SEVENTH FRAMEWORK PROGRAMME

PANACEA

SEVENTH FRAMEWORK PROGRAMME
THEME 3
Information and communication Technologies

PANACEA Project

Grant Agreement no.: 248064

Platform for **A**utomatic, **N**ormalized **A**nnotation and
Cost-**E**ffective **A**cquisition
of Language Resources for Human Language Technologies

D3.2
First version (v1) of the integrated platform and documentation

Dissemination Level:	**Public**
Delivery Date:	FEBRUARY 28th 2011
Status – Version:	Final
Author(s) and Affiliation:	Marc Poch (UPF), Olivier Hamon (ELDA), Gregor Thurmair (Linguatec), Núria Bel (UPF)

Relevant Panacea Deliverables

D3.1	Architecture and Design of the Platform
D4.2	Initial functional prototype and documentation describing the initial CAA subsystem and its components.
D5.2	Aligners integrated into the Platform

UNIVERSITAT POMPEU FABRA · IEA LSP · UNIVERSITY OF CAMBRIDGE · linguatec language technologies · DCU · elda

This document is part of technical documentation generated in the PANACEA Project, **P**latform for **A**utomatic, **N**ormalized **A**nnotation and **C**ost-**E**ffective **A**cquisition (Grant Agreement no. 248064).

Please send feedback and questions on this document to: iulatrl@upf.edu

TRL Group (Tecnologies dels Recursos Lingüístics), Institut Universitari de Lingüística Aplicada, Universitat Pompeu Fabra (IULA-UPF)

Table of contents

1 Introduction

The first version of the PANACEA platform is operational with crawling (WP4) and alignment (WP5) tools deployed as web services. Workflows have been developed and shared among partners and the PANACEA Registry is fully functional. The aim of this document is to list and explain the work developed in WP3 and the resulting documentation.

2 PANACEA Platform

The PANACEA Platform is defined in this Section considering the technological options chosen in the design phase and according to deliverable D3.1 and specifying some of the topics that were still to be established.

The PANACEA Platform definition will be divided in two parts: a **Formal Definition** and a **Technical Definition**. The Formal Definition is an abstract description and will be used in all PANACEA platform versions. On the other hand, the Technical Definition is used to establish the PANACEA Platform characteristics and it may vary between the different versions of the PANACEA Platform.

Formal Definition: PANACEA platform is an **interoperability space** based on tools, guidelines, a common interface definition, and a "Travelling Object" specification.

Technical Definition:

PANACEA Platform Version 1:
Tools: Taverna[1], BioCatalogue[2], Soaplab[3].
Common Interface (CI): defined in deliverable D3.1.
Travelling Object (TO): defined in deliverable D3.1.
Documentation: manuals, guidelines, etc.

The following sections list and describe these specifications.

2.1 Tools / Software

2.1.1 Taverna

As explained in D3.1 Section 4.3.3, Taverna was chosen to be the workflow editor for PANACEA. Different experiments have been done to test some Taverna versions (there is one new release every year, approximately) in UPF. Specifically, versions 1.7 and 2.1.2 were experimented and later version 2.2 was adopted because of its new features and better compatibility with PANACEA web services.

2.1.2 BioCatalogue

According to the workplan in D3.1, BioCatalogue was chosen to be the registry software for PANACEA. The release of myBioCatalogue, which allows you to set up your own local BioCatalogue easily, was expected by the end of September 2010. However, ELDA deployed a

[1] http://www.taverna.org.uk
[2] http://www.biocatalogue.org
[3] http://soaplab.sourceforge.net/soaplab2

BioCatalogue instance without the myBioCatalogue software package since this release is still not available.

2.1.3 Soaplab

Soaplab was explained in D3.1 Section 4.6.1 as a wrapper: it is software that allows easy deployment of tools in case PANACEAas web services. Soaplab has proven to be very useful and user-friendly and it has fulfilled all PANACEA Platform version 1 requirements regarding web services deployment. In fact, most partners have deployed more web services than required by PANACEA. On the other hand, some web services have been deployed using Axis2[4] technology at UPF. Although these tests were successful, Soaplab continues to provide many more features with fewer resources (in terms of time to deploy and to make modifications). While Soaplab will remain the main technology used to deploy web services, other technologies (e.g. Axis2) will still be analyzed and tested in case PANACEA needs them.

Several tests were done at UPF to find the best Soaplab deployment method and parameters usage for PANACEA. While previous Taverna versions were compatible with all kind of Soaplab protocols (JAXWS[5] and Axis1[6]), new Taverna versions (version 2.2.0) lack the necessary plugin for JAXWS protocol. This plugin is provided by Soaplab developers, who must release a new plugin for every new Taverna relase. On the other hand, Taverna developers adapt any new release to the Axis1 protocol making it the best option to guarantee interoperability.

2.2 Common Interface

The Common Interface for different kinds of tools was designed and presented in D3.1 Section 6.2. As it was explained, it is presented as an XML schema file containing all the designed *types*. However, it has been documented in different ways to improve presentation and help service providers. This documentation is presented in Section 2.4.1 of this document.

This Common Interface (CI) definition is very detailed (the whole schema is defined) and in some situations (e.g. Soaplab deployments) it is enough to follow it to a certain level (mandatory and optional parameter division) to guarantee interoperability. However, the full design of the CI can be used to deploy Axis2 web services with PANACEA compatibility. Soaplab adoption has been very helpful for service providers (all partners deploying web services) but its technology does not allow the service provider to modify its schema files (there are some Soaplab options about this but they have not been analyzed yet). In this scenario, the PANACEA service provider can follow the CI definition by dividing the parameters of its web services in mandatory and optional parameters following the CI design. The service provider must also follow the names of the parameters for each kind of tool described in the CI documentation. For example, two Soplab aligners following the CI will be very easily interchanged because its mandatory parameters are the same and have the exact same names.

[4] http://axis.apache.org/axis2/java/core/

[5] http://jax-ws.java.net/

[6] http://axis.apache.org/axis/

Some changes have been made to the original proposal in D3.1. Parameter "text" has been renamed to "input" and CI defined for aligners has been redesigned. All these changes are included in the documentation (Section 2.4.1).

2.3 Travelling Object

The Travelling Object (TO) has been used to transport data between components in PANACEA and it has not been changed from its original design. It was documented in deliverable D3.1 Section 6.1. To make *in-house* formats compatible with the TO, service providers had to develop a conversion tool. Most partners have adapted the Java tool developed by ILSP (called *FormatConverter*) which is documented in D4.2 Section 4. FormatConverter has been used to modify data inside a web service (web service with the tool and FormatConverter). However, some partners have deployed the FormatConverter as a web service (one web service with the tool and another web service with FormatConverter).

ILSP also developed an XSLT stylesheet to visualize the TO in a more user friendly way than XML. It is documented in D4.2 Section 4 and it is posted on the PANACEA web site[7].

2.4 Documentation: manuals, guidelines, articles

This Section is devoted to list and describe the documentation developed for WP3.

2.4.1 Common Interface documentation

The Common Interface documentation can be found on the PANACEA web site (*documents*[8] Section of *Info for Professionals*) and in this deliverable. It consists of four documents:

- **types1.2.xsd**: the types file.
- **PANACEA-CI_documentation_v01.2.pdf**: documentation for the service providers about the CI. It contains the basic and necessary information for the service providers.
- **types1.2.pdf**: detailed documentation about the CI.
- **Types1.2 Web documentation**: Internet version of the detailed documentation about the CI.

2.4.2 Travelling Object documentation

The Travelling Object was described in D3.1. There is no further documentation since there could be major changes in it for the next version of the platform.

2.4.3 Web services documentation

Web services can be documented in many different ways. For PANACEA platform version 1, web service providers must document their web services using the Soaplab metadata file (ACD[9]) and the registration process at the registry. This is the minimum amount of documentation all web service providers must provide.

[7] http://panacea-lr.eu/system/xces-xslt/annotCesDoc.xsl

[8] http://panacea-lr.eu/en/info-for-professionals/documents/

[9] ACD: Ajax Command Definition.

The ACD file is used to describe the web service: the script to be run, the parameters, help messages, etc. Thus, the web service providers are encouraged to provide as precise and descriptive information as possible to help web service users.

When a web service is registered, most metadata is extracted automatically by the registry and presented to the users. However, there can be an extra process of documentation called annotation. Web service providers can categorize the web service, add tags, fill in some forms with metadata, etc. A better annotated web service will be used by more users if it is easier to find (the BioCatalogue web site has more than 1,000 web services), its functionality is better understood and its technical aspects are better described.

2.4.4 Workflows documentation

For PANACEA Platform version 1 the workflows have no specific documentation (apart from deliverables like D4.2 and D5.2). Most of them are for test purposes and may require changes and improvements for the next version of the platform (because of massive data adaptation or changes in the TO).

As described in the D3.1 workplan and after the gained experience during development of platform version 1, a web portal to share workflows can be very useful. For platform version 2 a portal will be used and then the documentation task of workflows will be described and designed.

2.4.5 Frequently asked questions

A list of FAQs has been developed for fast access to some typical questions, tips and tricks. It can be found on the PANACEA web site (*FAQs*[10] Section of *Info for Professionals*).

2.4.6 PANACEA tutorial

A tutorial has been developed which can be used by users and service providers to find information and links to all necessary documentation to work with the PANACEA Platform. It is posted on the PANACEA web site (*documents*[11] Section of *Info for Professionals*).

2.4.7 Articles

The article "*Adapting Bioinformatics tools to work with NLP*" has been submitted to the magazine "*La Sociedad Española para el Procesamiento del Lenguaje Natural*" (SEPLN) and is included in this deliverable zip file.

3 Deployed web services

The list of deployed web services is presented in this Section and some relevant web services for PANACEA platform version 1 and other deliverables are commented.

[10] http://www.panacea-lr.eu/en/info-for-professionals/faqs/

[11] http://www.panacea-lr.eu/en/info-for-professionals/documents/

Current status of registered web services at the PANACEA registry (up to 09-02-2011):

Name	Type	Category	Provider (machine name)	Registry Number
freeling_morpho	Soaplab	Morphosyntactic Tagging	gilmere-upf-edu	25
freeling_tagging	Soaplab	Morphosyntactic Tagging	gilmere-upf-edu	26
Freeling_tokenizer	Soaplab	Tokenization	gilmere-upf-edu	28
freeling_dependency	Soaplab	Syntactic Tagging	gilmere-upf-edu	33
freeling_parsed	Soaplab	Syntactic Tagging	gilmere-upf-edu	34
freeling	Soaplab	Morphosyntactic Tagging, Tokenization, Syntactic Tagging	gilmere-upf-edu	24, 27, 32
kwic	Soaplab	Text Mining	gilmere-upf-edu	31
calcular_p_cue_class	Soaplab	Statistics Analysis	gilmere-upf-edu	35
ngrams	Soaplab	Statistics Analysis	gilmere-upf-edu	36
tfidf	Soaplab	Statistics Analysis	gilmere-upf-edu	37
catdoc	Soaplab	Format Conversion	gilmere-upf-edu	38
html2text	Soaplab	Format Conversion	gilmere-upf-edu	39
panacea_conversor	Soaplab	Format Conversion	gilmere-upf-edu	40
pdftotext	Soaplab	Format Conversion	gilmere-upf-edu	41
iula_tagger	Soaplab	Morphosyntactic Tagging	kurwenal-upf-	50

			edu	
iula_tokenizer	Soaplab	Tokenization	kurwenal-upf-edu	51
iula_lexicon_lookup	Soaplab	Stemming/Lemmatization	kurwenal-upf-edu	52
iula_preprocess	Soaplab	Corpus Processing	kurwenal-upf-edu	53
boilerplate_remover	Soaplab	Corpus Processing	sifnos-ilsp-gr	54
ilsp_bilingual_crawl	Soaplab	Crawling, Corpus Processing	sifnos-ilsp-gr	55
ilsp_fbt	Soaplab	Corpus Processing, Morphosyntactic Tagging	sifnos-ilsp-gr	56
ilsp_lemmatizer	Soaplab	Corpus Processing, Stemming/Lemmatization	sifnos-ilsp-gr	57
ilsp_mono_crawl	Soaplab	Crawling, Corpus Processing	sifnos-ilsp-gr	58
ilsp_sst	Soaplab	Corpus Processing, Tokenization	sifnos-ilsp-gr	59
aligner2to	Soaplab	Alignment, Format Conversion	www-cngl-ie	69
berkeley_aligner	Soaplab	Alignment	www-cngl-ie	70
berkeley_parser	Soaplab	Syntactic Tagging	www-cngl-ie	71
berkeley_tagger	Soaplab	Morphological Tagging	www-cngl-ie	72
berkeley_tagger2to	Soaplab	Format Conversion	www-cngl-ie	73
chunk_aligner	Soaplab	Alignment	www-cngl-ie	74
europarl_lowercase	Soaplab	Format Conversion	www-cngl-ie	75
Europarl_ sentence_splitter	Soaplab	Tokenization	www-cngl-ie	76
europarl_tokeniser	Soaplab	Tokenization	www-cngl-ie	77
gizapp	Soaplab	Alignment	www-cngl-ie	78

hunalign	Soaplab	Alignment	www-cngl-ie	9
Converter_Freeling_TO	Soaplab	Format Conversion	wiki2-ilc-cnr-it	87
Converter_KAF_TO	Soaplab	Format Conversion	wiki2-ilc-cnr-it	88
Freeling_It	Soaplab	Tokenization, Morphological Tagging, Morphosyntactic Tagging	wiki2-ilc-cnr-it	89
Freeling_outofthebox	Soaplab	Syntactic Tagging, Tokenization, Morphological Tagging, Morphosyntactic Tagging	wiki2-ilc-cnr-it	90
anymalign	Soaplab	Alignment	www-cngl-ie	91
bsa	Soaplab	Alignment	www-cngl-ie	92
gma	Soaplab	Alignment	www-cngl-ie	93
sentalg_tok_to2word_alg	Soaplab	Format Conversion	www-cngl-ie	94
sentsplit_tok2to	Soaplab	Format Conversion	www-cngl-ie	95
freeling	Soap	Morphosyntactic Tagging	193-145-50-98 (UPF)	3

3.1 WP3 web services

In this Section some of the deployed web services are explained regarding their relevance to the first version of the platform interoperability.

PANACEA Conversor

UPF has deployed the PANACEA Conversor web service based on the FormatConverter Java tool. This web service allows users to convert from *in-house* formats to the PANACEA TO (and back). This approach has several advantages:

- Service providers can deploy tools as web services without format modifications, thus simplifying the web service development.
- Users can access the web service using *in-house* formats.
- If there is a change in the TO, the service providers only need to update one web service instead of all the tools.

This approach has been followed by other partners and has proven to be very efficient. By adding a little more complexity to the workflows (an extra web service must be called to convert to the TO) both the maintenance of the web services and the flexibility of the platform have been improved. The following list presents the conversion web services deployed following this approach.

aligner2to

DCU has deployed this web service which converts aligner output to TO format. This web service is explained in D5.2.

sentalg_tok_to2word_alg

DCU has deployed this web service which converts from TO to word alignment input. This web service is explained in D5.2.

sentsplit_tok2to

DCU has deployed this web service which converts sentence-split tokenised text (Europarl format) to TO format. This web service is explained in D5.2.

Converter_Freeling_TO

ILC has deployed this web service which converts from Freeling format to TO.

Converter_KAF_TO

ILC has deployed this web service which converts from KAF[12] format to TO.

3.2 WP4 web services

In this Section, we list the most relevant web services deployed for WP4.

- **Focused monolingual crawler**: a web crawler which collects data from web pages with content in one language and a predefined domain.
- **Focused bilingual crawler**: a web crawler for multilingual web sites which downloads data from a predefined topic in the target languages.
- **Boilerplate remover**: a web service devoted to eliminate undesired text from web crawled data like links, advertisements, etc.

All these web services and the tools behind them are explained in deliverable D4.2.

3.3 WP5 web services

In this Section, we list the most relevant web services deployed for WP5.

Sentential alignment:

- **Hunalign**: based on Hunalign tool.

[12] KYOTO/Knowledge Annotation Format. KYOTO project (ICT-211423).

- **GMA**: based on Geometric Mapping and Alignment software.
- **BSA**: based on Bilingual Sentence Aligner tool.

Sub-sentential alignment:

- **Giza**++: based on Giza++ program.
- **BerckeleyAligner**: based on Berckeley Aligner.
- **OpenMaTrEx chunk aligner**: based on the OpenMaTrEx translation system.
- **Anymalign**: based on Anymalign software.

Travelling Object conversion:

These web services were presented in Section 3.1.

- **aligner2to**: converts aligner output to TO format.
- **sentalg_tok_to2word_alg**: converts from TO to word alignment input.
- **sentsplit_tok2to**: converts sentence-split tokenised text (Europarl format) to TO format.

All these web services and the tools behind them are explained in D5.2.

4 The Registry

4.1 Deployment

From the registry options defined in D3.1, BioCatalogue[13] is the one PANACEA is currently using to provide a web service registry. At first, it was planned to use myBioCatalogue, a BioCatalogue specifically designed to obtain and use local BioCatalogue for a specific project. This has not been released yet, and then it has been decided to use the BioCatalogue development version provided on the web site (which lacks specific documentation).

Thus, a PANACEA version of BioCatalogue has been installed on a local server at ELDA so as to conduct our own modifications and design at the beginning of the project and when needed. This means that the PANACEA catalogue of web services is independent from any other project and may follow its own development. The registry installed for PANACEA is available at http://registry.elda.org.

Web service providers can submit a new web service (providing its WSDL), or a list of web services from a Soaplab server or from a REST[14] server. A server daemon regularly checks for available services and Soaplab servers. Therefore, each time a new service is submitted, users have to wait for a few minutes before the service is available. Web service providers can add different kinds of annotations to document their web services. Web services can be categorized, tagged and different forms can be filled to add extra information.

[13] http://www.biocatalogue.org

[14] Representational State Transfer (REST). Explained in D3.1 Section 4.2.3

Web services can be searched using categories and tags. The search field is compatible with wildcards like asterisk (*) and question mark (?). For example, **crawl*** would find tags like **crawler** and **crawling**.

Furthermore, a monitoring system checks automatically and regularly the status of registered services and tests if they are still available. Namely, a service test may be *passed* (all tests are successful for the service), in *warning alert* (some or all the tests for the service did not succeed) or *failed* (it is not possible to reach the service). Each time the status of a web service changes, an e-mail is sent to an administrator.

After experiencing some issues regarding the installation and setup of BioCatalogue (related to updates, version inconsistencies, database, etc.), the PANACEA catalogue is now stable and works properly, since daemons and monitoring systems have not been restarted for several weeks. Until now (t14), 50 services have been registered from DCU, ILC, ILSP and UPF.

According to D3.1, the following requirements are available:

- List the available services;
- Allow a search function;
- Allow the addition and removal of web services;
- Provide formal descriptions;
- Check and show web service statuses.

The requirement "Allow user management" is fulfilled in part, since users may register, but no control is made on who is registering.

4.2 Modifications

Since detailed documentation on BioCatalogue is not available, modifications and development around the PANACEA catalogue was not an easy task. Moreover, configuration options are mainly hard-coded. Therefore, contacts have been made with BioCatalogue developers each time it was needed, by e-mail and through the "BioCatalogue-hackers" list, a mailing list created for people using BioCatalogue.

In a preliminary version, the monitoring system was launched manually. It has been adapted so as to run as a daemon and update service status regularly.

Furthermore, the BioCatalogue categories have been adapted to fit with the PANACEA features. These have been taken from the metadata defined within the BAMDES proposal (Parra et al., 2010). A total of 33 categories are available on the registry and, for the time being (t14), 12 are used by service providers.

Finally, the design of the registry web page has been modified, according to several aspects:

- Substitution of the BioCatalogue logo by the PANACEA logo;
- Modification of the main page;
- Addition of the PANACEA partners' logos;
- Modification of the layout, mainly the colours, in agreement with the PANACEA web site.

5 Workflows and Taverna

Several relevant workflows are listed in this Section.

5.1 WP4 workflows

- Focused monolingual crawler
- Focused bilingual crawler

These workflows and their documentation can be found on deliverable D4.2.

5.2 WP5 workflows

- Sentence alignment using Hunalign
- Sentence alignment using BSA
- Sentence alignment using GMA
- Word alignment using Giza++
- Word alignment using BerkeleyAligner
- Chunk alignment using OpenMaTrEx
- Combined workflow to perform sentence alignment and word alignment

All these workflows and their documentation can be found in D5.2.

5.3 WP3 workflows

Work package 3 workflows are some of the workflows developed at UPF. A few of them were developed to be used as models to build more complex workflows and to learn necessary tools and characteristics of Taverna. There is also one workflow combining WP4 and WP5 workflows and web services.

5.3.1 Bilingual crawler output language splitter

The aim of this workflow (*bilingual_crawler_output_lang_splitter_v01* shown in **Figure 1**) is to split the output data of the bilingual crawler by language. Every document in each language should be processed separately, and at the same time the information about the documents being parallel should be maintained.

The output of the crawler is a list of URLs of cesAlign documents each of them pointing to a pair of cesDoc documents (cesAlign and cesDoc are TO documents). There are examples of cesAlign and cesDoc documents from the crawler in the D4.2 Annex.

In this scenario two problems arise: multiple file processing and accessing data inside an XML file (cesAlign) from the workflow. The multiple file processing issue can be solved in Taverna by using several components and options. There is detailed online documentation (*Advanced workflow configuration*[15]) to train workflow developers how to handle *lists* (multiple files, parameters, data sets, etc.) in the myGrid[16] web site.

[15] http://www.mygrid.org.uk/dev/wiki/display/taverna/Advanced+workflow+configuration

[16] MyGrid is explained in D3.1, Section 4.4.1.

The *url_list_split* component is based on the "*Split string into string list by regular expression*" local processor. This processor can divide a text in different parts creating a list (in our workflow a list of URLs). Then, Taverna will automatically execute the rest of the components for every element of that list.

The *Xpath* local processor is a Taverna component which can be used to execute Xpath queries in an XML file. In our workflow this processor is used to extract the URL of one specific language document. For example, the xpath query used for Spanish is:

```
//translation[@lang='en']/@trans.loc
```

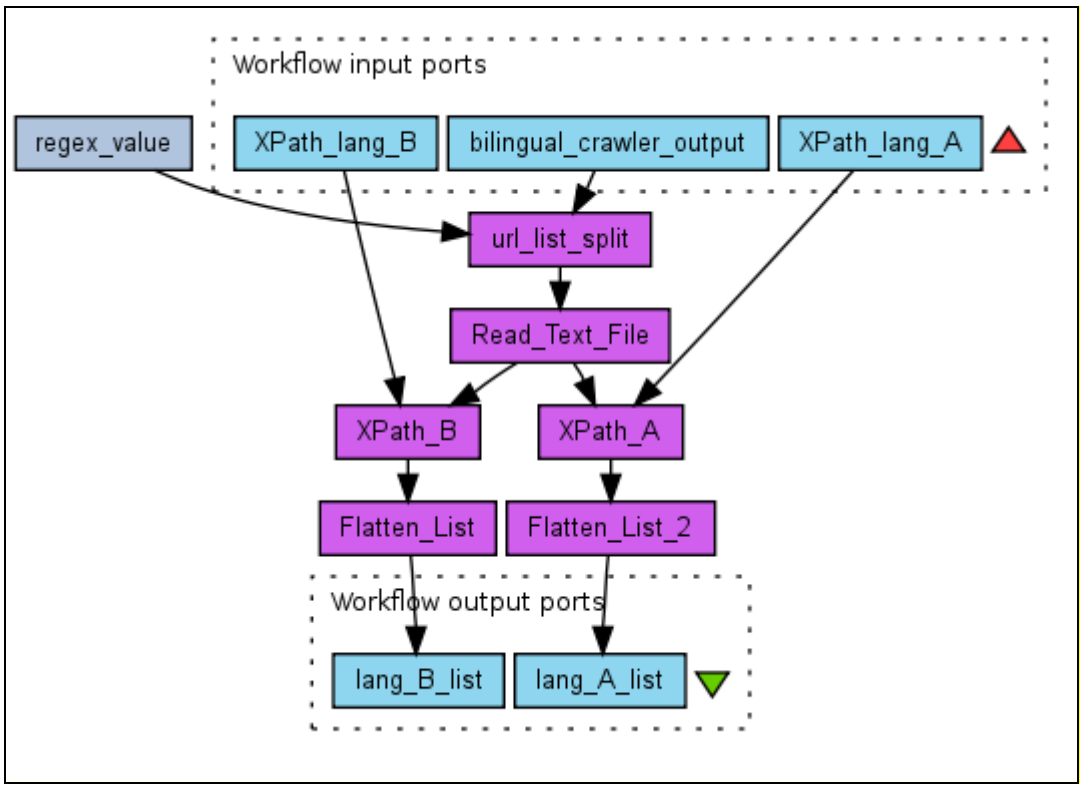

Figure 1. *bilingual_crawler_output_lang_splitter_v01*

5.3.2 Bilingual crawler to plain text

This workflow (*bilingual_crawler_to_plain_text_v01* shown in **Figure 2**) was used to show how Taverna can import already existing workflows to create more complex structures. In this case, the *panacea_conversor* web service is used to extract plain text from the cesDoc documents.

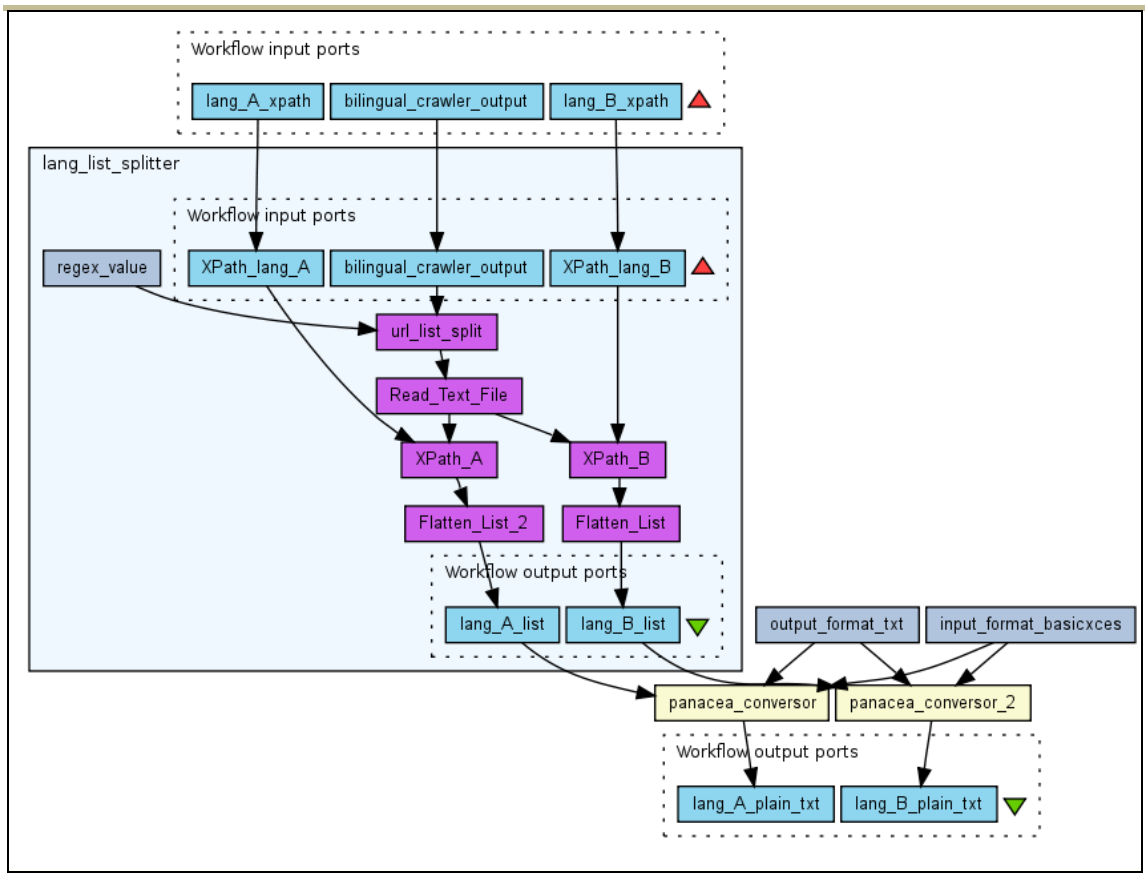

Figure 2. *bilingual_crawler_to_plain_text_v01*

5.3.3 Bilingual process for Spanish and English

This workflow (*bilingual_process_ES_EN_v04*) was designed for testing purposes. It combines different nested workflows and shows how different language documents are processed following different paths. One path is for Spanish and the other one for English. Each path is based on an already existing workflow.

This workflow's schema is shown in **Figure 3** and **Figure 4** of the Annex 9.3.

5.3.4 Bilingual crawler sentence aligner for Spanish and English

WP4 and WP5 workflows and web services are combined in this workflow (*bilingual_crawler_sental_ES_EN*) to create a complex workflow integrating the bilingual crawler and a sentence aligner. At the end the data is transformed with the *annotCesDoc.xsl* stylesheet to make it more human readable.

This workflow's schema is shown in **Figure 5** and **Figure 6** of the Annex 9.4.

5.4 Taverna on Windows

Linguatec has done some research about Taverna input data capabilities under Windows. Some problems were detected and reported. Some of them can be solved and all this information has been reported to Taverna developers.

Problem 1
When running Taverna under Windows, the system I/O functions Read Text File and Write Text File do not behave as expected, in case of UTF-8 files.

The reason is that the readers and writers assume the standard character encoding, in case nothing else is specified; this is CP1252 under Windows; as a result, the UTF-8 characters are broken.

Fix:
To avoid this, the two scripts for reading and writing text files must be overwritten. The necessary code is presented in the Annex 9.1 and 9.2.

Problem 2
Now, while this can be done explicitly in cases where the *Beanshell* code can be edited, it cannot be done in cases where an input port is used; there is no option to edit the underlying code.

Modifying the batch scripts (*executeworkflow.bat*) by adding the line:

```
set ARGS=%ARGS% -Dfile.encoding=UTF-8
```

was not successful either.

The options are then:

a. Modify the readers / writers of the input / output ports (in case it's possible).
b. Use the explicit *Read* / *Write* Text File options, and modify them as described. Drawback: changing filenames for a run always means to edit the workflow.
c. The Taverna software is adapted so that it enforces UTF-8 processing by e.g. taking over the character code setting as proposed above, or by any other action.

This problem has been classified as "critical" by Taverna developers and it might be solved in future releases of Taverna.

These two problems will have a very limited effect on PANACEA since all the designed workflows avoid these situations. Using URL files to move data and parameters which do not require UTF-8 is enough. However, it is very valuable to detect these problems soon, because new workflows can be designed properly and Taverna developers are already aware of the situation.

6 Workplan updates

The workplan updates are listed and described in this Section.

Temporary files management
WS-GR-01. ILC has developed a research and development task for the temporary files management. Soaplab handles temporary files automatically but the deletion of old files is an important issue. ILC developed a tool to erase temporary files. However, since the real challenge for temporary files management is massive data handling (not only deletion, but storage control, etc.), this task will be further addressed in D3.3 (t22).

Provenance
WS-GR-02. ILC has managed to extract data from the Taverna automatic provenance system (Taverna handles provenance using an intern database). This complex task is being done outside the standard Taverna provenance system which is useful but cannot be extracted. Taverna is releasing a new workbench at the end of February, which could bring some changes to the actual situation. The provenance data issue will become more important in the version of PANACEA handling massive data, where the Taverna Server solution will be examined. Further details will be provided in the context of D3.3 (t22).

Soaplab Test massive data
WS-SL-01. Massive data can be handled in two different ways: as large collections of small (or average-sized) files, or as small collections of large files. UPF has used some workflows to test large files with Soaplab web services. File sizes were tested successfully from 100 KB to 100MB. However, bigger files should be tested in combination with the new Taverna Server (to be released in February 2011). The scenario with many small files has been tested up to 50 files. Again, these tests must be further examined in combination with the Taverna Server. This task and its result will be presented in D3.3 (t22).

Axis web services
All tasks regarding Axis web services are depending on the Soaplab success. UPF has developed different web services using Axis2. Some of these web services are being successfully used in other projects. However, the cost of development and maintenance of these web services is much bigger than using Soaplab. Moreover, Axis2 web services, with this development stage, provide fewer features than Soaplab: more complex error handling and temporary files management, etc. From this point of view, Axis web service development tasks will be postponed and used as a fallback position or for some specific situations.

Taverna Server test
WF-TV-02. Some tests with Taverna server were done at UPF. However, they were conducted at an early development stage of Platform version 1, when there was a lack of real workflows and finalized Soaplab web services. The new Taverna Server to be released in February 2011 makes it more interesting to continue those tests with the new version in order to address the real challenge of massive data with the now more stable PANACEA web services. This task and its result will be presented in D3.3 (t22).

7 Conclusions and future work

In this document, the first version of the platform for the PANACEA project has been described. A concrete and precise definition of the platform has been provided, together with references to all relevant documentation. This information can be used by users and service providers towards developing a concrete version of the platform.

Web services and workflows have been presented with the necessary Common Interfaces and Travelling Object definitions. The registry is operational and adapted to PANACEA. The experiences of the registry deployment as well as its capabilities have been explained in this document.

PANACEA platform 1 has given to PANACEA developers the necessary know-how to face the coming challenges. Large data management will probably be the most difficult requirement for

the platform, not only for web services capabilities but also for workflows design and Taverna (workbench and server) features.

8 Bibliography

[Bamdes] C. Parra, M. Villegas, N. Bel. (2010). "The BAsic Metadata DEScription (BAMDES) and TheHarvestingDay.eu: Towards Sustainability and Visibility of LRT", Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10). Paris, France: European Language Resources Association (ELRA). pag. 49-53.

[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.

[Deliverable D3.1] M. Poch, P. Prokopidis, G. Thurmair, C. Schnober, R. Del Gratta, and N. Bel. 2010. D3.1 - architecture and design of the platform. Confidential deliverable, The PANACEA Project (7FP-ITC-248064).

[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. Riceand T. Oinn. "Soaplab - a unified Sesame door to analysis tools (2003)" In UK e-Science All Hands Meeting.

[Taverna] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. Pocock, P. Li, and T. Oinn, "Taverna: a tool for build-ing 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.

9 Annex

9.1 Read Text File

Original:

```
BufferedReadergetReader (String fileUrl) throws IOException {
          InputStreamReader reader;
          try {
                reader = new FileReader(fileUrl);
          }
          catch (FileNotFoundException e) {
                // try a real URL instead
                URL url = new URL(fileUrl);
                reader = new InputStreamReader (url.openStream());
          }
          return new BufferedReader(reader);
     }
StringBuffersb = new StringBuffer(4000);
```

```
BufferedReader in = getReader(fileurl);
String str;
String lineEnding = System.getProperty("line.separator");

while ((str = in.readLine()) != null) {
     sb.append(str);
     sb.append(lineEnding);
}
in.close();
filecontents = sb.toString();
```

The code should be:

```
BufferedReadergetReader (String fileUrl) throws IOException {
          InputStreamReader reader;
          try {
                reader = new InputStreamReader(new
FileInputStream(fileUrl),"UTF-8");
          }
          catch (FileNotFoundException e) {
                // try a real URL instead
                URL url = new URL(fileUrl);
                reader = new InputStreamReader (url.openStream(), "UTF-8");
          }
          return new BufferedReader(reader);
     }
StringBuffersb = new StringBuffer(4000);
BufferedReader in = getReader(fileurl);
String str;
String lineEnding = System.getProperty("line.separator");

while ((str = in.readLine()) != null) {
     sb.append(str);
     sb.append(lineEnding);
}
in.close();
filecontents = sb.toString();
```

9.2 Write Text File

Original:

```
BufferedWriter out = new BufferedWriter(new FileWriter(outputFile));
out.write(filecontents);
out.close();
outputFile = filecontents;
```

The code should be:

```
BufferedWriter out = new BufferedWriter(new OutputStreamWriter(new
          \\     FileOutputStream(outputFile), "UTF-8"));
out.write(filecontents);
out.close();
outputFile = filecontents;
```

9.3 Bilingual process for Spanish and English

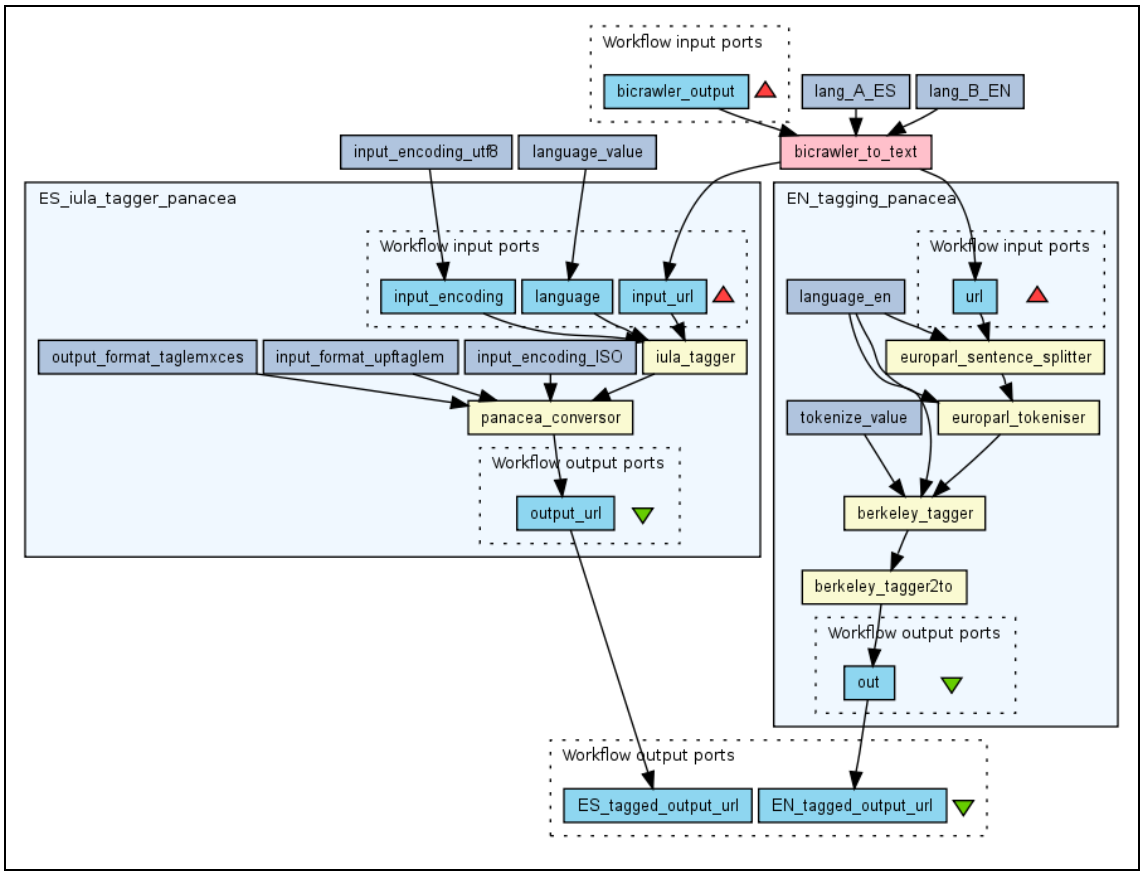

Figure 3. *bilingual_process_ES_EN_v04*

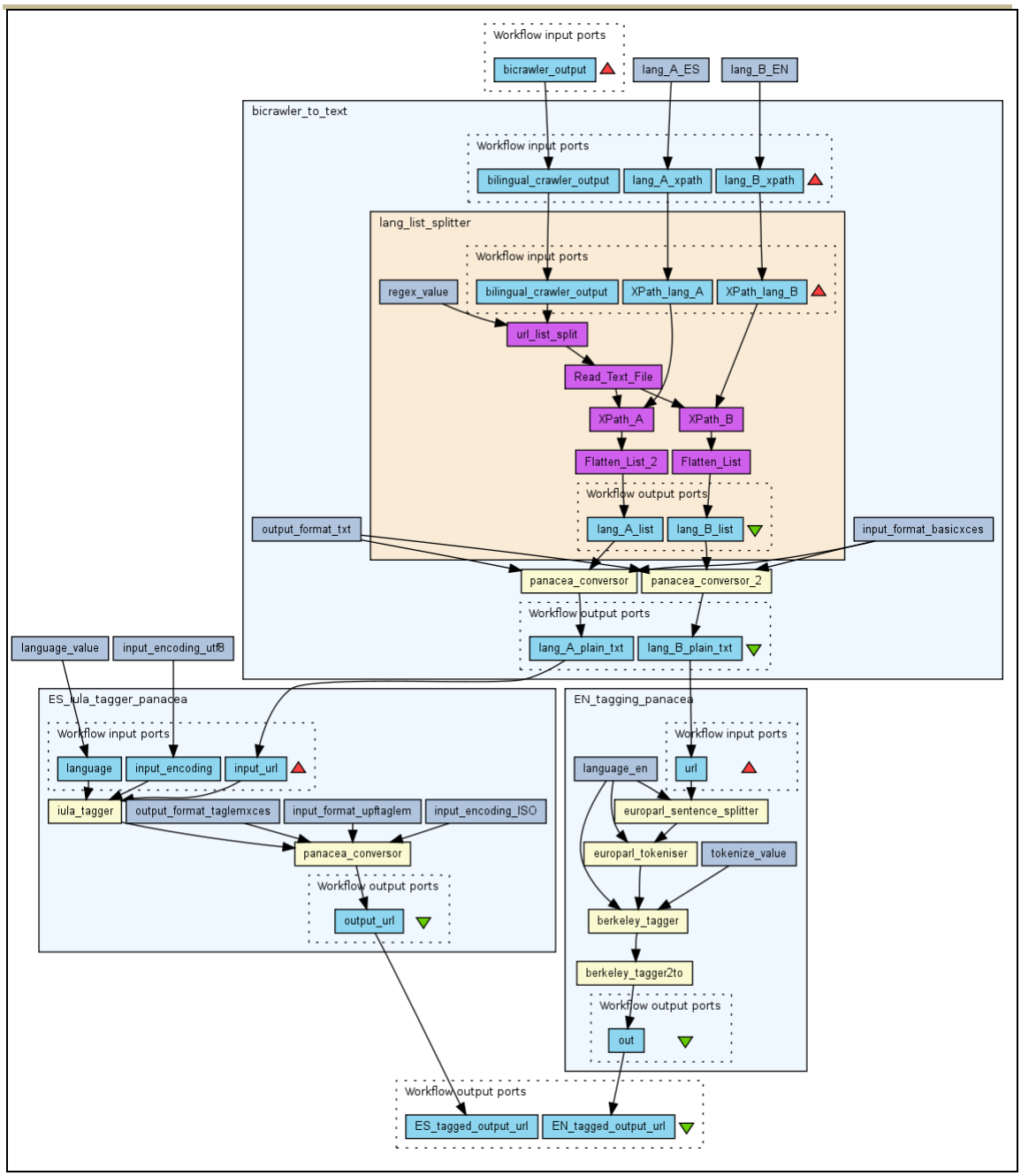

Figure 4. *bilingual_process_ES_EN_v04 (expanded)*

9.4 Bilingual crawler sentence aligner for Spanish and English

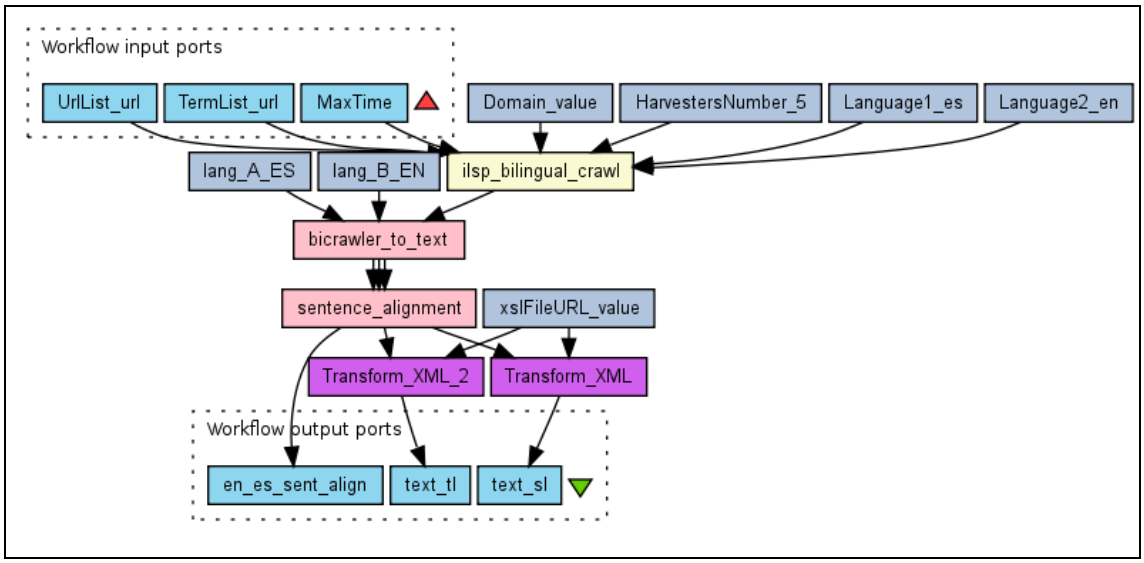

Figure 5. *bilingual_crawler_sental_ES_EN*

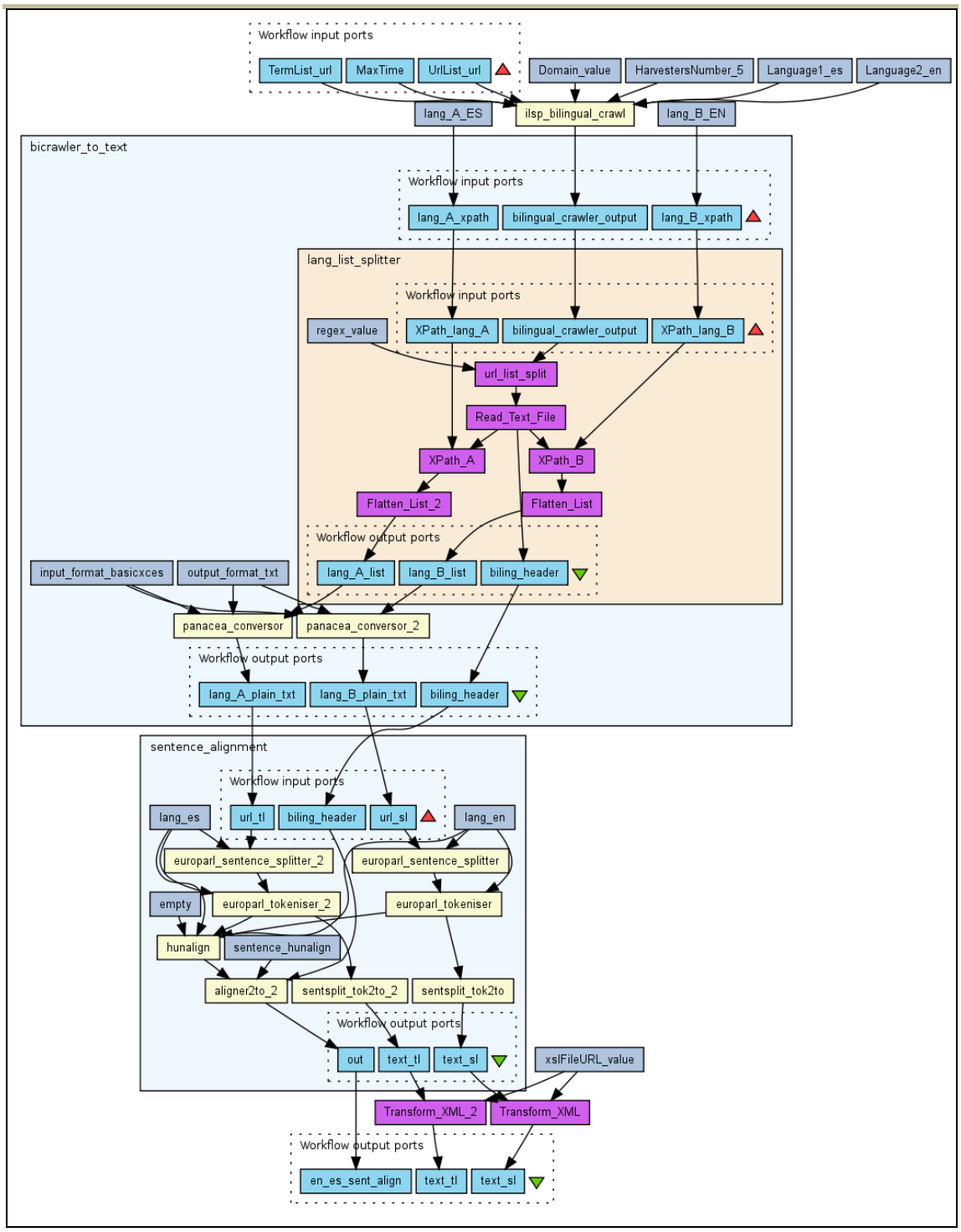

Figure 6. *bilingual_crawler_sental_ES_EN (expanded)*