverable D8.1. On the other hand, the functional requirements will be adopted from the work developed in Workpackage 7 (Evaluation of components integration and produced resources) in deliverable D7.1.

6 PANACEA Platform design

This section is devoted to design the whole PANACEA platform: choose the necessary technologies, choose alternatives, design the data format and guidelines, etc.

The section is divided as follows:

- Travelling object. Corpus and data format.
- Common interfaces design.
- Panacea Platform technologies.

6.1 Travelling object. Corpus and data format

6.1.1 Introduction

This is a proposal on the format of the PANACEA corpora files (a.k.a. travelling objects) to be annotated by, and exchanged between, the different tools of the PANACEA factory.

It has been observed that the LRs and Technologies community has not reached a consensus in defining an encoding for annotated corpora, while relevant efforts have resulted in proposals that have not been used widely. On the other hand, there is the need to harmonize the output of the large variety of tools to be integrated in the PANACEA factory, and in doing so it would be nice if we could avoid reinventing the wheel, at least in some aspects of the definition of this output. Thus, this document is based on, among other sources:

- PANACEA partners' descriptions of tools and encodings they already use⁴²
- Gr. Thurmair's (LG) "Proposal for corpus representation in PANACEA"⁴³
- the XCES Corpus Encoding Standard⁴⁴.
- informal communications concerning similar efforts in the Accurat project
- the KAF from the KYOTO project

In a technical meeting held on April 15th, PANACEA decided to proceed stepwise before deciding the final format of the Travelling Object. It was decided to use the minimal common vertical in-line format used by WP4 tools for the first version of the platform (t14), and to carefully decide on further extensions after having achieved a first common definition.

In the following sections of this document, we propose encodings for the output of the tools for corpus acquisition and boilerplate removal, text processing and alignment, which are the ones to be delivered in t14. A revision and extension of this proposal is planned (see section 7).

6.1.2 Crawling and boilerplate removal

In this section, we propose encodings for the output of the tools for corpus acquisition and boilerplate removal. In this and the following sections, we will use two web pages⁴⁵ (and their derivatives) as examples in English and Spanish. The two web pages focus on the same international news, i.e. the EU aid for Haiti after the 2010 earthquake.

English (http://ec.europa.eu/news/external_relations/100218_en.htm)

```

1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
2 <html lang="en">
3   <head>
4     <META http-equiv="Content-Type" content="text/html; charset=UTF-
      8">
5     <meta name="Title" content="Haiti on our minds">
6     <meta name="Creator" content="">
7     <meta http-equiv="Content-Language" content="en">

```

⁴² <http://projectmanagement.panacea-lr.eu:9950/projects/panacea-project/conversations/17>

⁴³ <http://projectmanagement.panacea-lr.eu:9950/projects/panacea-project/conversations/12>

⁴⁴ <http://www.xces.org>

⁴⁵ Both "web pages" include the title, some sentences, and some boilerplate text from the actual web pages on the European Commission news site. Sentences are modified versions of the original, for exemplification purposes.

```

8      <meta name="Type" content="57">
9      <meta name="Classification" content="26000">
10     <meta name="Keywords"
      content="EU, Europe, European, commission, Haiti, earthquake, homeless, aid, ass
      istance, humanitarian, response, relief, shelter, hurricane, rainy, season, fund
      ing, support, ECHO, donors, conference, rebuilding, reconstruction, gendarme, po
      lice, military">
11     <meta name="Description" content="Shelter seen as the top
      priority as EU ups aid to Haiti">
12     <meta name="Date" content="18/02/2010">
13     <title>Haiti on our minds</title>
14     </head>
15     <body>
16     <h1>Haiti on our minds</h1>
17     <p>Commission calls for €90m more in aid for the quake-stricken
      country. This amount will be drawn from EU emergency funds. </p>
18     <div><a href="notice.html">Legal notice</a> | <a
      href="#top">Top</a></div>
19     </body>
20 </html>

```

Spanish (http://ec.europa.eu/news/external_relations/100218_es.htm)

```

1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
2 <html lang="es">
3   <head>
4     <META http-equiv="Content-Type" content="text/html; charset=UTF-
      8">
5     <meta name="Reference" content="EUROPA/">
6     <meta name="Title" content="La UE enviará más ayuda a Haití">
7     <meta name="Creator" content="">
8     <meta http-equiv="Content-Language" content="es">
9     <meta name="Type" content="57">
10    <meta name="Classification" content="26000">
11 <meta name="Keywords"
      content="UE, Europa, Europea, Comisión, Haití, terremoto, personas sin
      hogar, ayuda, asistencia, humanitaria, respuesta, auxilio, refugio, huracán, de
      lluvias, estación, financiación, apoyo, ECHO, donantes, conferencia, reconstruc
      ción, gendarme, policía, militar">
12    <meta name="Description" content="Los refugios, máxima prioridad
      de las ayudas de la UE a Haití">
13    <meta name="Date" content="18/02/2010">
14    <title>La UE enviará más ayuda a Haití</title>
15    </head>
16    <body>
17    <h1>La UE enviará más ayuda a Haití</h1>
18    <p>La Comisión pide otros 90 millones de euros de los fondos de
      emergencia europeos.</p>
19    <div><a href="aviso.html">Aviso jurídico</a> | <a
      href="#top">Comienzo</a></div>
20    </body>
21 </html>

```

We will assume that the PANACEA bilingual crawler has fetched both pages and stored them⁴⁶ in a local repository. The basename convention for the html file will be “Domain + _ + YYYYMMDD

⁴⁶ We assume that the two pages will be both logged as candidates for extraction of parallel sentences. In the case of monolingual crawling, no similar information will be logged.

+ _ + ShortTitle + _ + Lang”, where YYYYMMDD is the date the data was crawled and stored in the repository. Thus our 2 example documents will be stored as news_20100514_haiti_en.html and news_20100514_haiti_es.html.

For each locally stored web page, the crawler will also create an XML file with the extension `basic.xml`. This file will contain a header with some automatically extracted metadata and a link to the html file, as in the next listing for the Spanish document.

```

1 <?xml version="1.0"?>
2 <cesDoc id="news_20100514_haiti_es" version="0.4"
  xmlns="http://www.xces.org/schema/2003">
3   <cesHeader version="0.4">
4     <fileDesc>
5       <titleStmt>
6         <title>La UE enviará más ayuda a Haití</title>
7         <respStmt>
8           <resp>
9             <type>Crawling</type>
10            <name>Panacea partner</name>
11          </resp>
12        </respStmt>
13      </titleStmt>
14      <sourceDesc>
15        <biblStruct>
16          <monogr>
17            <author>EU web author if available</author>
18            <imprint>
19              <publisher>EU</publisher>
20              <pubDate>2010-02-20</pubDate>
21              <eAddress
type="web">http://ec.europa.eu/news/external_relations/100218_es.htm</eAdd
ress>
22            </imprint>
23          </monogr>
24        </biblStruct>
25      </sourceDesc>
26    </fileDesc>
27
28    <profileDesc>
29      <langUsage>
30        <language iso639="es"/>
31      </langUsage>
32      <textClass>
33        <keywords>
34          <keyTerm>Comisión</keyTerm>
35          <keyTerm>Haití</keyTerm>
36          <keyTerm>terremoto</keyTerm>
37          <keyTerm>. . .</keyTerm>
38        </keywords>
39        <domain>International News</domain><!-- or (automotive,
environment, legal)-->
40        <subdomain>Optional information on subdomain</subdomain>
41        <subject>Optional information on the subject</subject>
42      </textClass>
43      <annotations>
44        <annotation ann.loc="news_20100514_haiti_es.html"
type="htmlsource"/>
45      </annotations>
46    </profileDesc>
47  </cesHeader>
48 </cesDoc>

```

As shown on the listing above this is a `cesDoc` that can be validated against the already available XCES standard schemas. All metadata for this file is contained inside a `cesHeader` element⁴⁷. Extensive documentation for this and all other element in the XCES standard can be obtained from the XCES site. Here we can briefly discuss some crucial subelements of the header.

- The `<fileDesc>` element can be used for information about the title of the document and any annotations added. The `<sourceDesc>` subelement can be used for information on the original author and publication date, the publisher of the document, and the URL it was downloaded from. One or more `<respStmt>` subelements can be used to describe operations and people/groups responsible for these operations on this particular document.
- The `<profileDesc>` element groups information describing the language(s) of the document (`<langUsage>`) and the nature or topic of a text (`<domain>`, `<subdomain>`, `<subject>`, `<keywords>`).
- The `<annotations>` subelement of the `<profileDesc>` can be used for storing links to other documents relevant to this basic version. In the example above, a link to the original html document is shown. In future versions of the Panacea travelling object, this element can be used to include links to files with stand-off annotation.

Assuming that the html source has been cleaned from a boilerplate removal tool as described in the corpus acquisition process of D4.1, the `basic.xml` file will also contain the paragraph-segmented textual content of the HTML pages as in the following listing:

```

1 <?xml version="1.0"?>
2 <cesDoc id="news_20100514_haiti_es" version="0.4"
   xmlns="http://www.xces.org/schema/2003">
3 <cesHeader>
4 <!-- . . . -->
5 <!-- We add another respStmt for the cleaning. Everything else as in the
   header above -->
6   <respStmt>
7     <resp>
8       <type>Boilerplate removal, text extraction, paragraph detection,
       etc.</type>
9       <name>Panacea partner</name>
10    </resp>
11  </respStmt>
12 </cesHeader>
13 <!-- . . . -->
14 <!-- We add a text and a body element for storing the clean text. These are
   necessary elements so that this file can be validated against XCES
   schemas. -->
15 <text>
16 <body>
17 <p id="p1" type="title">
18 La UE enviará más ayuda a Haití
19 </p>
20 <p id="p2">
```

⁴⁷ It should be noted that similar headers document manually and automatically annotated files in large corpora like the American National Corpus. A similar header has also been proposed for accompanying comparable corpora in the Accurat project.

```

21 La Comisión pide otros 90 millones de euros de los fondos de emergencia
    europeos.
22 </p>
23 </body>
24 </text>
25 </cesDoc>

```

In this version of the basic.xml file, we can either keep the paragraph elements `<p>` from the HTML source, or perform paragraph detection using a specific tool. To each paragraph element we add an obligatory `id` attribute whose values follow the convention `p1,p2,...,pN`. If the paragraph language has been detected as different from the main language of the document, an optional `lang` attribute is added, whose values are ISO 639-1 two letter language codes. If needed, an optional `type` attribute is also added. The value of this attribute is indicative of the paragraph's function in the text, for example `title`. Another optional attribute for `<p>` elements is `topic`, to be used in cases where the topic has been detected as different from the one described in the `<domain>`, `<subdomain>`, `<subject>` elements of the header.

The `basic.xml` document will be the starting point for the rest of the processing tools.

6.1.3 Text processing

In this section, we propose encodings for the output of the tools for text processing.

6.1.3.1 Sentence splitting and tokenization

Operations by the NLP tools assumed at this stage:

- Paragraphs are split into sentences
- Sentences are tokenized

The output of these tools should be saved in an XML file that will follow the same basename conventions and will have the extension `tok.xml`.

In this file, we add a sentence element `<s>` which has an obligatory `id` element and an optional `lang` attribute, if the sentence language is different from the main language of the document. Another optional attribute for `<s>` elements is `topic`, to be used in cases where the topic has been detected as different from the one described in the `<domain>`, `<subdomain>`, `<subject>` elements of the header. As it has been decided in the Panacea Athens technical meeting, the tokens for each sentence will be included in a one token per line fashion inside the sentence they belong to⁴⁸.

```

1 <?xml version="1.0"?>
2 <cesDoc id="news_20100514_haiti_es" version="0.4"
    xmlns="http://www.xces.org/schema/2003">
3 <cesHeader>
4 <!-- . . . -->
5 <!-- We add another respStmt for sentence splitting and the tokenization.
    Everything else as in the header above -->
6 <respStmt>
7 <resp>
8 <type>Sentence splitting and tokenization.</type>
9 <name>Panacea partner</name>

```

⁴⁸ Please notice that although such a file can still be validated against the XCES schemas, this file is no longer a proper XCES document, since the latter standard targets stand-off annotation.

```

10     </resp>
11     </respStmt>
12 </cesHeader>
13<!-- . . . ->
14<!-- We add sentences and tokens in a verticalized format. -->
15 <text>
16   <body>
17   <p id="p1" type="title">
18   <s id="s1">
19   La
20   UE
21   enviará
22   más
23   ayuda
24   a
25   Haití
26   </s>
27   </p>
28   <p id="p2">
29   <s id="s2">
30   La
31   Comisión
32   pide
33   otros
34   90_millones
35   de
36   euros
37   de
38   los
39   fondos
40   de
41   emergencia
42   europeos
43   .
44   </s>
45   </p>
46 </body>
47 </text>
48 </cesDoc>

```

6.1.3.2 POS Tagging and lemmatization

Operations by the NLP tools at this stage:

- Each token is assigned a tag conveying POS + morphosyntactic descriptions
- Each token is optionally assigned a lemma

The output of these tools should be saved in an XML file that will follow the same basename conventions and will have the extension `tag.xml`.

Each token in this file will be accompanied by a tag conveying POS and morphosyntactic information, and optionally a lemma. Tokens, tags and lemmas will be separated by tabs.

```

1 <?xml version="1.0"?>
2 <cesDoc id="news_20100514_haiti_es" version="0.4"
   xmlns="http://www.xces.org/schema/2003">
3 <cesHeader>
4 <!-- . . . ->
5 <!-- We add another respStmt for tagging and lemmatization. Everything else
   as in the header above -->

```

```

6     <respStmt>
7     <resp>
8     <type>Tagging and lemmatization.</type>
9     <name>Panacea partner</name>
10    </resp>
11    </respStmt>
12 </cesHeader>
13<!-- . . . ->
14<!-- We add tags and lemmas in a verticalized format. -->
15 <text>
16   <body>
17   <p id="p1" type="title">
18   <s id="s1">
19   La AFS el
20   UE N4666 UE
21   enviará VDU3S- enviar
22   más D más
23   ayuda N5-FS ayuda
24   a P a
25   Haití N4666 Haití
26   </s>
27   </p>
28   <p id="p2">
29   <s id="s2">
30   La AFS el
31   Comisión N4666 Comisión
32   pide VDR3S- pedir
33   otros EN--66 otro
34   90_millones X 90_millones
35   de P de
36   euros N5-MP euro
37   de P de
38   los AMP el
39   fondos N5-MP fondo
40   de P de
41   emergencia N5-FS emergencia
42   europeos JQ--MP europeo
43   . SENT .
44   </s>
45   </p>
46   </body>
47   </text>
48 </cesDoc>

```

6.1.3.2.1 An alternative format for tokens

It could be the case that a PANACEA tool (like a word/chunk aligner) may need to refer to separate tokens in the tokenized or the tagged file. In that case, a better suggestion would be to have a `t` element for the representation of tokens, instead of a whitespace separated, one-token-per-line representation.

The `t` elements in such a file would be minimally composed by a properly valued `id` attribute (`t1`, `t2`, ..., `tN`), and a `word` element. In the case of tagged files, a `tag` and an optional `lemma` attribute are also included, as in the listing below:

```

1 <text>
2   <body>
3     <p id="p1" >
4       <s id="s1">
5         <t id="t1_1" tag="AFS" lemma="el" word="La"/>
6         <t id="t1_2" tag="N4666" lemma="UE" word="UE"/>
7         <t id="t1_3" tag="VDU3S-" lemma="enviar" word="enviará"/>

```

```

8     <t id="t1_4" tag="D" lemma="mas" word="mas"/>
9     <t id="t1_5" tag="N5-FS" lemma="ayuda" word="ayuda"/>
10    <t id="t1_6" tag="P" lemma="a" word="a"/>
11    <t id="t1_7" tag="N4666" lemma="Haiti" word="Haiti"/>
12    </s>
13  </p>
14  <p id="p2" >
15    <s id="s2">
16    <t id="t2_1" tag="AFS" lemma="el">La</tok>
17    <t id="t2_2" tag="N4666" lemma="Comisión">Comisión</tok>
18    <t id="t2_3" tag="VDR3S-" lemma="pedir">pide</tok>
19    <t id="t2_4" tag="EN--66" lemma="otro">otros</tok>
20    <t id="t2_5" tag="X" lemma="90_millones">90_millones</tok>
21    <t id="t2_6" tag="P" lemma="de">de</tok>
22    <t id="t2_7" tag="N5-MP" lemma="euro">euros</tok>
23    <t id="t2_8" tag="P" lemma="de">de</tok>
24    <t id="t2_9" tag="AMP" lemma="el">los</tok>
25    <t id="t2_10" tag="N5-MP" lemma="fondo">fondos</tok>
26    <t id="t2_11" tag="P" lemma="de">de</tok>
27    <t id="t2_12" tag="N5-FS" lemma="emergencia">emergencia</tok>
28    <t id="t2_13" tag="JQ--MP" lemma="europeo">europeos</tok>
29    <t id="t2_14" tag="SENT" lemma=".">.</tok>
30  </s>
31  </p>
32 </body>
33 </text>

```

Another advantage for such an XML representation for tokens is that it would be easy to add a new piece of information to each token without worrying about where to insert this information. For example, partners could opt for providing both a tool/language-specific tag and a mapped tag *mtag* according to a standard like Parole⁴⁹, as in the following example.

```

1 <t id="t2_9" mtag="Tdmp-" tag="AMP" lemma="el">los</tok>
2 <t id="t2_10" mtag="Ncmp-" tag="N5-MP" lemma="fondo">fondos</tok>
3 <t id="t2_11" mtag="Sps" tag="P" lemma="de">de</tok>
4 <t id="t2_12" mtag="Ncfs-" tag="N5-FS" lemma="emergencia">emergencia</tok>
5 <t id="t2_13" mtag="A-pmp-" tag="JQ--MP" lemma="europeo">europeos</tok>

```

6.1.3.3 Constituency and/or dependency parsing

Operations by the NLP tools at this stage:

- A constituency parser builds a phrase structure tree and/or
- A dependency parser builds a dependency tree.

6.1.3.3.1 Phrase Structure Trees

Assuming the XML representation discussed in 6.1.3.2.1 above, phrase structure trees can be represented as in the following example:

```

1 <body>
2   <p id="p1" >
3     <s id="s1">
4 <graph root="s1_500">
5   <terminals>
6     <t id="s1_1" tag="AFS" lemma="el" word="La"/>

```

⁴⁹ For the Parole tagset, see for example <http://www.lsi.upc.es/~nlp/tools/parole-eng.html>

```

7     <t id="s1_2" tag="N4666" lemma="UE" word="UE"/>
8     <t id="s1_3" tag="VDU3S-" lemma="enviar" word="enviará"/>
9     <t id="s1_4" tag="D" lemma="mas" word="mas"/>
10    <t id="s1_5" tag="N5-FS" lemma="ayuda" word="ayuda"/>
11    <t id="s1_6" tag="P" lemma="a" word="a"/>
12    <t id="s1_7" tag="N4666" lemma="Haiti" word="Haiti"/>
13  </terminals>
14  <non-terminals>
15    <nt cat="S" id="s1_500">
16      <edge idref="s1_501" />
17      <edge idref="s1_502" />
18    </nt>
19    <nt cat="NP" id="s1_501">
20      <edge idref="s1_1" />
21      <edge idref="s1_2" />
22    </nt>
23    <nt cat="VP" id="s1_502">
24      <edge idref="s1_3" />
25      <edge idref="s1_503" />
26      <edge idref="s1_504" />
27    </nt>
28    <nt cat="NP" id="s1_503">
29      <edge idref="s1_4" />
30      <edge idref="s1_5" />
31    </nt>
32    <nt cat="PP" id="s1_504">
33      <edge idref="s1_6" />
34      <edge idref="s1_505" />
35    </nt>
36    <nt cat="NP" id="s1_505">
37      <edge idref="s1_7" />
38    </nt>
39  </non-terminals>
40 </graph>
41 </s>
42 </p>
43 </body>

```

In this representation⁵⁰,

- Each sentence element contains a graph element.
- The graph element contains a `terminals` and a `non-terminals` element.
- The `terminals` element contains one or more `t` elements that represent tokens as in 6.1.3.2.1 above.
- The `non-terminals` element contains one or more `nt` elements.
- Each `nt` element has an `id` and a `cat` attribute. The value of the latter represents the constituent category depending on the output of the parser (typically *S*, *NP*, *VP*, etc.)

⁵⁰ This representation is based on the TIGER-XML format <http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/doc/html/TigerXML.html>

- The graph element has an attribute `root` whose value is the `id` of one of the `nt` elements (prototypically one `nt` of `cat S`).
- Each `nt` element contains one or more `edge` elements. Each `edge` element has an attribute `idref` that corresponds to terminal or non-terminal children nodes of the current `nt`. In the example above, the `s1_502 VP` has a terminal child `s1_3`, and two non-terminals `s1_503` and `s1_504`, an `NP` and a `PP`, respectively.

Notice that by using this format, it is easy to represent non-contiguous constituents, something that would not be straightforward if one decided to simply nest XML elements representing constituents.

In order to use this representation for the output of a chunker, we can generate a flat `non-terminals` section as in the following example:

```

1  <body>
2    <p id="p1" >
3      <s id="s1">
4        <graph root="s1_500">
5          <terminals>
6            <t id="s1_1" tag="AFS" lemma="el" word="La"/>
7            <t id="s1_2" tag="N4666" lemma="UE" word="UE"/>
8            <t id="s1_3" tag="VDU3S-" lemma="enviar" word="enviará"/>
9            <t id="s1_4" tag="D" lemma="mas" word="mas"/>
10           <t id="s1_5" tag="N5-FS" lemma="ayuda" word="ayuda"/>
11           <t id="s1_6" tag="P" lemma="a" word="a"/>
12           <t id="s1_7" tag="N4666" lemma="Haiti" word="Haiti"/>
13         </terminals>
14         <non-terminals>
15           <nt cat="S" id="s1_500">
16             <edge idref="s1_501" />
17             <edge idref="s1_502" />
18             <edge idref="s1_503" />
19             <edge idref="s1_504" />
20           </nt>
21           <nt cat="NP" id="s1_501">
22             <edge idref="s1_1" />
23             <edge idref="s1_2" />
24           </nt>
25           <nt cat="VP" id="s1_502">
26             <edge idref="s1_3" />
27           </nt>
28           <nt cat="NP" id="s1_503">
29             <edge idref="s1_4" />
30             <edge idref="s1_5" />
31           </nt>
32           <nt cat="PP" id="s1_504">
33             <edge idref="s1_6" />
34             <edge idref="s1_505" />
35           </nt>
36           <nt cat="NP" id="s1_505">
37             <edge idref="s1_7" />
38           </nt>
39         </non-terminals>
40       </graph>
41     </s>
42   </p>
43 </body>

```

In the above example, all but one non-recursive constituents (= chunks) are nested inside an artificially generated `S` element. Since the structure of these flat trees is the same with recursive trees generated by parsers, it can also accommodate nested chunks (NPs inside PPs as in the example above) in case a particular chunker produces them.

6.1.3.3.2 Dependency trees

Dependency trees can be represented at the token level in a format similar to the CoNLL data format⁵¹ as in the following example:

```

1 <s id="s1">
2   <!-- ....-->
3   <t id="t1_2" tag="N4666" lemma="UE" word="UE"
4     head="t1_3" depRel="Sb"/>
5   <t id="t1_3" tag="VDU3S-" lemma="enviar" word="enviará"
6     head="0" depRel="ROOT"/>
7   <!-- ....-->
8 </s>

```

In the above example, to each `t` element we add

- an attribute `head`. The value of this attribute is the `id` of the head token element, or 0. Depending on the output of the dependency parser, we may have more than one 0-valued tokens.
- an attribute `depRel`. The value of this attribute is the dependency relation to the head token element. Depending on the output of the dependency parser, the `depRel` of tokens with 0-valued `head` attributes may be meaningful (i.e. `Pred`) or simply `ROOT`.

In the above example, the dependency tree is headed by the verb `enviará`. The second token of the sentence (UE) is a dependent node of `enviará` (`t1_3`) with a subject dependency relation (`Sb`).

6.1.4 Alignment

In this section, we propose an encoding for storing alignments between files annotated up to the level of syntax in two or more languages. The structure of this alignment file is based on the schema for alignment⁵² described in the latest (1.0.4) release of XCES.

```

1 <?xml version="1.0"?>
2 <cesAlign version="1.0" xmlns="http://www.xces.org/schema/2003">
3   <cesHeader version="1.0">
4     <profileDesc>
5       <translations>
6         <translation trans.loc="news_20100514_haiti_en.tag.xml"
7           wsd="UTF-8" n="1"/>
8         <translation trans.loc="news_20100514_haiti_es.tag.xml"
9           wsd="UTF-8" n="2"/>
10      </translations>
11    </profileDesc>
12  </cesHeader>
13  <linkList>
14    <linkGrp domains="p1 p1" targType="s">
15      <link>

```

⁵¹ <http://nextens.uvt.nl/depparse-wiki/DataFormat>

⁵² <http://www.xces.org/schema/#align>

```

16     <align xlink:href="#s1"/>
17     <align xlink:href="#s1"/>
18     </link>
19 </linkGrp>
20 <linkGrp domains="p2 p2" targType="s">
21     <link>
22         <align xlink:href="#xpointer(id('s3')/range-to(id('s4')))" />
23         <align xlink:href="#s2"/>
24     </link>
25 </linkGrp>
26 </linkList>
27 </cesAlign>

```

As specified in XCES, this document contains a `<cesHeader>` element, followed by a `<linkList>` element. The `<cesHeader>` element can be stored in file or as a separate file. In both cases it may contain metadata information for the alignment process. It also contains links to the path where the aligned (linguistically annotated in the Panacea context) documents are stored.

The `<linkList>` element contains one or more `<linkGrp>` elements considered to be a group. Groups of links apply to data within a particular text division, paragraph, etc. In the example above, we indicate this by creating `<linkGrp>` elements for each paragraph in the annotated files and storing the `ids` of paragraphs in the `domains` attribute of each `<linkGrp>`. The `targType` attribute of each `<linkGrp>` is used to indicate the type of links to be stored inside this element. In the case of the first two `<linkGrp>` elements, the links refer to sentences.

The `<link>` elements in the `<linkGrp>` elements contain the actual links. According to the XCES documentation “the order of the `<align>` elements within a `<link>` element is significant. Unless otherwise specified the order is assumed to match the ordering of `<translation>` elements in the header. If a different ordering is required the attribute `n` in the `<translation>` element and the attribute `n` in the `<align>` element can be used to explicitly link an `<align>` element with a specific translation.”

The XLink locators in the `<align>` elements identify the aligned elements from the annotated files. As again specified in the XCES documentation “many-to-one alignments and many-to-many alignments can be represented by providing a range for the XPointer expression.” This is the case in the alignment between sentences 2 and 3 in the english, and sentence 2 in the spanish document (cf. input in section 6.1.2 above). Notice that similar N-to-N alignments could also be produced for tokens, assuming that tokens are represented as XML elements, and not as tab-separated lines enclosed in an `<s>` element.

In a similar fashion, to represent alignments between chunks or phrases the `xlink` attributes in alignment files may refer to non-terminals from constituency parsers or chunkers (`targType="nt"`), and/or subtrees (`targType="st"`) from dependency trees as in the following artificial example:

```

1     <linkGrp domains="p1 p1" targType="nt">
2         <link>
3             <align xlink:href="#s1_504"/>
4             <align xlink:href="#s1_506"/>
5         </link>
6     </linkGrp>
7     <linkGrp domains="p1 p1" targType="st">
8         <link>
9             <align xlink:href="#t1"/>
10            <align xlink:href="#t3"/>
11        </link>

```

```

12 </linkGrp>
13 </linkList>
14 </cesAlign>

```

In all the alignment examples, we assume an XML tool that parses the alignment files, visits relevant files and extracts necessary pieces of information (like, for example, all subtrees headed by the #t1 and #t3 tokens above).

6.1.5 Revision and distribution metadata

Revisions to the files above can be documented via multiple `<change>` elements in a `<revisionDesc>` child element of the `<cesHeader>`.

```

1 <?xml version="1.0"?>
2 <cesDoc id="news_20100514_haiti_es" version="0.4"
  xmlns="http://www.xces.org/schema/2003">
3   <cesHeader version="0.4">
4     <fileDesc>
5       <!-- As above -->
6     </fileDesc>
7     <profileDesc>
8       <!-- As above -->
9     </profileDesc>
10    <revisionDesc>
11      <!--Summarizes the revision history for a file. -->
12      <change>
13        <changeDate>2012-05-20</changeDate>
14        <respName>Panacea Partner</respName>
15        <item>Corrected an error in the POS tags.</item>
16      </change>
17    </revisionDesc>
18  </cesHeader>
19 </cesDoc>

```

Once the Panacea corpora are ready for distribution, information on rights and availability can be documented via a `<publicationStmt>` child element of the `<cesHeader>`.

```

1 <?xml version="1.0"?>
2 <cesDoc id="news_20100514_haiti_es" version="0.4"
  xmlns="http://www.xces.org/schema/2003">
3   <cesHeader version="0.4">
4     <fileDesc>
5       <!-- . . . -->
6     <publicationStmt>
7       <distributor>Panacea project</distributor>
8       <eAddress>http://www.panacea-lr.eu</eAddress>
9       <availability>Free for research purposes or ...</availability>
10      <pubDate>2013-05-20</pubDate>
11    </publicationStmt>
12    <!-- . . . -->
13  </fileDesc>
14  <!-- Rest as above -->
15 </cesHeader>
16 </cesDoc>

```

6.2 Common interfaces design

The following proposal creates a division between the parameters that are used by most implementations of a concrete operation (*Main parameters*) and those parameters which are only used by a specific implementation (*Optional parameters*).

We start defining a *POSTagger* operation with the corresponding input/output messages. Assuming we take the wrapped document style, the *POSTaggerRequest* message has only one part. All implementation parameters will be represented in that part by means of the associated type *POSTaggerParams*:

Figure 7: POSTagger operation & message in WSDL

```
<operation name="POSTagger">
  <input message="POSTaggerRequest"/>
  <output message=" POSTaggerResponse "/>
</operation>
<message name=" POSTaggerRequest ">
  <part name=" POSTaggerParams"
    element="types:POSTaggerParams"/>
</message>
```

In Figure 8 we give a sketch version of the type declarations from Figure 7. The element *POSTaggerParams* is defined as belonging to the *ParamsType*. *ParamsType* is a complex type consisting of two kinds of elements: the *mainParams* and *optParams*. The *mainParams* refer to objects that typically move around in NLP services either as inputs or as outputs. The *optParams* refer to what we call ‘application parameters’, which are typically used as configuration parameters. *mainParameters* are expected to be common enough so as to be typed using some general type system. *optParams* are optional (they are assigned some default value) and may lack a general type.

Figure 8: preliminary type descriptions

```
<xs:element name=" POSTaggerParams " type="ParamsType"/>
<xs:complexType name="ParamsType">
  <xs:sequence>
    <xs:element ref="mainParams" />
    <xs:element ref="optParams" />
  </xs:sequence>
</xs:complexType>
```

In our PoS tagger example, *mainParams* include text and language. Both types are expected to be general and therefore collected in the general model. Language will obviously be declared in terms of ISOcat. Text type is a bit more complex as here we can further specify mimetypes, encoding formats etc. Functional parameters are collapsed into the *optParams*.

The corresponding XML payload for the *POSTaggerParams* type in Figure 8 goes as follows:

Figure 9: XML payload for PoS tagger

```
< POSTaggerParams>
  < mainParams>
    <language>some language</language>
    <text>some input text to be tagged</text>
  </mainParams>
  <optParams>optional params</optParams>
< /POSTaggerParams>
```

Defining a CI for a tool following this proposal means defining a Type for *mainParams* and a type for the *response*. We have defined CI for a couple of tools: *POSTagger* and *Tokenizer*:

- The *POSTagger* CI is described in *Figure 15: POSTaggerParams* and *Figure 16: POSTaggerResponse* (section 9.2.3 Common Interfaces proposal).

MainParams:

- language (language code ISO)

- text (text is a complex type that can be a string or a URL)

Response:

- text (text is a complex type that can be a string or a URL)

- The *Tokenizer* CI is described in *Figure 17: TokenizerParams* and *Figure 18: TokenizerResponse* (section 9.2.3 Common Interfaces proposal).

MainParams:

- language (language code ISO)
- text (text is a complex type that can be a string or a URL)

Response:

- text (text is a complex type that can be a string or a URL)

Following this analysis and with all data gathered from all partners a first proposal of design for the CI has been developed. For every operation or functionality a CI has been designed and can be found in the appendix 9.2.7 Common interfaces design.

6.3 PANACEA Platform technologies

Different technologies have been surveyed in this deliverable regarding some specific functionalities or components necessary for the PANACEA platform. Some of these technologies are more compatible than others or are designed to work together.

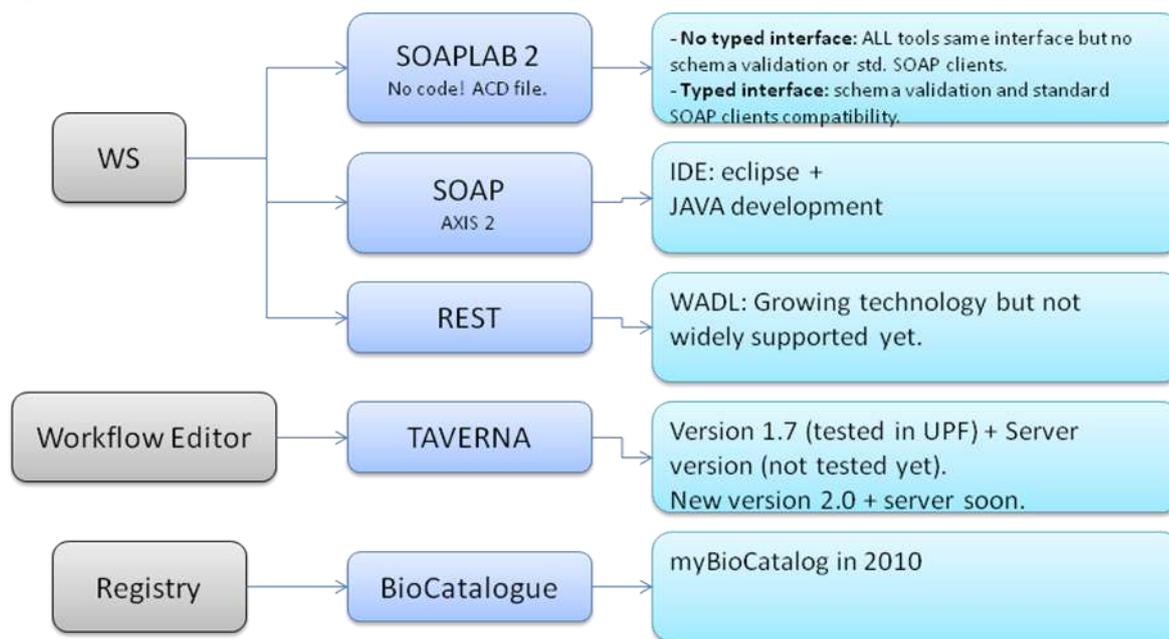
The following section aims to group these technologies creating a few technological options to deploy the platform. After choosing one of these options future researches to be done on those technologies are described and some alternatives are proposed.

6.3.1 Options

6.3.1.1 Option 1: MyGrid environment

The first option is based on using some of myGrid tools and is described in Figure 10.

Figure 10: myGrid environment



In these options web services are developed using Soaplab2 or SOAP(Axis). It is possible that in the near future REST can be used too. By using this option the service provider is given two options to develop its web services: using Axis technology based on SOAP or Soaplab2.

Axis gives the service provider full control of the development since all the needed code has to be done from scratch. Service provider needs programming skills and needs to solve all web service topics himself, for example, temporary files, timeouts, etc.

Soaplab2 service provider needs no programming skills since all code is already developed. The service provider only needs to describe its web service and tool using Soaplab metadata and most web service issues are solved. For example, timeouts need only to be configured. On the other hand, the service provider has less configuration capabilities than developing the code himself.

We can say Soaplab is a low cost and fast solution for service providers with standard requirements and few resources or programming skills. For those service providers who have requirements not fulfilled by Soaplab the solution is Axis.

The myGrid workflow editor is Taverna. Latest release is version 2 which is compatible with Axis web services and Soaplab2 web services using the typed interface. The plug-in for Soaplab2 is still only available for Taverna 1.7 but the new plug-in is supposed to be finished in a few months.

Although it's not technologically necessary to use the same registry being used in myGrid, using BioCatalogue has some advantages: The web 2.0 interfaces and the usability are carefully designed; the integration with Taverna is guaranteed for Taverna 2.0 thanks to a plug-in. Taverna users can access Biocatalogue within the Taverna workbench.

To summarize it can be said that the myGrid approach is a low cost and user friendly solution with a small learning curve for both, developers and users. Its tools have carefully designed interfaces for a nice user experience and are based on robust software but with a few functional limitations compared to other grid solutions like huge data handling and security. However, myGrid developers are adding new features and improvements often thus massive data and security topics can be addressed.

6.3.1.2 Option 2: Large scale Grids

Different grid architectures have been surveyed: Globus, EGEE, NorduGrid, etc. Some of these solutions are modular and have very different levels of compatibility between them. They can be viewed as individual options for PANACEA.

Globus and EGEE are very similar solutions based and designed for large amounts of data and machine resources. They have some operating system compatibility limitations and have a high cost to be installed and to be maintained. Usually, these kinds of architecture require specific resource and people to be assigned to each module.

On the other hand, NorduGrid consortium claims to have developed highly compatible middleware software and a low cost installation and maintenance grid architecture compared to the rest of grids. From this point of view, NorduGrid could be an interesting solution for PANACEA using a whole grid middleware to deploy the platform.

6.3.1.3 Option 3: Fallback position: UIMA / GATE

Frameworks can represent the fallback position for PANACEA in case the other technological options fail. Frameworks have been used for years despite their limited functionalities compared to what other studied technologies can provide. However, newer versions of these frameworks are adding new and interesting capabilities. It must be said that this frameworks are, in some cases, specially designed for NLP work giving them some really suitable and well designed features.

6.3.2 Primary option, future research and alternatives

To begin PANACEA development one technological option must be chosen. There are several considerations that must be taken into account: the functionalities or features provided by each solution, the usability and learning curve for developers and users, the developers and promoters future plans, the documentation, the resources for PANACEA development, etc.

Option 1 (myGrid environment) offers a mature technology that has proven to be successful for many different research fields and that has a lot of support and future plans. All the tools have good documentation, nice graphical interfaces and small learning curve.

Massive data handling has recently been improved for some of the tools and further work is expected. There is a similar situation for security topics: no security features were developed for these tools but some new features are being added and developed for the new releases.

There are several software updates per year and some tools can be integrated and used altogether.

Table 25: Option 1. Mygrid environment. Pros and Cons.

Pros	Cons
------	------

<ul style="list-style-type: none"> - Mature technology (lots of success histories). - Most tools have support. - Tools have nice GUI. - Small learning curve. - Tools are free and open source. - Massive data (further analysis). - Nice semantic model (ontology). - WS oriented. 	<ul style="list-style-type: none"> - No security tools yet.
---	--

Option 2 (Large scale Grids) is a very robust technological approach that can fulfil many of PANACEA requirements but with a high cost in development and maintenance. Tool interfaces are not as user friendly as in option 1 and the learning curve is much larger.

Table 26: Option 2. Large scale Grids. Pros and Cons.

Pros	Cons
<ul style="list-style-type: none"> - Mature technology (lots of success histories). - All tools have support. - Tools are free and open source. - Massive data. - Security with proxy certificates. - WS oriented. 	<ul style="list-style-type: none"> - Most tools have only command line interfaces. - Very complex. - Linux only. - Large learning curve. - No workflow editor (has a job monitor).

In the technical meeting in Athens (April 2010) it was agreed that option 1 (myGrid environment) would be the PANACEA primary option for the platform development. The small learning curve and low cost development compared to the **option 2** were key aspects for the decision.

However, it was agreed to keep on surveying the other options in case there was a problem or in case some tools from other technologies could be used together with the ones in **option 1**. For example, some security tools or data transfer tools.

From the **option 2** survey it can be concluded that grid architectures like Globus or EGEE have a too high cost and compatibility issues (scientific linux only) to be used in PANACEA. On the other hand, NorduGrid could be an interesting option for surveying the grid features and tools to be used in combination with **option 1** or to be used in case **option 1** has a critical problem.

Keeping in mind that some partners have UIMA know-how the PANACEA fallback strategy would be to use UIMA framework to wrap the tools and execute them remotely.

7 Workplan

7.1 To-do list

7.1.1 Travelling Object

TO-GR-01: Define/choose simple TO

- Design of the first version of the travelling object (simple TO).
- Inline annotation.

TO-GR-02: Conversion tools (simple TO)

- Develop scripts or tools to convert in-house formats to TO format (simple TO).

TO-GR-03: Define/choose complex TO

- Design of a second version of the travelling object (complex TO).
- Stand-off annotation?
- New features and improvements.

TO-GR-04: Conversion tools (complex TO)

- Develop scripts or tools to convert in-house format to complex format.

7.1.2 Web Services

7.1.2.1 General

WS-GR-01: Temporary files management.

- Detailed design and test.

WS-GR-02: Provenance.

- Detailed design and test.

WS-GR-03: Security.

- Detailed design and test.

7.1.2.2 Common Interfaces

WS-CI-01: Define operations Common Interface.

- Define input / output parameters commonly used by every kind of tool.
- Improve or change the Common interface if necessary in every version of the platform.

WS-CI-02: Controlled Vocabularies.

- Establish a first version of the closed vocabularies suitable for PANACEA WS, to define operations, inputs, outputs, parameters, etc.
- Standardization: make PANACEA vocabularies ISOCAT compliant.
- Maintenance: make changes and improvements in every following version of the platform.

7.1.2.3 Soaplab

WS-SL-01: Soaplab Test massive data:

- Large data support.
- Multiple files support.
- Multiple files stand-off.
- Improvements or alternatives.

WS-SL-02: Soaplab Test Common Interface capabilities:

- Verify Typed interface feature.
- Verify that Soaplab typed interface can fulfil PANACEA Common Interface.
- Design and develop any software needed to guarantee interoperability.

WS-SL-03: Soaplab security

- Check if there are plans for security development. Soaplab developers willing to help?
- Develop security features (if needed)

WS-SL-04: Soaplab operating system

- Windows Test:
 - o Test if Soaplab can be installed and WS can be generated (probably not).
 - o Deployment test: Soaplab WS web files (.war) should be deployed without problems in Windows.

WS-SL-05: Soaplab temporary files and provenance

- Verify the temporary files management (test, analysis, improvement)
- provenance management (test, analysis, improvement)

WS-SL-06: Soaplab test and deploy new versions

- Test and deploy new versions of the Soaplab software.
- Deploy new or improved web services using Soaplab with newer versions.

7.1.2.4 AXIS

WS-AX-01: Axis Temporary files management development.

WS-AX-02: Axis Provenance support development.

WS-AX-03: Axis Massive data support development.

WS-AX-04: Axis Multiple files support development.

WS-AX-05: Axis Multiple files stand-off support development.

WS-AX-06: Axis Async. execution management development.

7.1.2.5 Alternatives

WS-AL-01: Survey on REST and others.

WS-AL-02: Security: SAML survey.

WS-AL-03: Security: certificates survey.

7.1.3 Workflow editor and engine

7.1.3.1 Workflow Editor

WF-TV-01: Taverna versions test.

- Test and use of versions 1.7 and 2 of Taverna. Problems and solutions.

WF-TV-02: Taverna Server test.

- Test the server version. Async. remote execution for massive data and long lasting workflows.
- Usability test of server version. Can it be used from a portal? (myExperiment?)

WF-TV-03: Taverna Server integration.

- Remote execution for long lasting workflows and massive data.

WF-TV-04: Taverna Server web integration.

- Usability improvement by web integration / myExperiment.

7.1.4 Registry

7.1.4.1 General

RG-GR-01: Temporary registry development.

- MyBioCatalogue software expected for third quarter 2010 Jul-Sep. Is a temporary registry necessary?
- Develop a temporary registry if needed. A simple web site to list services and collect some metadata.

RG-GR-02: Registry metadata design.

- What metadata for each WS, user, workflow etc. need to be stored?
- Will workflows be posted and shared in the registry or in a specific portal like myExperiment?.

7.1.4.2 *MyBioCatalogue* > *PanaceaCatalogue*

BC-GR-01: MyBioCatalogue deployment test.

- Software expected for third quarter 2010 Jul-Sep.
- Verify that MyBioCatalogue can fulfil PANACEA requirements.

BC-GR-02: MyBioCatalogue modification, improvement and test.

- Make changes to suit Panacea requirements:
 - o Adding extra metadata
 - o Helping user to correctly annotate the services with closed vocabularies

BC-GR-03: PanaceaCatalogue deployment.

7.1.5 Portal

ME-GR-01: Analyze myExperiment or other portal features.

- Share information, workflows, executes workflows online, etc.
- Download and test myExperiment / portal.
- Decide whether or not to use a portal?

ME-GR-02: myExperiment/portal improvement.

- Modification and improvement: make changes to suit Panacea requirements.

ME-GR-03: PanaceaPortal deployment.

7.1.6 Tools

7.1.6.1 *Work Package 4 tools*

TL-W4-01: WP4 CAA prototype integration.

- Develop / deploy tools for the formats conversion between proprietary formats and the TO.
- Deploy the tools.
- Test the tools integration.

TL-W4-02: WP4 CAA integration.

- Develop / deploy tools for the formats conversion between proprietary formats and the TO.
- Deploy the tools.
- Test the tools integration.

TL-W4-03: WP4 PoS modules integration.

- Develop / deploy tools for the formats conversion between proprietary formats and the TO.
- Deploy the tools.

- Test the tools integration.

7.1.6.2 Work Package 5 tools

TL-W5-01: WP5 aligners.

- Develop / deploy tools for the formats conversion between proprietary formats and the TO.
- Deploy the tools.
- Test the tools integration.

TL-W5-02: WP5 Bilingual Dictionary Extractor integration.

- Develop / deploy tools for the formats conversion between proprietary formats and the TO.
- Deploy the tools.
- Test the tools integration.

TL-W5-03: WP5 Transfer Grammar Extractor.

- Develop / deploy tools for the formats conversion between proprietary formats and the TO.
- Deploy the tools.
- Test the tools integration.

7.1.6.3 Work Package 6 tools

TL-W6-01: WP6 Lexical Acquisition components integration.

- Develop / deploy tools for the formats conversion between proprietary formats and the TO.
- Deploy the tools.
- Test the tools integration.

7.1.6.4 Travelling object improvements

TL-TO-01: Format converters to newer TO.

- Develop / deploy /improve tools for the formats conversion between proprietary formats and newer versions of the TO.

7.1.7 Other technologies and alternatives. Fallback strategies.

7.1.7.1 Surveys

AL-SV-01: Grid survey.

- NorduGrid or other grid technology alternative to be analyzed.
- Verify if it could fulfil the PANACEA requirements.

AL-SV-02: UIMA / GATE continuous state of the art analysis.

- UIMA and GATE frameworks can be an alternative for PANACEA platform.

7.1.7.2 Tests

AL-TS-01: Grid test.

- Is NorduGrid (or other grid) a feasible option?
- Test deployment. Verify that it can fulfil all the requirements.

AL-TS-02: Test UIMA / GATE.

- Verify if UIMA and / or GATE could fulfil the PANACEA requirements.
- Choose one option

7.1.7.3 Workplan changes

AL-WP-01: Change workplan.

- Change tasks according to partners' decisions.
- Adapt / change workplan according to the chosen technologies.

7.2 Workplan table

- 'X' means a task must be executed / developed.
- '◊' means a task has been done in a previous version and its results should still be operative.
- '*' means a task must be executed if necessary.

Table 27

Category	Task	1 st Ver- sion (t14)	2 nd Ver- sion (t22)	3 rd Ver- sion (t30)
Travelling Object	TO-GR-01: Define/choose simple TO	X	◊	
	TO-GR-02: Conversion tools (simple TO)	X	◊	
	TO-GR-03: Define/choose complex TO	X	X	◊
	TO-GR-04: Conversion tools (complex TO)		X	◊
Web Services	WS-GR-01: Temporary files management	X	◊	◊
	WS-GR-02: Provenance	X	◊	◊
	WS-GR-03: Security			X
Common Inter- faces	WS-CI-01: Define operations Common Interface	X	◊	◊
	WS-CI-02: Controlled Vocabularies		X	◊
Soaplab	WS-SL-01: Soaplab Test massive data	X		
	WS-SL-02: Soaplab Test Common Interface capabilities	X		
	WS-SL-03: Soaplab security			X

	WS-SL-04: Soaplab operating system	X		
	WS-SL-05: Soaplab temporary files and provenance	X	◇	◇
	WS-SL-06: Soaplab test and deploy new versions	X	◇	◇
Axis	WS-AX-01: Axis Temporary files management development	X	◇	◇
	WS-AX-02: Axis Provenance support development	X	◇	◇
	WS-AX-03: Axis Massive data support development		X	◇
	WS-AX-04: Axis Multiple files support development		X	◇
	WS-AX-05: Axis Multiple files stand-off support development		X	◇
	WS-AX-06: Axis Async. execution management development		X	◇
Web services alternatives	WS-AL-01: Survey on REST and others		X	◇
	WS-AL-02: Security: SAML survey		X	
	WS-AL-03: Security: certificates survey		X	
Workflow editor	WF-TV-01: Taverna versions test	X	◇	◇
	WF-TV-02: Taverna Server test	X	◇	◇
	WF-TV-03: Taverna Server integration		X	◇
	WF-TV-04: Taverna Server web integration			X
Registry	RG-GR-01: Temporary registry development	X		
	RG-GR-02: Registry metadata design	X	◇	◇
Biocatalogue	BC-GR-01: MyBioCatalogue deployment test		X	
	BC-GR-02: MyBioCatalogue modification, improvement and test		X	◇
	BC-GR-03: PanaceaCatalogue deployment			X
Portal	ME-GR-01: Analyze myExperiment or other portal features		X	
	ME-GR-02: myExperiment/portal improvement			X
	ME-GR-03: PanaceaPortal deployment			X
Work package 4 tools	TL-W4-01: WP4 CAA prototype integration	X		
	TL-W4-02: WP4 CAA integration		X	◇

	TL-W4-03: WP4 PoS modules integration			X
Work package 5 tools	TL-W5-01: WP5 aligners	X	◇	◇
	TL-W5-02: WP5 Bilingual Dictionary Extractor integration			X
	TL-W5-03: WP5 Transfer Grammar Extractor			X
Work package 6 tools	TL-W6-01: WP6 Lexical Acquisition components integration			X
Travelling Object improvements	TL-TO-01: Format converters to newer TO		X	◇
Alternatives surveys	AL-SV-01: Grid survey	X	◇	◇
	AL-SV-02: UIMA / GATE continuous state of the art analysis	X	◇	◇
Alternatives tests	AL-TS-01: Grid test		X	
	AL-TS-02: Test UIMA / GATE		*	
Workplan changes	AL-WP-01: Change workplan	*	*	*

7.2.1 Resources

Expressed in person-month.

Table 28

	UPF	ILC	ILSP	LG	DCU	ELDA
TO-GR-01-04	1	1,5	1	1	1	
WS-GR-01-03		3				
WS-CI-01-02	2					
WS-SL-01-03	1,5					
WS-SL-04				1		
WS-SL-05-06		2				
WS-AX-01-06		1,5		6		
WS-AL-01-03		2				
WF-TV-01-03	3					
WF-TV-04						1
RG-GR-01-02	1					1

BC-GR-01-02	2					4
BC-GR-03		1				3
ME-GR-01-03		1				5
TL-W4-01-02			2			
TL-W4-03		1		1		
TL-W5-01-02					3	
TL-W5-03				1		
TL-W6-01	1	1				
TI-TO-01	1	1	1	1	1	
AL-SV-01	1					
AL-SV-02			0,5			
AL-TS-01	1,5					
AL-TS-02			0,5			
TOTAL	15	15	5	11	5	14

8 Bibliography

[ARC Nordugrid] M. Ellert, M. Gronager, A. Konstantinov, B. Konya, J. Lindemann, I. Livenson, J.L. Nielsen, M. Niinimaki, O. Smirnova, A. Waananen, Advanced Resource Connector middleware for lightweight computational Grids, Future Generation Computer Systems, Volume 23, Issue 2, February 2007, Pages 219-240, ISSN 0167-739X, DOI: 10.1016/j.future.2006.05.008.

[Berners-Lee 2005] Berners-Lee, T, et al., "Uniform Resource Identifier (URI): Generic Syntax", IETF RFC 3986, January 2005, <http://tools.ietf.org/rfc/rfc3986.txt>

[Biocatalogue] K. Belhajjame, C. Goble, F. Tanoh, J. Bhagat, K. Wolstencroft, R. Stevens, E. Nzuobontane, H. McWilliam, T. Laurent, and R. Lopez, "BioCatalogue: A Curated Web Service Registry for the Life Science Community," in Microsoft eScience conference, 2008.

[Brown 2004] Brown, A. and Haas, H. (2004). Web Services Glossary. *W3C working group note*. <http://www.w3.org/TR/ws-gloss/>

[CiTER] Citation of Electronic Resources, ISO Draft (2008)

[Feta] P. Lord, P. Alper, C. Wroe, and C. Goble, "Feta: A Light-Weight Architecture for User Oriented Semantic Service Discovery," in *European Semantic Web Conference*, 2005, pp. 17-31.

[gLite] Newhouse, S. 2009. The EGEE Distributed Computing Infrastructure. *Connexions*, September 21, 2009. <http://cnx.org/content/m32047/1.1/>.

[Globus] Globus Toolkit Version 4: Software for Service-Oriented Systems. I. Foster. IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13, 2005.

[Globus] The Anatomy of the Grid: Enabling Scalable Virtual Organizations. I. Foster, C. Kesselman, S. Tuecke. International J. Supercomputer Applications, 15(3), 2001.

[Guenther 2004] Rebecca Guenther (Library of Congress), "PREMIS - Preservation Metadata Implementation Strategies Update 2: Core Elements for Metadata to Support Digital Preservation" RLG DigiNews: December 2004 http://www.rlg.org/en/page.php?Page_ID=20492#article2

[KYOTO] Eneko Agirre, Xabier Artola, Arantza Diaz de Ilarraza, German Rigau, Aitor Soroa, and Wauter Bosma. 2009. KAF: Kyoto Annotation Framework. Technical Report TR 1-2009, Dept. Computer Science and Artificial Intelligence, University of the Basque Country.

[KYOTO] K. Lee, J. Pustejovsky, H. Bunt, B. Boguraev, and N. Ide. Language resource management - Semantic annotation framework (SemAF) - Part 1 :Time and events. International Organization for Standardization, Geneva, Switzerland, 2007. <http://lirics.loria.fr/doc/pub/SemAFCD24617-1Rev12.pdf>.

[KYOTO] Lionel Clément and Eric Villemonte de la Clergerie. Maf: a morphosyntactic annotation framework. In Proceedings of the 2nd Language & Technology Conference, page 90–94, April 2005.

[KYOTO] Nancy Ide and Laurent Romary (2004). International standard for a linguistic annotation framework. Natural Language Engineering, 10 , pp 211-225, doi:10.1017/S135132490400350X

[KYOTO] Thierry Declerck. Synaf: Towards a standard for syntactic annotation. In Nicoletta Calzolari, Khalid Choukri, Aldo Gangemi, Bente Maegaard, Joseph Mariani, Jan Odijk, and Daniel Tapias, editors, Proceedings of the Fifth Conference on International Language Resources and Evaluation, pages 229–233. European Language Resources Association (ELRA), May 2006.

[LoonyBin] J. Clark, [A. Lavie](#), "LoonyBin: Keeping Language Technologists Sane through Automated Management of Experimental (Hyper)Workflows", LREC 2010. Malta. [[PDF](#)]

[LoonyBin] J. Clark, J. Weese, [B. Ahn](#), [A. Zollmann](#), [Q. Gao](#), [K. Heafield](#), [A. Lavie](#), "The Machine Translation Toolpack for LoonyBin: Automated Management of Experimental Machine Translation HyperWorkflows", *Prague Bulletin of Mathematical Linguistics (Presented at the Fourth Machine Translation Marathon)* January 2010. Dublin, Ireland [[PDF](#)] [[MT Lunch Slides](#)] [[MT Marathon Slides](#)]

[Mackenzie 2006] MacKenzie C.M., Laskey K., McCabe F., Brown P.F., Metz R., Hamilton B.A. OASIS Reference Model for Service Oriented Architecture 1.0, August 2006, <http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf>

[myExperiment] D. De Roure, C. Goble, and R. Stevens, "The Design and Realisation of the myExperiment Virtual Research Environment for Social Sharing of Workflows," *Future Generation Computer Systems*, vol. 25, pp. 561 - 567, 2008.

[Soaplab] M. Senger, P. Rice and T. Oinn. "Soaplab - a unified Sesame door to analysis tools (2003)" In UK e-Science All Hands Meeting.

[Stanica 2006] Stanica M., Wiberg T., Wierenga K., Winter S., Rauschenbach J., JRA5 Glossary of Terms - Second Edition- update of DJ5.1.1

[Taverna] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. Pocock, P. Li, and T. Oinn, "Taverna: a tool for building and running workflows of services.," *Nucleic Acids Research*, vol. 34, iss. Web Server issue, pp. 729-732, 2006.

[Taverna] T. Oinn, M. Greenwood, M. Addis, N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe, "Taverna: lessons in creating a workflow environment for the life sciences," *Concurrency and Computation: Practice and Experience*, vol. 18, iss. 10, pp. 1067-1100, 2006.

[U-compare] Kano, Yoshinobu, William A. Baumgartner Jr., Luke McCrohon, Sophia Ananiadou, K. Bretonnel Cohen, Lawrence Hunter and Jun'ichi Tsujii U-Compare: share and compare text mining tools with UIMA. *Bioinformatics*. 25(15), pp. 1997-1998, 2009; doi: 10.1093/bioinformatics/btp289

[U-compare] Kano, Yoshinobu, William A. Baumgartner Jr., Luke McCrohon, Sophia Ananiadou, K. Bretonnel Cohen, Lawrence Hunter and Jun'ichi Tsujii U-Compare: share and compare text mining tools with UIMA. *Bioinformatics*. 25(15), pp. 1997-1998, 2009;; doi: 10.1093/bioinformatics/btp289

[Wulong 2001] Wulong T., 2001, http://searchcio.techtarget.com/sDefinition/0,,sid182_gci213384,00.html

9 Appendix

9.1 Appendix A

9.1.1 Current state of the art analysis schema

Analysis of main characteristics

Functionalities: what they do.

Integration: how easy will be to use it in our scenario: versions, operating systems, ...

Maturity: history of the supplier and of the product to see reliability.

Support and plans for the future: is there documentation or information about future evolutions of the product.

Availability: is it free, open source, ... etc.

Table 29

	Functionalities	Cost / learning curve	Integrability	maturity	Support and plans for the future	Availability
Tool A	😊		😊😊	😊	😐	😐
Tool B	😊😊		😊😊	😊😊	😊😊	😊
Technology A	😊😊		😊	😊	😊	😊😊
Framework A	😊		😐	😐	😐	😊😊
Wrapper A	😊😊		😊😊	😊😊	😊😊	😐

9.2 Appendix B

9.2.1 Emboss Groups

Table 30: Emboss groups table

Top Level	Second Level	Description
Acid		ACD file utilities
Alignment	Consensus	Merging sequences to make a consensus
	Differences	Finding differences between sequences
	Dot_plots	Dot plot sequence comparisons
	Global	Global sequence alignment
	Local	Local sequence alignment
	Multiple	Multiple sequence alignment
Assembly	Fragment_assembly	DNA sequence assembly
Display		Publication-quality display
Edit		Sequence editing
Enzyme_Kinetics		Enzyme kinetics calculations
Feature_tables		Manipulation and display of sequence annotation
HMM		Hidden Markov Model analysis
Information		Information and general help for users
Menus		Menu interface(s)
Nucleic	2D_structure	Nucleic acid secondary structure
	Codon_usage	Codon usage analysis
	Composition	Composition of nucleotide sequences
	CpG_islands	CpG island detection and analysis
	Gene_finding	Predictions of genes and other genomic features
	Motifs	Nucleic acid motif searches
	Mutation	Nucleic acid sequence mutation
	Profiles	Nucleic acid profile generation and searching
	Primers	Primer prediction
	Repeats	Nucleic acid repeat detection
	RNA_folding	RNA folding methods and analysis
	Restriction	Restriction enzyme sites in nucleotide sequences
	Transcription	Transcription factors, promoters and terminator prediction
	Translation	Translation of nucleotide sequence to protein sequence
Phylogeny	Consensus	Phylogenetic consensus methods
	Continuous_characters	Phylogenetic continuous character methods
	Discrete_characters	Phylogenetic discrete character methods
	Distance_matrix	Phylogenetic distance matrix methods
	Gene_frequencies	Phylogenetic gene frequency methods
	Molecular_sequence	Phylogenetic tree drawing methods
	Tree_drawing	Phylogenetic molecular sequence methods
	Misc	Phylogenetic other tools
Protein	2D_structure	Protein secondary structure

	3D_structure	Protein tertiary structure
	Composition	Composition of protein sequences
	Motifs	Protein motif searches
	Mutation	Protein sequence mutation
	Profiles	Protein profile generation and searching
Test		Testing tools, not for general use.
Utils	Database_creation	Database installation
	Database_indexing	Database indexing
	Misc	Utility tools

9.2.2 Soaplab test

Figure 11: splitter WS

Simple

cat	<i>Cat example using STDIN</i>
freeling_file	<i>Freeling example using freeling_test.sh</i>
freeling_stdin	<i>Freeling example</i>
helloworld	<i>Classic greeting from the beginning of the UNIX epoch</i>
splitter	<i>Simple sentence splitter example</i>

Report

Inputs

input

```

this is my default input
data example. This is
sentence 2. This is
sentence 3.

```

as URL
 direct data or local file

notags yes no

Figure 12: Freeling WS

freeling_file
Freeling example using freeling_test.sh

Run service

COMPLETED

Result	Size	Type
output	160	text
report	463	text
detailed_status	1	unknown

Remove

Inputs

input

This is a text. This is more text.

as URL

 direct data or local file

Examiner...

language en.cfg

mandatory

optional

<i>noafx</i>	yes	<input type="radio"/>	no	<input checked="" type="radio"/>
<i>nodate</i>	yes	<input type="radio"/>	no	<input checked="" type="radio"/>
<i>nodict</i>	yes	<input type="radio"/>	no	<input checked="" type="radio"/>
<i>noflush</i>	yes	<input type="radio"/>	no	<input checked="" type="radio"/>
<i>noloc</i>	yes	<input type="radio"/>	no	<input checked="" type="radio"/>
<i>nonumb</i>	yes	<input type="radio"/>	no	<input checked="" type="radio"/>
<i>noprob</i>	yes	<input type="radio"/>	no	<input checked="" type="radio"/>
<i>nopunt</i>	yes	<input type="radio"/>	no	<input checked="" type="radio"/>
<i>noquant</i>	yes	<input type="radio"/>	no	<input checked="" type="radio"/>

Report

Summary:

```

Completed: Successfully
Termination status: 0
Started: 2010-mar-18 13:02:52 (CET)
Ended: 2010-mar-18 13:02:53 (CET)
Duration: 0:00:00.857
                    
```

Report:

```

Name: simple.freeling_file
Job ID: [simple.freeling_file]37f89d0f.12771
Program and parameters:
/home/mpoch/scripts/freeling_test.sh
-i
/home/mpoch/apache-tomcat-5.5.28/temp/_R/SA
-f
en.cfg
--- end of parameters
Exit: 0
                    
```

Figure 13: Test workflow

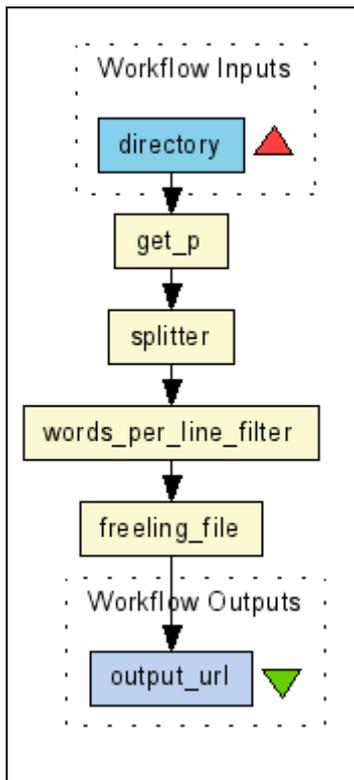
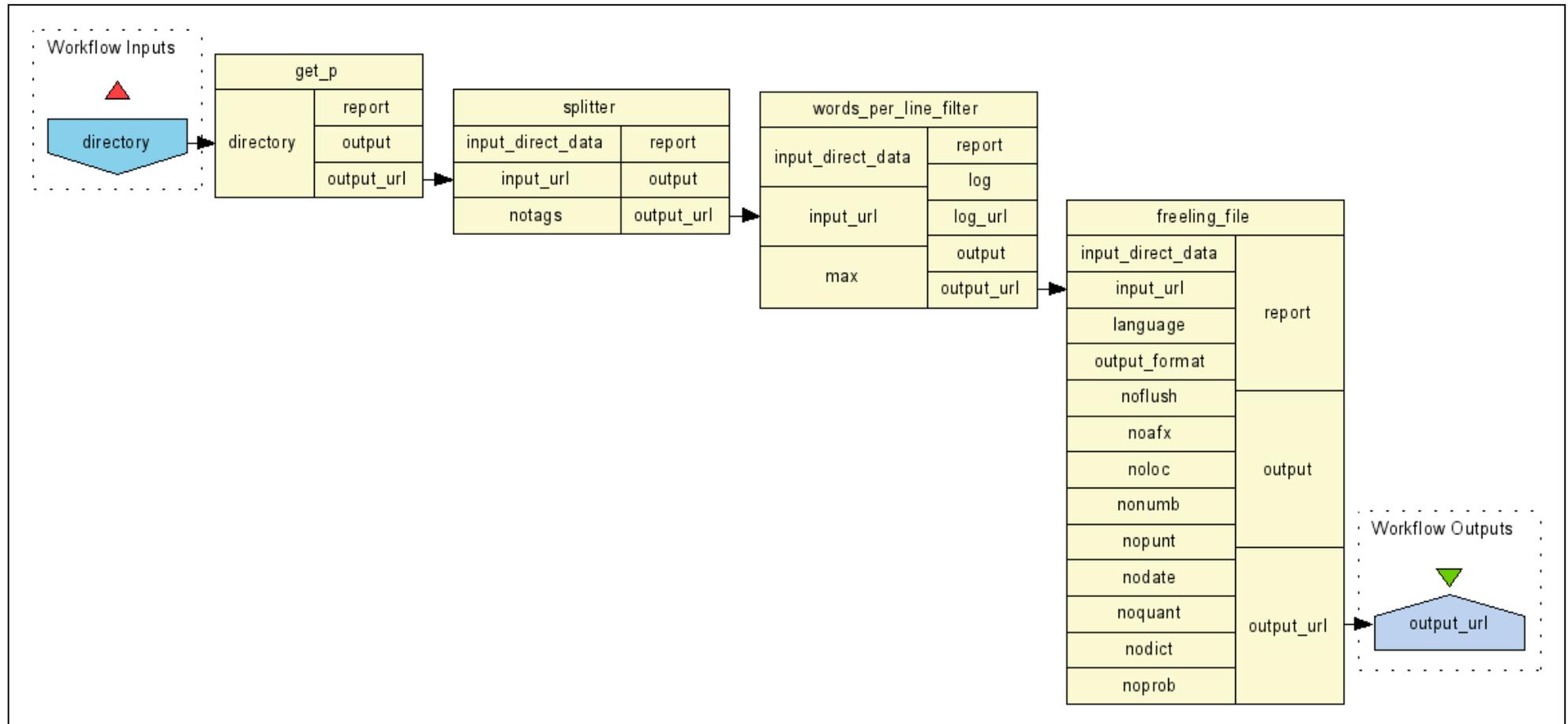


Figure 14: detailed test workflow



9.2.3 Common Interfaces proposal

Figure 15: POSTaggerParams

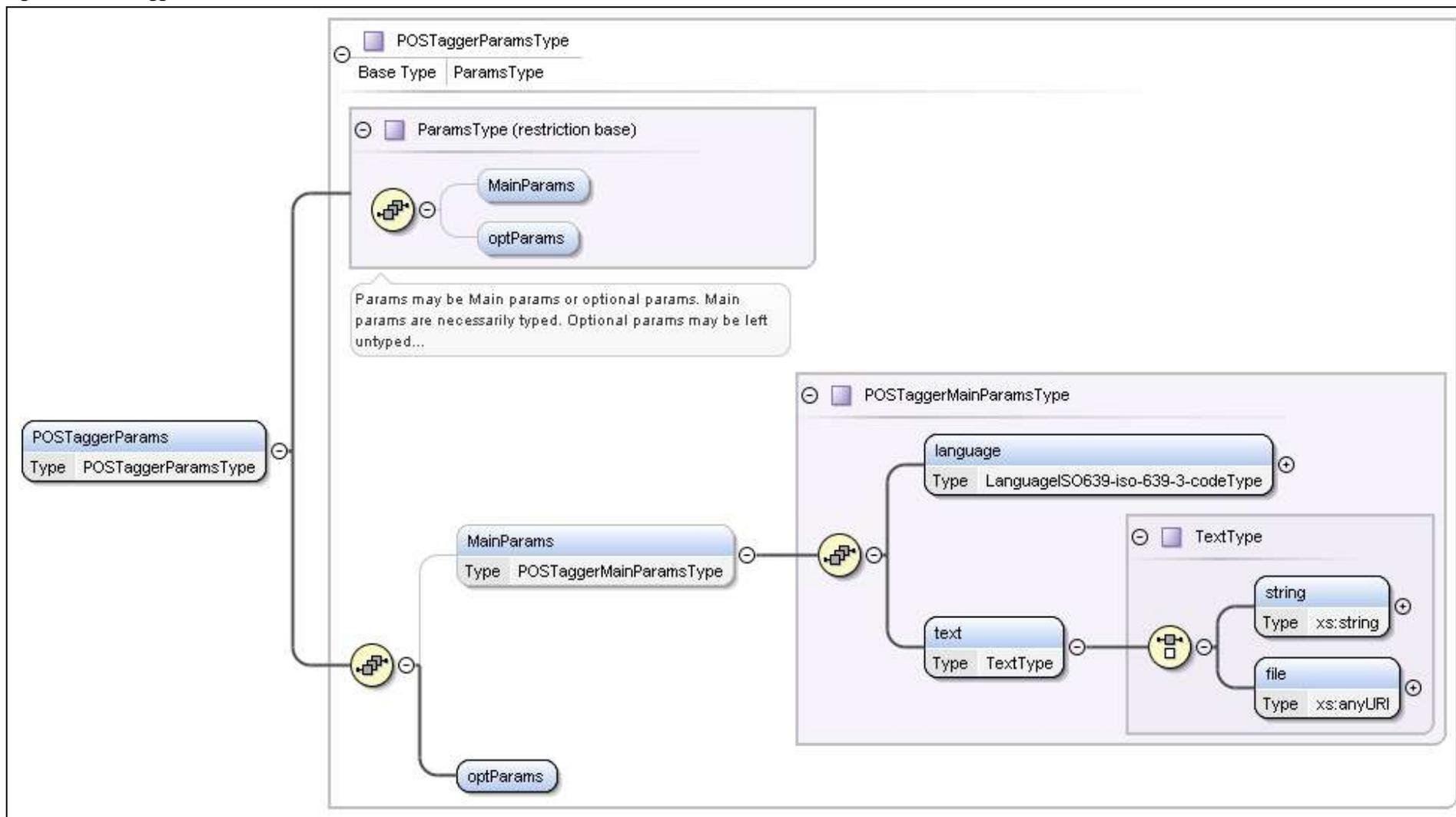


Figure 16: POSTaggerResponse

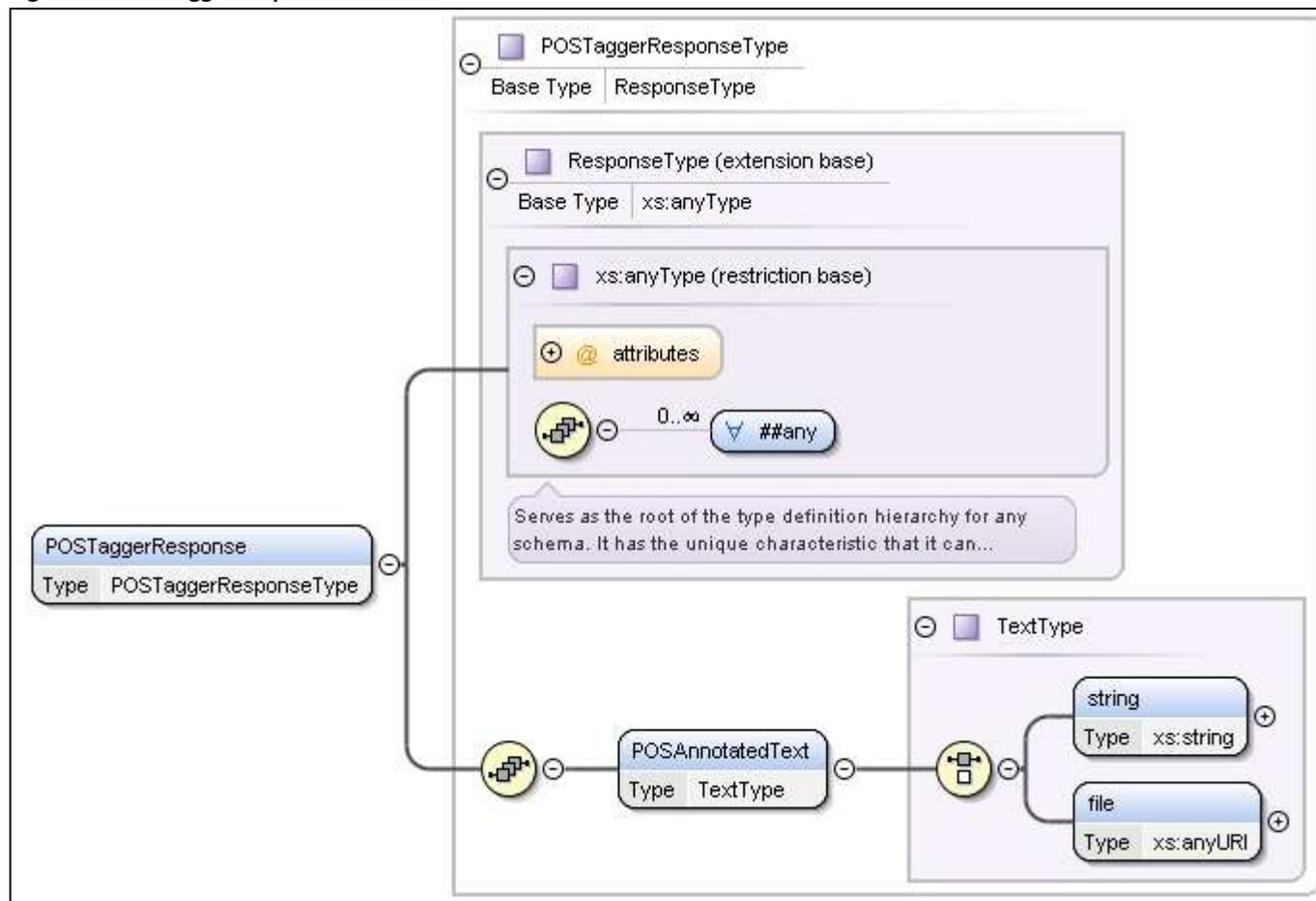


Figure 17: TokenizerParams

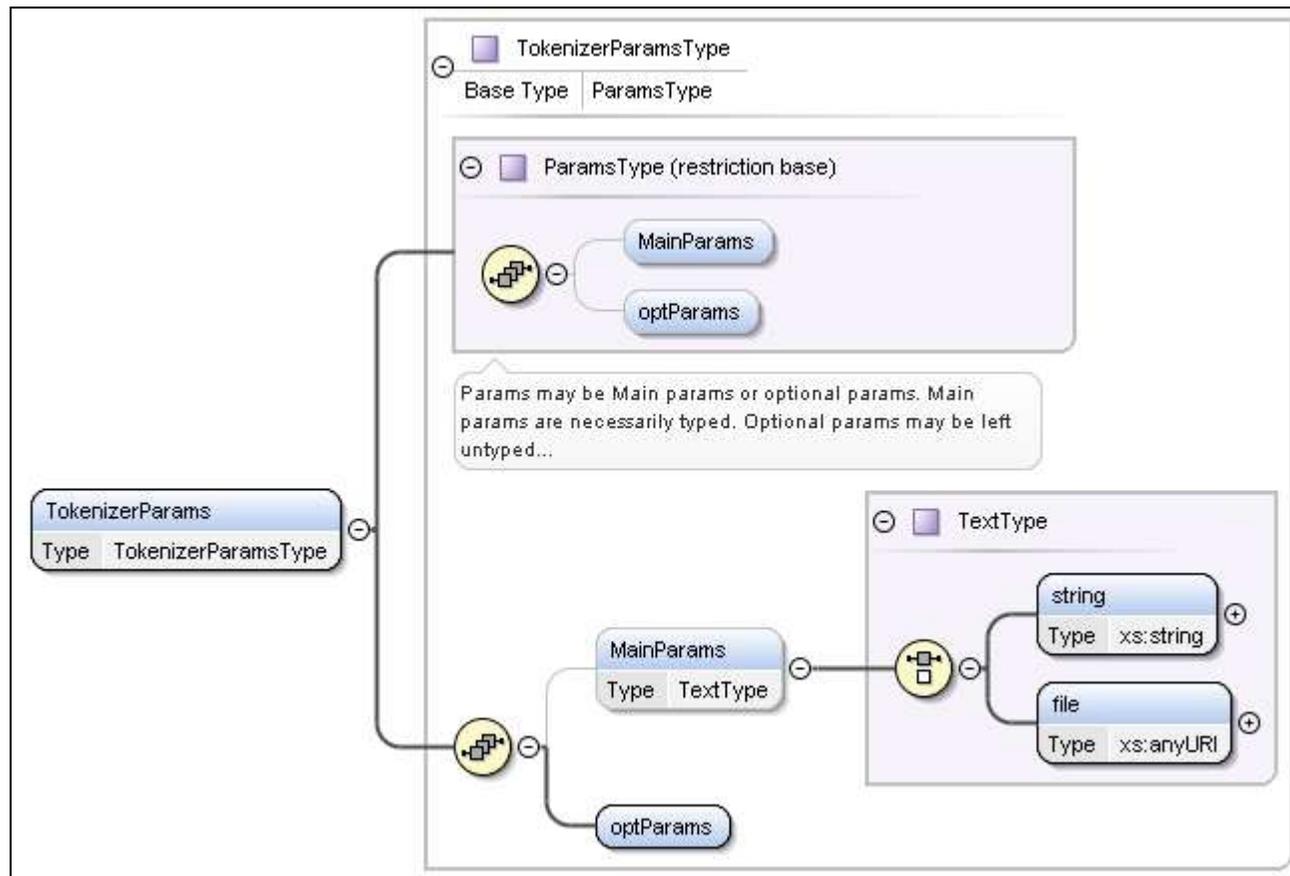
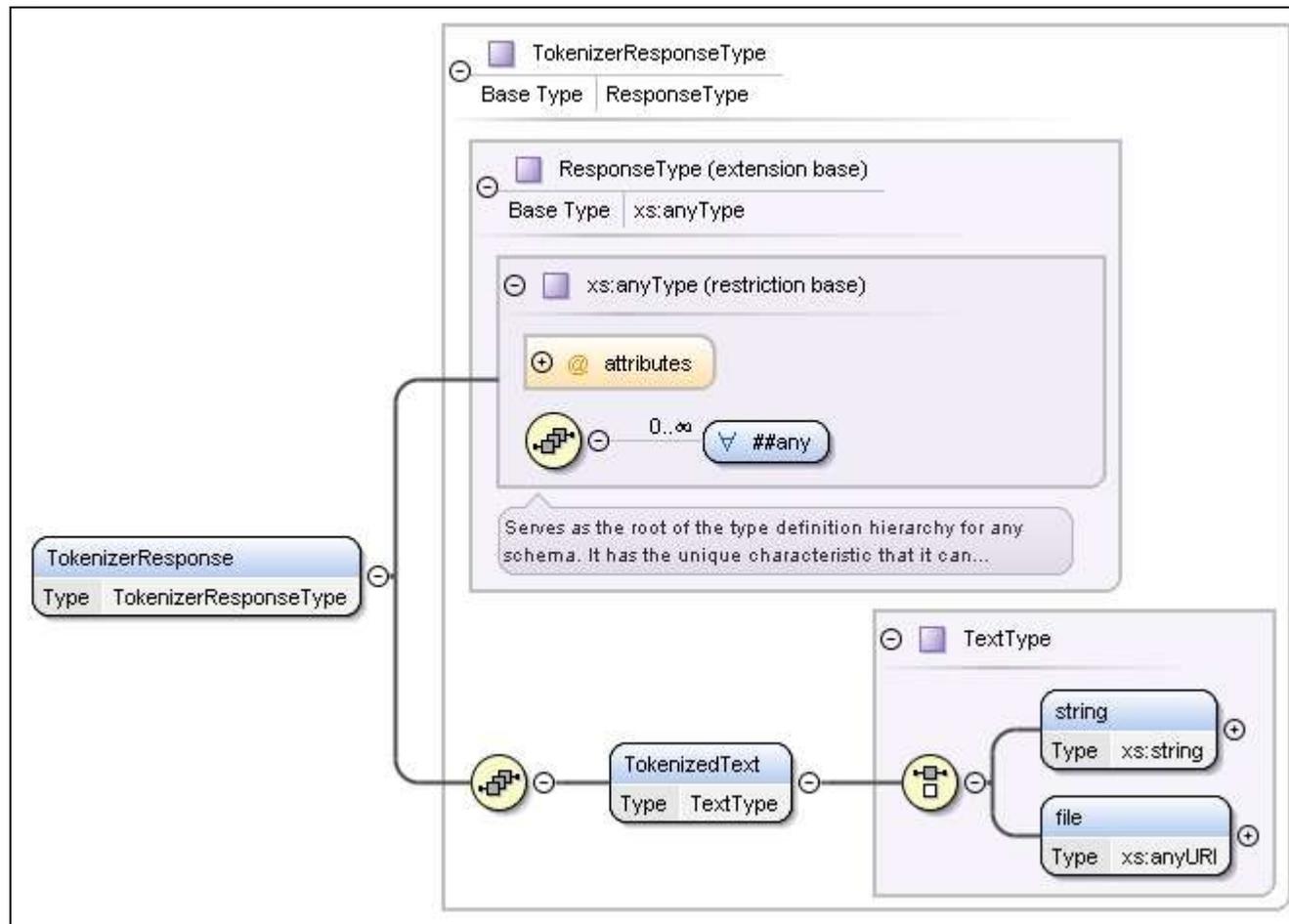


Figure 18: TokenizerResponse

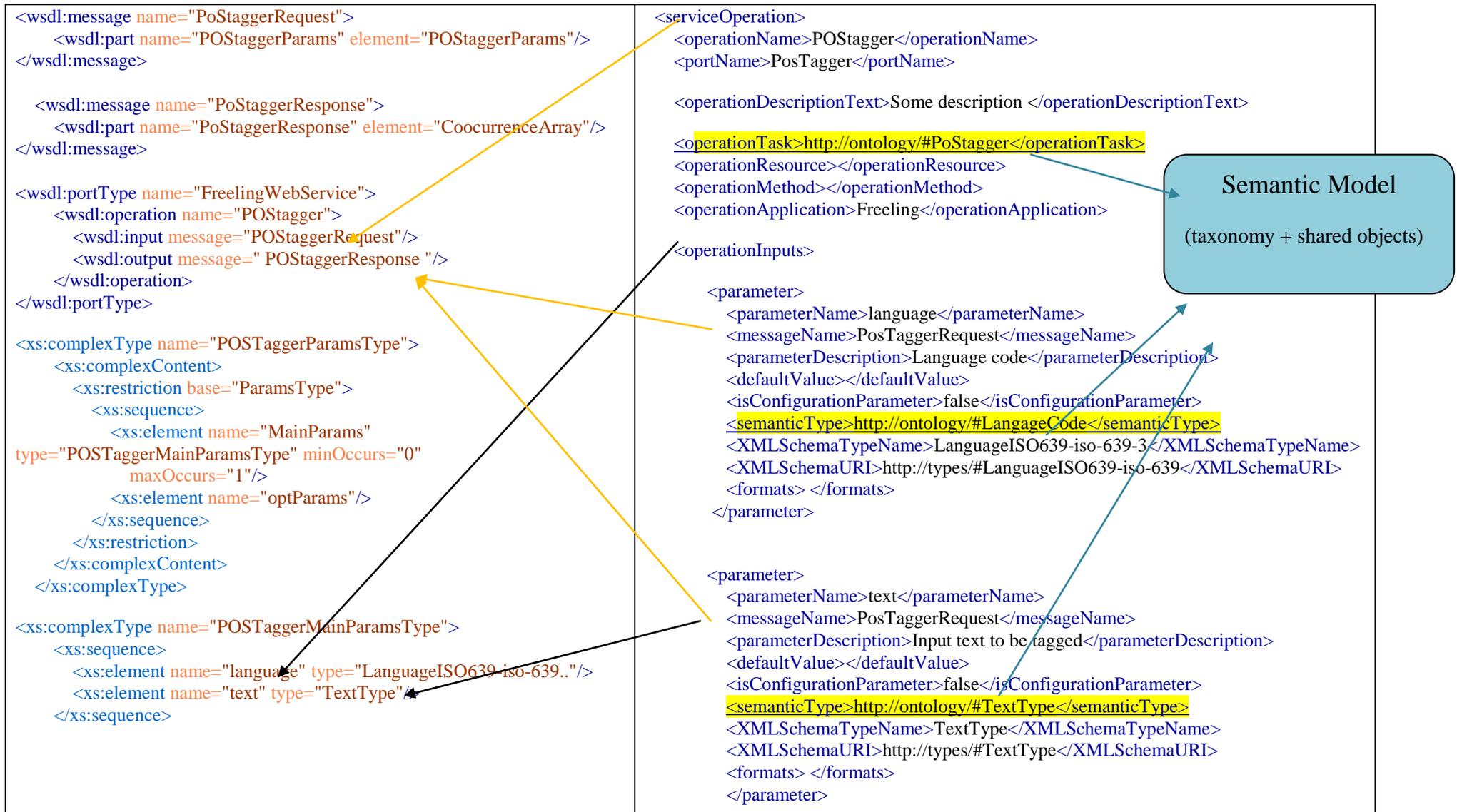




9.2.4 Semantic Service Description (myGrid way)

Figure 19: WSDL / Semantic service description

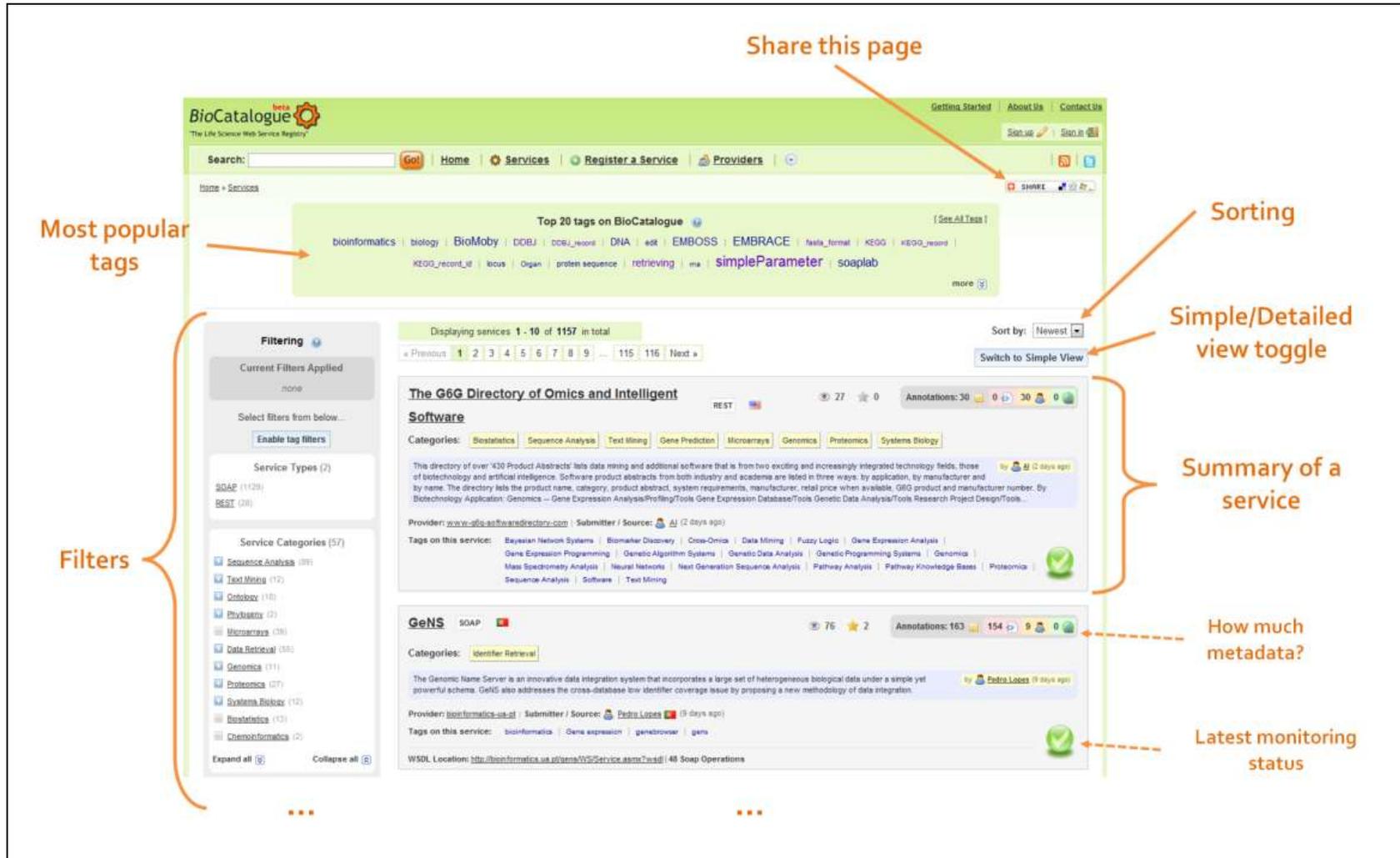
WSDL	Semantic Service Description (“a la MyGrid”)
------	--



<pre></xs:complexType> <xs:complexType name=" LanguageISO639-iso-639-3-codeType "> ... <xs:complexType name="TextType"> <xs:choice> <xs:element name="string" type="xs:string"/> <xs:element name="file" type="xs:anyURI"/> </xs:choice> </xs:complexType></pre>	
--	--

9.2.5 Biocatalogue web user interface

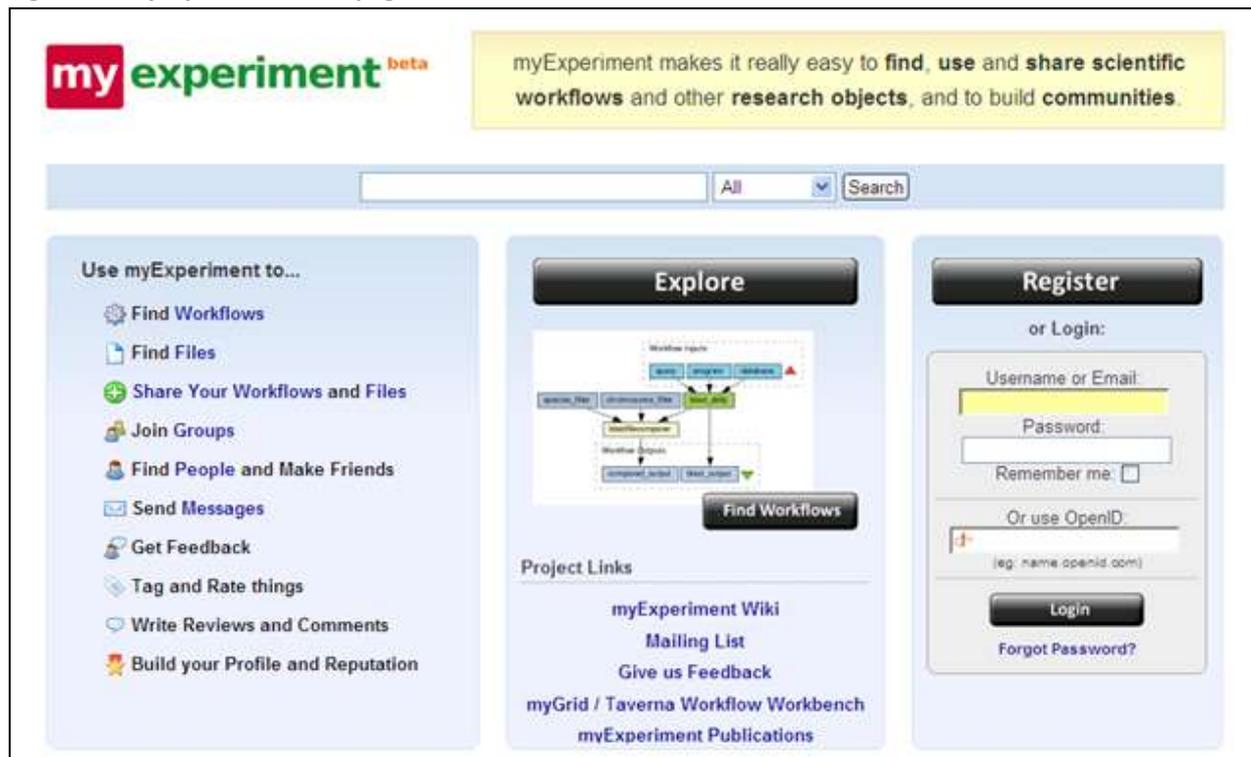
Figure 20: Biocatalogue user interface



The screenshot displays the BioCatalogue web user interface. At the top, there is a navigation bar with a search field, a 'Go!' button, and links for Home, Services, Register a Service, and Providers. A 'Share this page' button is located in the top right corner. Below the navigation bar, a section titled 'Top 20 tags on BioCatalogue' lists various tags such as bioinformatics, biology, BioMoby, and simpleParameter. A 'Sorting' dropdown menu is set to 'Newest', and a 'Switch to Simple View' button is visible. On the left side, a 'Filters' sidebar shows 'Current Filters Applied' as 'none' and lists 'Service Types' (e.g., SOAP, REST) and 'Service Categories' (e.g., Sequence Analysis, Text Mining). Two service cards are shown: 'The G6G Directory of Omics and Intelligent Software' and 'GeNS'. The G6G card includes a 'Summary of a service' section with categories like Bioinformatics and Sequence Analysis, and a 'How much metadata?' section showing 30 annotations. The GeNS card shows 163 annotations and a 'Latest monitoring status' icon. Annotations for each service include counts for stars, annotations, and other metrics.

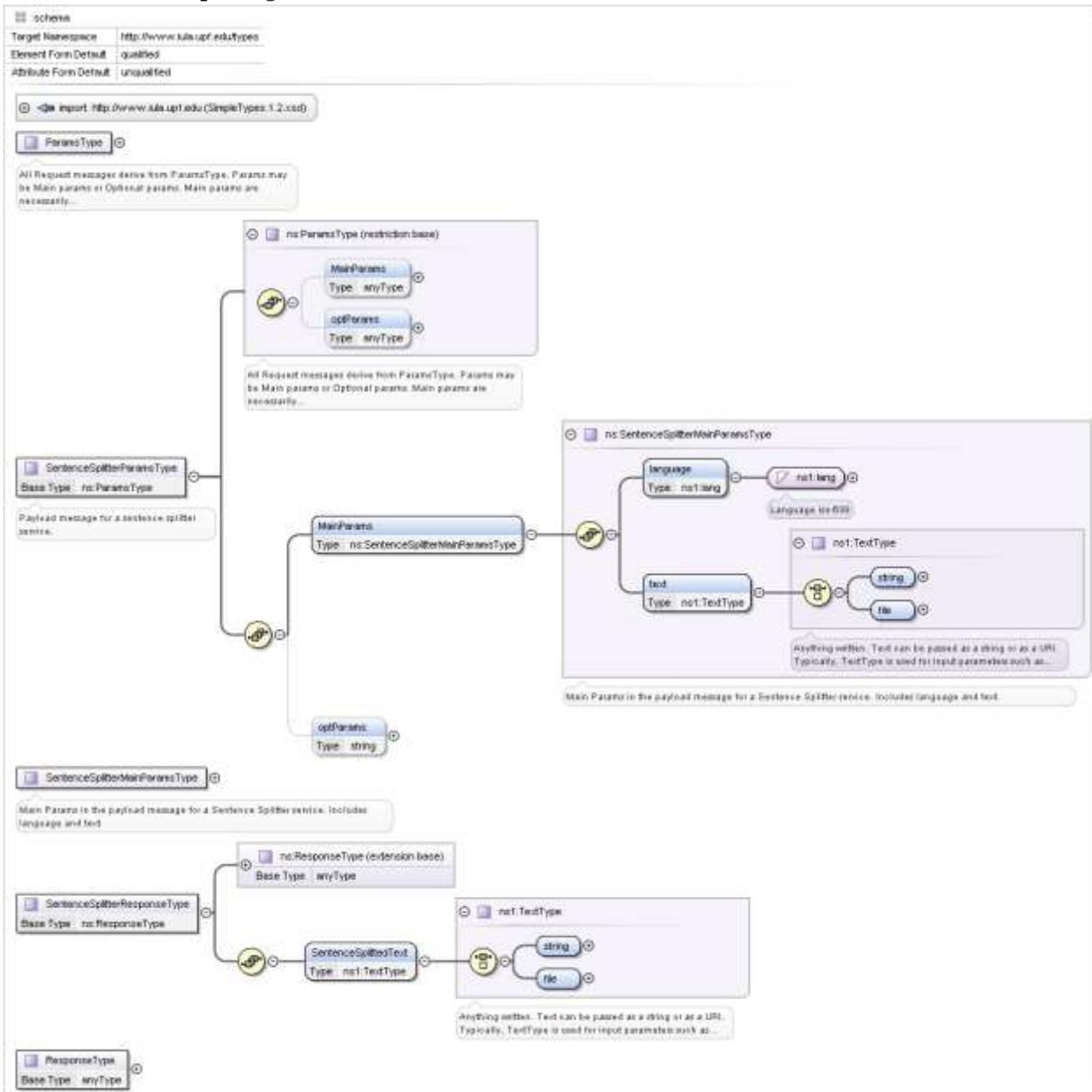
9.2.6 myExperiment

Figure 21: myExperiment main page

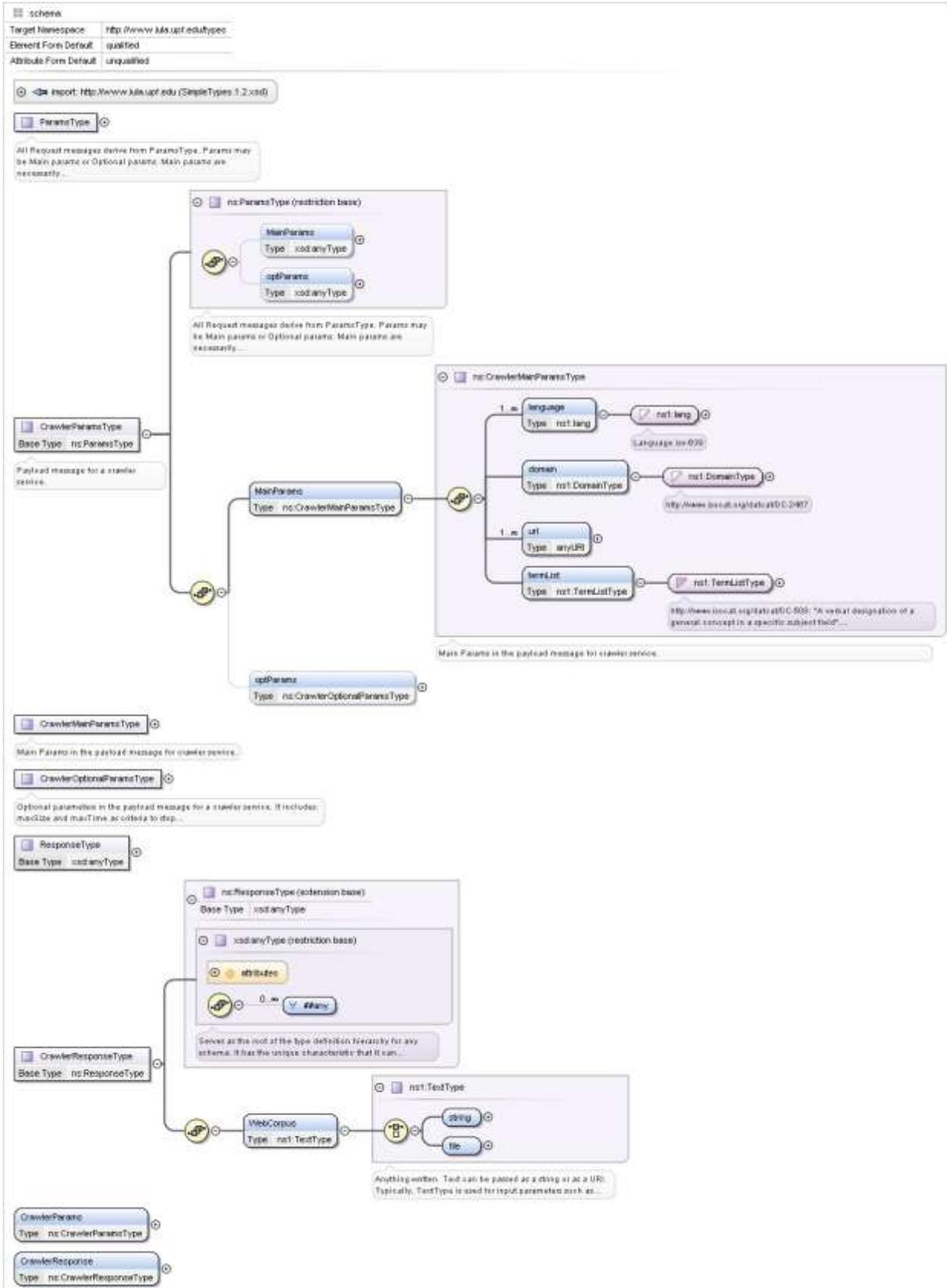
The screenshot shows the myExperiment main page. At the top left is the myExperiment logo with 'beta' in orange. To its right is a yellow banner with the text: 'myExperiment makes it really easy to find, use and share scientific workflows and other research objects, and to build communities.' Below the banner is a search bar with a dropdown menu set to 'All' and a 'Search' button. The page is divided into three main columns. The left column, titled 'Use myExperiment to...', lists several actions: Find Workflows, Find Files, Share Your Workflows and Files, Join Groups, Find People and Make Friends, Send Messages, Get Feedback, Tag and Rate things, Write Reviews and Comments, and Build your Profile and Reputation. The middle column, titled 'Explore', features a diagram of a workflow with nodes like 'Workflow Input', 'Workflow Output', and 'Workflow Steps'. Below the diagram is a 'Find Workflows' button and a 'Project Links' section with links to myExperiment Wiki, Mailing List, Give us Feedback, myGrid / Taverna Workflow Workbench, and myExperiment Publications. The right column, titled 'Register', has a 'Register' button and an 'or Login:' section. The login section includes fields for 'Username or Email' and 'Password', a 'Remember me' checkbox, and an 'Or use OpenID:' section with a field for an OpenID (e.g., name.openid.com) and a 'Login' button. A 'Forgot Password?' link is also present.

9.2.7 Common interfaces design

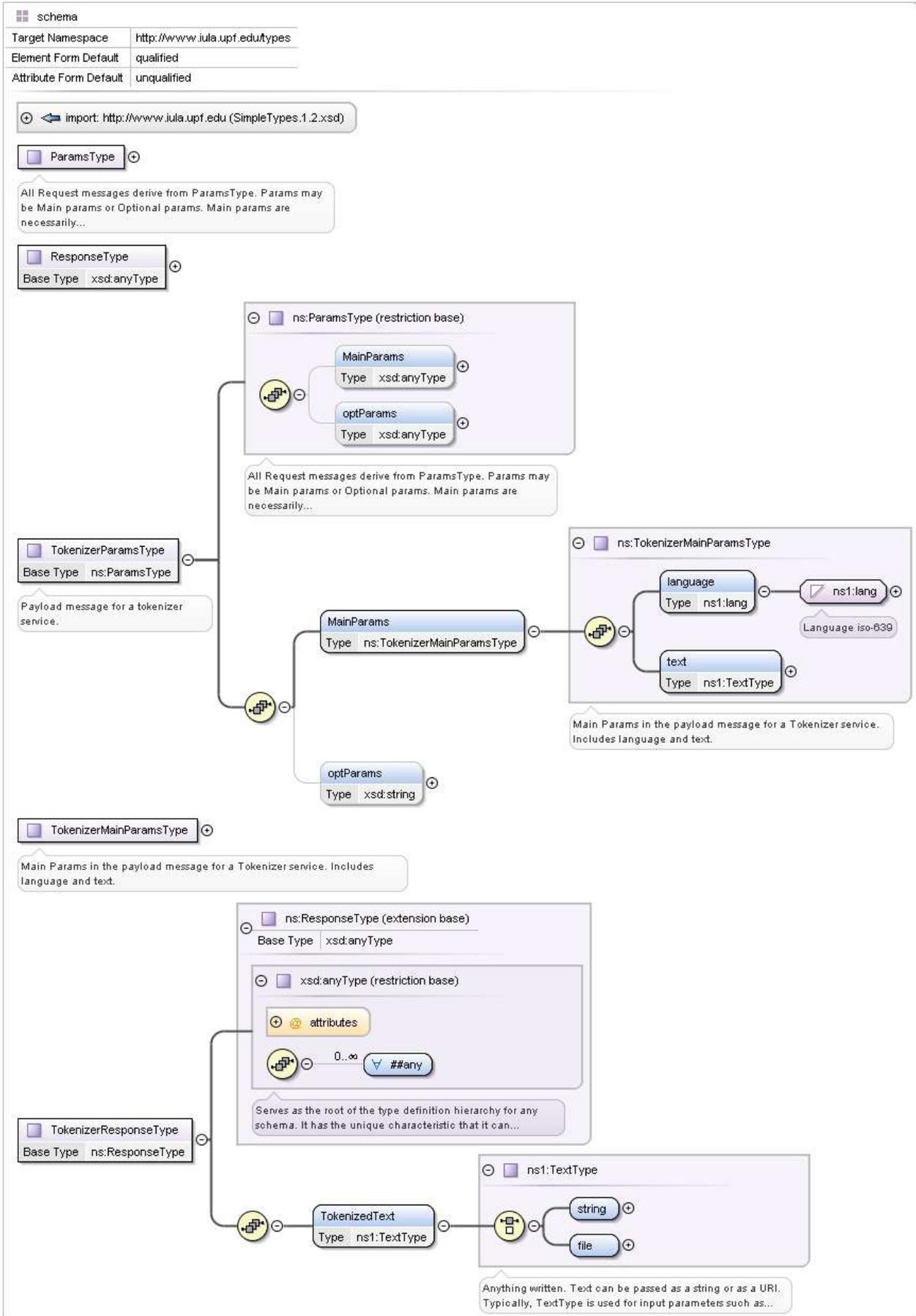
9.2.7.1 Sentence Splitting



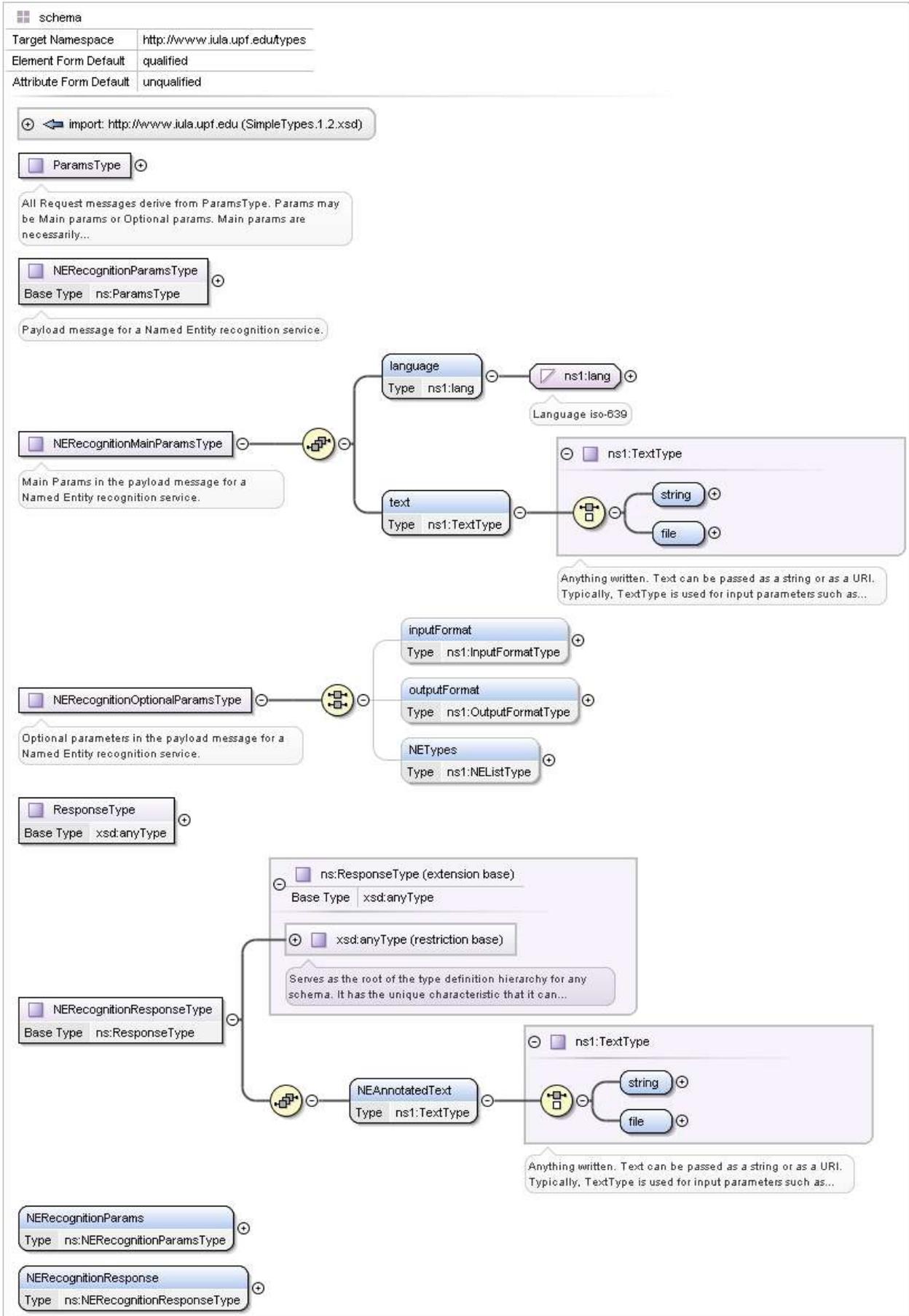
9.2.7.2 Crawling



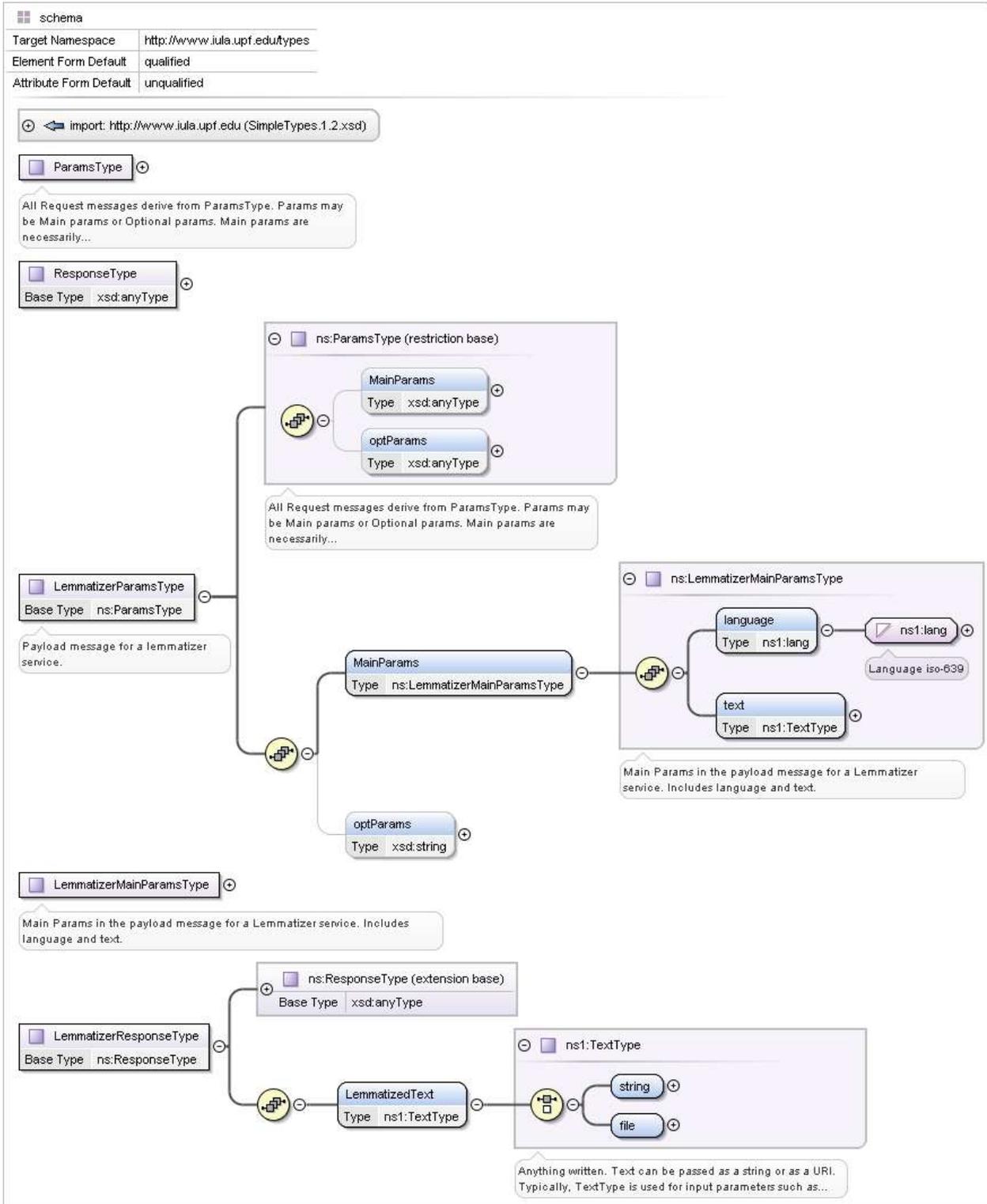
9.2.7.3 Tokenization



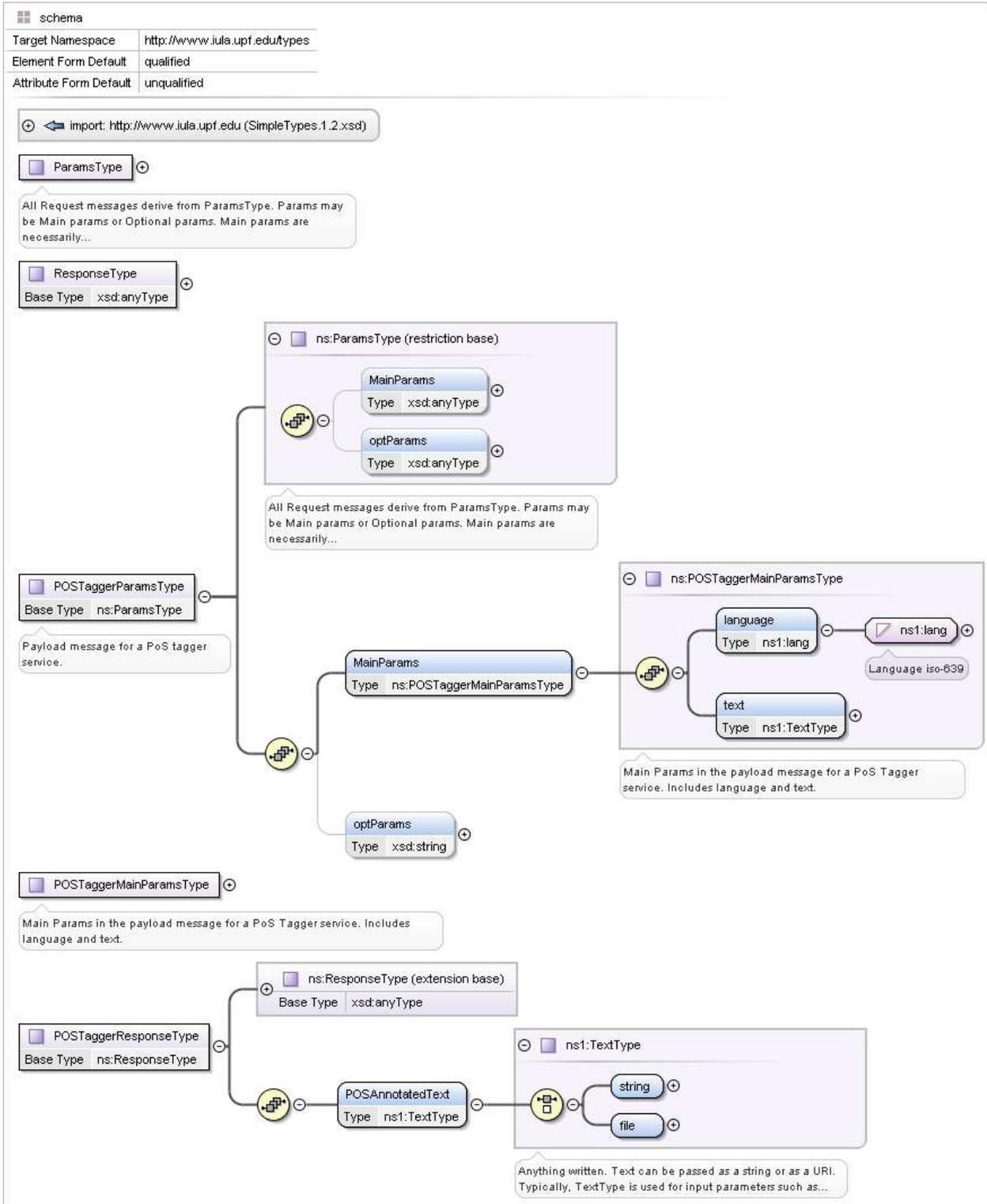
9.2.7.4 Named Entity Recognition



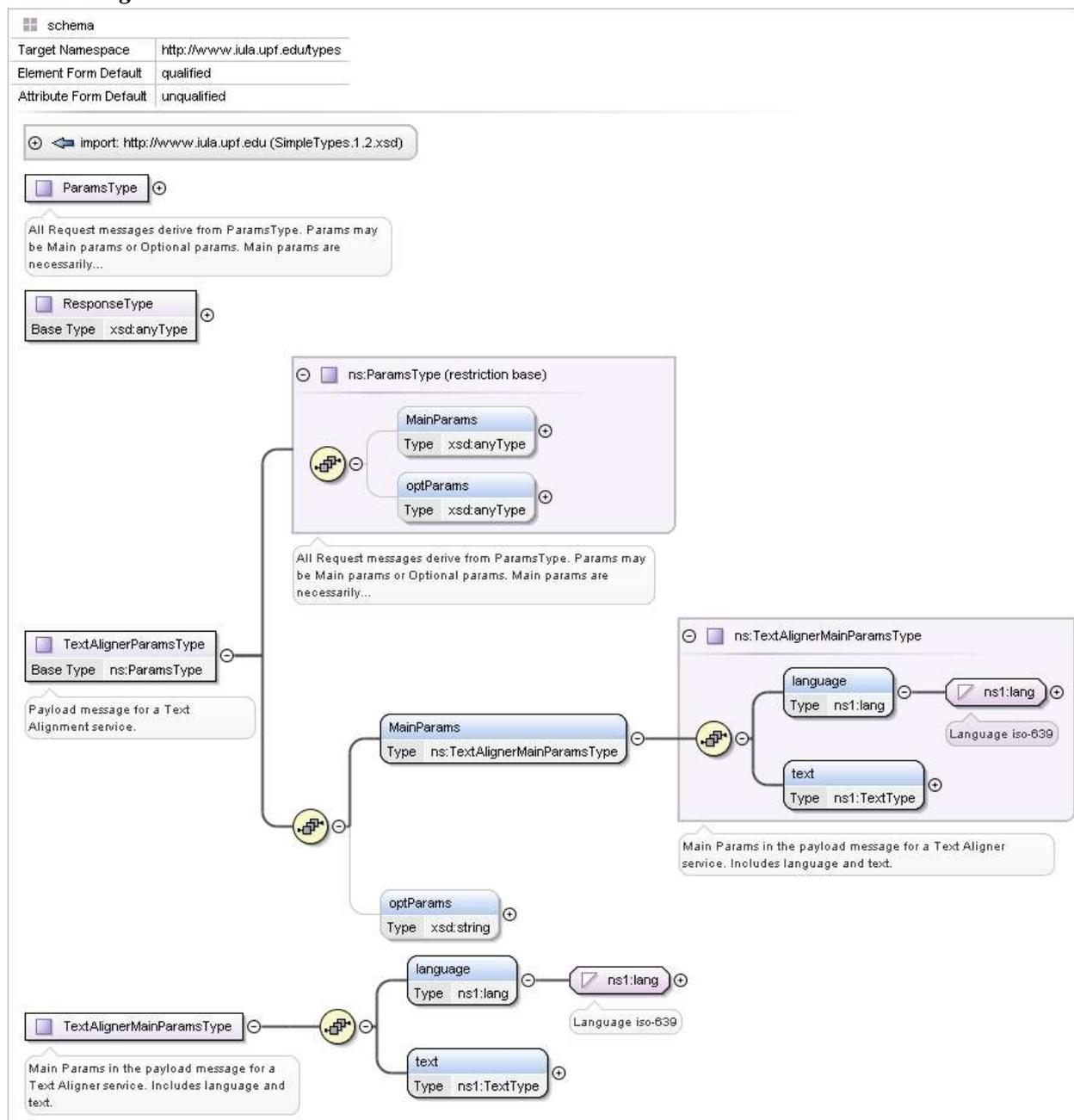
9.2.7.5 Lemmatization



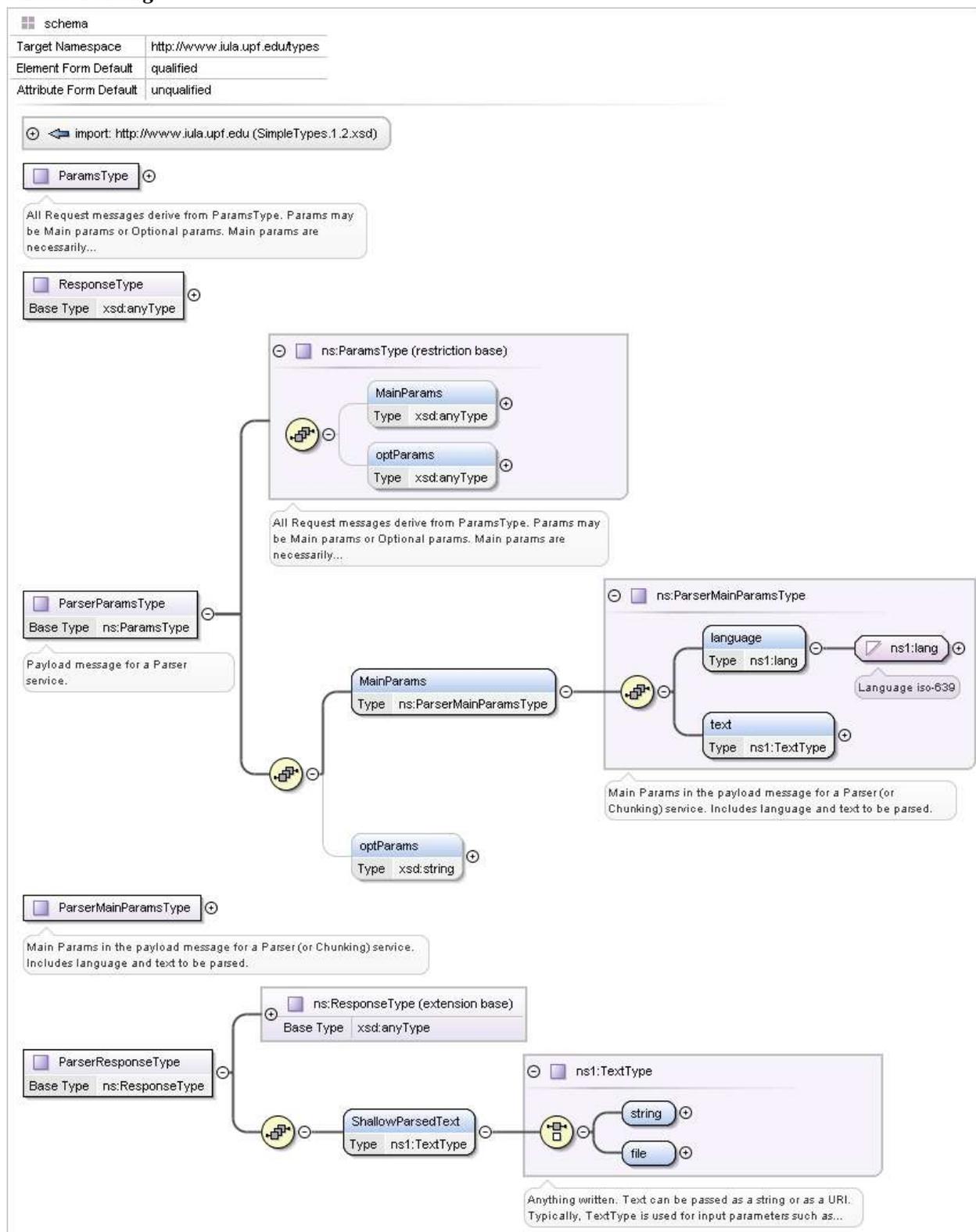
9.2.7.6 PoS tagging



9.2.7.7 Alignment



9.2.7.8 Parsing



9.2.7.9 Term Extraction

9.2.7.10 Topic Identification

