SEVENTH FRAMEWORK PROGRAMME
THEME 3
Information and communication Technologies

PANACEA Project

Grant Agreement no.: 248064

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

D3.3
Second version (v2) of the integrated platform and documentation

Dissemination Level:	Public
Delivery Date:	October 21th 2011
Status – Version:	Final v01
Author(s) and Affiliation:	Marc Poch (UPF), Olivier Hamon (ELDA), Antonio Toral (DCU), Prokopis Prokopidis (ILSP), Roberto Bartolini (CNR-ILC), Francesco Rubino (CNR-ILC), Gregor Thurmair (LG), Vassilis Papavassiliou (ILSP) Núria Bel (UPF)

Relevant Panacea Deliverables

D3.1	Architecture and Design of the Platform
D3.2	First version (v1) of the integrated platform and documentation
D7.2	First evaluation report. Evaluation of PANACEA v1 and produced resources

D3.3 Second version (v2) of the integrated platform and documentation

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 2nd version of the platform is working and WP4 CAA and WP5 aligners are deployed. The Registry, deployed for the 1st version of the platform, is operational and it has now 70 registered web services. The PANACEA myExperiment portal has been deployed to share workflows among users who can execute those workflows with Taverna. Massive data solutions have been developed for users and web service providers and workflows have been redesigned and improve to handle larger amounts of data. This deliverable will present all the work developed for WP3 and its documentation.

2 Panacea Platform definition (version 2)

The Panacea Platform is defined in this section considering the technological options chosen in the design phase and according to deliverable D3.1.

The Panacea Platform definition will be divided in two parts: a **Stable Definition** and a **Variable Definition**. The Stable Definition is an abstract description and will be used in all Panacea platform versions. On the other hand, the Variable Definition is used to establish the Panacea Platform characteristics and it may have differences between Panacea Platform versions.

Stable Definition: Panacea platform is an **interoperability space** based on tools, guidelines, a common interface definition, and a "travelling Object" specification.

Variable Definition:

> **Panacea Platform Version 2**:
> **Tools**: Taverna[1], BioCatalogue[2], Soaplab[3]. myExperiment[4],
> **Common Interface (CI)**: defined in deliverable D3.1.
> **Travelling Object (TO)**: TO1 defined in deliverable D3.1 and GrAF[5].
> **Documentation**: Manuals, guidelines, etc.
> (New tools or specifications with respect to the platform version 1 are underlined)

The following sections are going to list and describe these specifications or reference another document.

[1] http://www.taverna.org.uk

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

[3] http://soaplab.sourceforge.net/soaplab2

[4] http://www.myexperiment.org

[5] The Graph Annotation Format (Ide and Su-dermam, 2007)

2.1 Tools / Software

2.1.1 Soaplab

Soaplab is the wrapper that allows service providers to easily deploy command line tools as web services. The 2nd version of the platform uses Soaplab version 2.3.1 with some PANACEA improvements to help with larger files (Section 3.1). The new 2.3.2 was released in 16th of June and will be tested to be used in the 3rd version of the platform.

2.1.2 Biocatalogue

Biocatalogue was deployed and modified to be the PANACEA registry for the first version of the platform. It has proven to be useful and user friendly.

2.1.3 Taverna

Taverna is the workflow editor for the PANACEA platform. For the platform version 2 the used Taverna version is Taverna workbench 2.2.0. On 14th of July the Taverna workbench 2.3.0 was released and it will probably be used in platform version 3 after some tests. This release was expected sooner and this is why it hasn't been used for the actual version of the platform.

2.1.4 myExperiment

MyExperiment is a social network to share workflows and other scientific objects. It has been deployed by ELDA for the 2nd version of the platform for users to share and find workflows and it's presented as the "PANACEA myExperiment portal" (http://myexperiment.elda.org).

2.2 Common interface

As explained in D3.1 and D3.2 the Common Interface (CI) was designed for different kind of tools and documented in several ways to facilitate its use. The CI hasn't been changed for this version of the platform as all the needed definitions were already done.

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.

A new Format has been introduced to be used in some specific situations as TO. The adopted stand-off format is the GrAF standard. Converters, web services and workflows have been developed for some scenarios to work with GrAF.

2.4 Documentation: manuals, guidelines, articles

This section is devoted to list and describe the documentation developed for the 2nd version of the platform.

2.4.1 Common Interface documentation

The Common Interface documentation can be found on the Panacea website (*documents*[6] section of *info for professionals*) and in this deliverable zip file. It consists of four documents:

- **types1.2.xsd**: the types file.

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

- **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**: very detailed document about the CI.
- **Types1.2 Web documentation**: web version of the very detailed documentation.

2.4.2 Travelling Object documentation

The Travelling Object (TO) was thoroughly described in deliverable D3.1. There is no further documentation about it. That TO, based on XCES format, and now called TO1 is still being used in most workflows.

A new format has been introduced and it will be explained in detail section 7 of this deliverable.

2.4.3 Web services documentation

Web services can be documented in many different ways. For Panacea platform version 1 and 2, web service providers must document their web services using the Soaplab metadata file (ACD[7]) and the registration process at the registry. This is the minimum amount of documentation all web service providers must do.

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’s easier to find (the BioCatalogue website has more than 1000 web services), its functionality is better understood and its technical aspects are better described.

Two disclaimers regarding usage conditions were developed at ELDA for the 2nd version of the platform and shared among partners. These disclaimers are being used in the “usage conditions” field which is one of the metadata fields used to describe a web service in the PANACEA registry. Both texts can be found on Section 14.1 of this document.

2.4.4 Workflows documentation

All workflows documentation can be found on the PANACEA myExperiment portal. Each uploaded workflow has its own metadata and graphic representation (low and high resolution pictures). Description field is used to give a general overview of the workflow and tags are used by the search mechanism.

[7] ACD: Ajax Command Definition.

2.4.5 Frequently asked questions

A FAQs list has been updated for fast access to some typical questions, tips and tricks. It can be found in the Panacea website (*FAQs*[8] section of *info for Professionals*). It is based on PANACEA developers experience and some real users' feedback.

2.4.6 Panacea tutorial

The PANACEA tutorial has been updated and now includes different documents. All documents can be found on this deliverable and the last updated are posted on the tutorials page of the PANACEA website (http://panacea-lr.eu/en/tutorials/). Feedback from users was taken into account when developing these tutorials.

The first document is a documentation index. It lists all the relevant documentation for PANACEA. It lists all tutorials and guidelines made by PANACEA partners and other manuals found on the internet that can be helpful for the user.

The second document is the general PANACEA tutorial which is an updated version of the previous one released for the platform version 1.

Afterwards, there are two specific tutorials focused on Soaplab and Taverna for PANACEA version 2.

Soaplab tutorial includes all the relevant information for platform version 1 and now includes all new topics about platform version 2. Soaplab tutorial content is as follows:

- Platform version 1
    - Technical description summary
    - Describing your command line tool: Metadata
    - Deployment and configuration
    - Test your web service: Spinet web client
    - Clients for Soaplab
- Platform version 2
    - Bug: Tomcat 7 and Soaplab 2.3.1 Spinet (Solved)
    - Soaplab output size limit patch
    - Soaplab web services limits

For the second version of the platform, a few **video tutorials** have been prepared and posted on the tutorials page (http://panacea-lr.eu/en/tutorials/). These videos can be very helpful for users because they show the PANACEA platform live.

Videos are recorded in High Definition (HD) and it's recommended to see them in full screen.

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

There is also some specific documentation files (sometime together with software) posted on the PANACEA myExperiment portal:

- "**Limiting web services**"[9] these document and software developed at DCU is used to avoid abuses of users when using web services. All this information can be found on the Soaplab tutorial and will be further described in the Soaplab Section of this document (Section 3.1).

- "**Soaplab (2.3.1) output size limit patch** "[10] is a piece of software and its documentation. It's used to limit the data being transferred inside the SOAP message. It helps improving the network use and forces users to make use of URLs instead of big SOAP messages. UPF developed this patch after some experiments using PANACEA workflows.

- "**Temporary files cleaner**"[11] is a document describing how to handle temporary files deletion with different software and platforms developed at ILC. It also includes a Java program to delete those files.

- "**Naming convention**"[12] is a naming convention proposal developed at ILC aimed to harmonize the names used with web services and workflows in different situations.

2.4.7 Articles, publications, etc.

This is a list of PANACEA articles, papers etc.:

- "**Interoperability and technology fo r a language resources fa ctory**" this article will be presented on the Workshop on Language Resources, Technology and Services in the Sharing Paradigm – November 12, 2011 at IJCNLP 2011 (Chiang Mai, Thailand). The article can be found on this deliverable[13].

3 Web Services

Following section describes the work done for platform v2 regarding web services.

3.1 Soaplab

In this section we are going to describe the work developed with Soaplab for the 2nd version of the platform:

- Tomcat 7 and Soaplab 2.3.1 bug

[9] http://myexperiment.elda.org/files/4

[10] http://myexperiment.elda.org/files/3

[11] http://myexperiment.elda.org/files/1

[12] http://myexperiment.elda.org/files/2

[13] Publications/IJCNLP2011.pdf

- Parameter name bug
- Soaplab messaging improvement
- Soaplab polling
- Limit web service usage
- Temporary files deletion

3.1.1 Tomcat 7 and Soaplab 2.3.1 Spinet bug

When the 2nd version of the platform development started it was a good moment to look for new versions and updates of the software involved. There was a new version of the Apache Tomcat server. After the installation process it was time to verify the compatibility of our existing Sopalab web services. Web services responded normally to the new server but Spinet web client wasn't working at all. We reported the bug to Soaplab developers who soon send us the solution which can be found on the documentation (Soaplab tutorial). Soaplab developers fixed this bug for Soaplab 2.3.2 release.

3.1.2 Parameter name bug

A bug was found on the schema files of web services deployed at DCU due to the parameter names. Aligners' web services following the CI must have these mandatory parameters:

- source_corpus
- source_language
- target_corpus
- taget_language

All of them have the underscore character which is the cause for this bug. The expected schema piece of xml for one of these parameters should be like below:

```
<xs:choice id="source_corpus">
<xs:annotation>
<xs:documentation>help: Tokenised one-sentence-per-line
text.</xs:documentation>
</xs:annotation>
<xs:element name="source_corpus_direct_data" type="xs:string"/>
<xs:element name="source_corpus_url" type="xs:string"/>
</xs:choice>
```

In this correct schema the parameter "source_code" has a "choice" between direct_data and URL.

The following piece of xml shows the malformed schema:

```
<xs:element name="source" minOccurs="0" maxOccurs="unbounded">
    <xs:simpleType>
      <xs:restriction base="xs:token">
<xs:enumeration value="corpus_direct_data" />
<xs:enumeration value="corpus_url" />
<xs:enumeration value="language" />
```

In the malformed schema the parameter name is incorrect and it no longer offers a choice but an enumeration of parameters totally incorrect.

This bug was reported to Soaplab developers who fixed it and send us a patch to solve it in Soaplab version 2.3.1 and they also fixed it for Soaplab 2.3.2 release.

3.1.3 Soaplab output size limit patch

During the numerous test and studies carried out in UPF regarding Soaplab and Taverna it was found that Taverna Soaplab plugin is not using the best option to gather results from web services.

Soaplab operations include methods to gather the output result, only the URL or both. The idea is simple: if you have small data you can gather the data itself and the URL but if you're dealing with big files is much better to use only the URL. Using references instead of the data helps reducing the use of the network, memory and performance of all the software involved.

The problem is that Taverna plugin is using always the Soaplab operation which uses both direct data and the URL. Therefore, the data is being transferred twice. To solve this particular issue UPF has developed a patch for Soaplab to introduce a new parameter. This parameter limits the amount of data that can be transferred inside the SOAP message but it doesn't modifies the Soaplab operations at all. When the data is larger than a limit (specified by the web service provider) the direct data result is changed by a message telling the workflow designer that URL should be used instead.

This technique will help reduce the network usage and improve performance for workflows using large data files with the Taverna Soaplab plugin. During the experiments it was found that Taverna hangs for some workflows with files larger than a few MB if the Soaplab web services don't use this patch.

The patch and its documentation can be found on the PANACEA myExperiment portal (http://myexperiment.elda.org/files/3).

3.1.4 Soaplab polling

All programs calling web services have timeouts to protect the program from hanging waiting for a web service to answer (usually 5 minutes or similar). A web service may fail, or its server could be down or hang, etc. so these timeouts are very useful.

However, when we want to process a big file with a lot of data this will take more time than the timeout limit in most cases. Soaplab has a solution for this problem: the polling technique. The idea is simple: instead of sending a request to the web service and wait for the answer, the program must make periodic requests to the web service to see if it has finished. If these requests are made more often than the timeout the web service will be able to process the data and the client program will know when it finishes.

For the 2nd version of the platform workflows are designed using "polling" and detailed documentation can be found on the "Taverna tutorial". It's presented there because the polling is always used from Taverna and the Taverna Soaplab plugin offers a user friendly approach.

3.1.5 Limit web services usage

With the growing number of web services and users the concurrent calls to the same server may have an important impact on machine resources. DCU has some of the most machine resources consuming web services and has developed a technique to limit web services. Limits will affect concurrent executions and input/output data size.

Concurrent executions limit is based on some scripts used to monitor the amount of web services being executed. If that limit is reached, next requests are queued if the queue is not full; otherwise the request is rejected.

Input data size limit for aligners has been done modifying the scripts used to run the aligners. DCU modified those scripts adding a sentence limit for input data. This limit combined with the patch to limit output size using direct data (3.1.3) offer a safe system for data size limit.

Limiting web service is described in detail in "Limiting web services" technical report where scripts and examples can be found. This document is posted on the PANACEA myExperiment portal (http://myexperiment.elda.org/files/4).

3.1.6 Temporary files deletion

With the growing number of experiments the number of temporary files and results stored on service providers' servers grow very fast. Running a single Soaplab web service generates a minimum of 5 files. A workflow could have several calls to the same server and experiments with more than 1000 calls are made several times a day. This means that our servers must be able to handle more than 100000 temporary files a day.

Each web service provider has the responsibility to deal with these files and it's for their best interest to do it. ILC developed software and a tutorial describing how to handle temporary files even if there is more than one server involved and some files must be kept for more time than others. For example, results can be kept for a few days while temporary files can be deleted in a few hours.

The file with the manual and the software can be found on the PANACEA myExperiment portal (http://myexperiment.elda.org/files/1).

3.2 Deployed web services

The list of deployed web services is presented in this section and some relevant web services for Panacea platform version 2 presented.

Current status of registered web services at the Panacea registry on 19-09-2011. Links and names may change for technical reasons:

Name Type		Category	Provider	Registry Number
grafconverter_skeleton	Soaplab	Format conversion	ws04-iula-upf-edu	143
grafconverter_postagging	Soaplab	Format conversion	ws04-iula-upf-edu	142
signatures2weka	Soaplab		ws04-iula-upf-edu	138

Sed	Soaplab	Corpus Processing	ws04-iula-upf-edu	137
theta_estimation	Soaplab	Lexicon/Terminology Extraction	ws04-iula-upf-edu	136
classify	Soaplab	Lexicon/Terminology Extraction	ws04-iula-upf-edu	135
soaplab_wsdl_validator	Soaplab	Management	ws04-iula-upf-edu	134
xsltproc	Soaplab	Format Conversion	ws04-iula-upf-edu	117
pdftotext	Soaplab	Format Conversion	ws04-iula-upf-edu	116
panacea_conversor	Soaplab	Format Conversion	ws04-iula-upf-edu	115
iconv	Soaplab	Format Conversion	ws04-iula-upf-edu	114
html2text	Soaplab	Format Conversion	ws04-iula-upf-edu	113
catdoc	Soaplab	Format Conversion	ws04-iula-upf-edu	112
vocabulary_analysis	Soaplab	Statistics Analysis	ws04-iula-upf-edu	110
tfidf	Soaplab	Statistics Analysis	ws04-iula-upf-edu	109
ngrams	Soaplab	Statistics Analysis	ws04-iula-upf-edu	108
calcular_p_cue_class	Soaplab	Statistics Analysis	ws04-iula-upf-edu	107
freeling_parsed	Soaplab	Syntactic Tagging	ws04-iula-upf-edu	106
freeling_dependency	Soaplab	Syntactic Tagging	ws04-iula-upf-edu	105
kwic	Soaplab	Text mining	ws04-iula-upf-edu	103
freeling_tokenizer	Soaplab	Tokenization	ws04-iula-upf-edu	101
freeling_tagging	Soaplab	Morphosyntactic Tagging	ws04-iula-upf-edu	99
freeling_morpho	Soaplab	Morphosyntactic Tagging	ws04-iula-upf-edu	98
ilsp_sst	Soaplab	Corpus Processing, Tokenization	nlp-ilsp-gr	131
ilsp_mono_crawl	Soaplab	Crawling, Corpus	nlp-ilsp-gr	130

		Processing		
ilsp_lemmatizer	Soaplab	Stemming/Lemmatization, Corpus Processing	nlp-ilsp-gr	129
ilsp_fbt	Soaplab	Morphosyntactic Tagging, Corpus Processing	nlp-ilsp-gr	128
ilsp_bilingual_crawl	Soaplab	Crawling, Language Guessing, Corpus Processing	nlp-ilsp-gr	127
boilerplate_remover	Soaplab	Corpus Processing	nlp-ilsp-gr	126
sentsplit_tok2to	Soaplab	Format Conversion	www-cngl-ie	95
sentalg_tok_to2word_alg	Soaplab	Format Conversion	www-cngl-ie	94
gma	Soaplab	Alignment	www-cngl-ie	93
bsa	Soaplab	Alignment	www-cngl-ie	92
anymalign	Soaplab	Alignment	www-cngl-ie	91
hunalign	Soaplab	Alignment	www-cngl-ie	80
hello_panacea	Soaplab		www-cngl-ie	79
gizapp	Soaplab	Alignment	www-cngl-ie	78
europarl_tokeniser	Soaplab	Tokenization	www-cngl-ie	77
europarl_sentence_splitter	Soaplab	Tokenization	www-cngl-ie	76
europarl_lowercase	Soaplab	Format Conversion	www-cngl-ie	75
chunk_aligner	Soaplab	Alignment	www-cngl-ie	74
berkeley_tagger2to	Soaplab	Format Conversion	www-cngl-ie	73
berkeley_tagger	Soaplab	Morphological Tagging	www-cngl-ie	72
berkeley_parser	Soaplab	Syntactic Tagging	www-cngl-ie	71
berkeley_aligner	Soaplab	Alignment	www-cngl-ie	70

aligner2to	Soaplab	Format Conversion	www-cngl-ie	69
ExportLMF	Soap	Lexicon/Terminology Extraction	wiki-ilc-cnr-it	21
FC_Freeling_text_2_conll_IT	Soap	Format Conversion	wiki2-ilc-cnr-it	141
TPC_Desr_dependencyparser_it	Soap		wiki2-ilc-cnr-it	140
TPC_Freeling_token_split_POSTagger_it	Soap	Morphological Tagging, Tokenization	wiki2-ilc-cnr-it	139
TPC_Freeling_token_split_POSTagger_en_ca_es_it	Soap	Morphosyntactic Tagging, Syntactic Tagging, Morphological Tagging	wiki2-ilc-cnr-it	90
FC_converter_kaf_to	Soap	Format Conversion	wiki2-ilc-cnr-it	88
FC_converter_fl_to	Soap	Format Conversion	wiki2-ilc-cnr-it	87
iula_paradigma	Soaplab		kurwenal-upf-edu	125
iula_preprocess	Soaplab		kurwenal-upf-edu	124
get_concordances	Soaplab	Querying	kurwenal-upf-edu	123
busca_signatura_in_corpus	Soaplab	Querying	kurwenal-upf-edu	122
apply_re	Soaplab	Querying	kurwenal-upf-edu	121
iula_lexicon_lookup	Soaplab		kurwenal-upf-edu	120
iula_tokenizer	Soaplab		kurwenal-upf-edu	119
iula_tagger	Soaplab		kurwenal-upf-edu	118

3.3 WP3 web services

In this section some of the deployed web services are explained regarding its relevance to the 2nd version of the platform.

3.3.1 Panacea Conversor

This web service has been updated to be able to convert the output of Freeling PoS to the TO1. This will allow the workflow designers to choose between Freeling and the IULA tagger for Spanish.

3.3.2 Grafconverter_skeleton and Grafconverter_postagging

These two web services developed at UPF make use of the grafconverter tool developed at ILC. The first web service is in charge of extracting information from the basic XCES documents crawled by ILSP web services. It creates a few basic files needed to create a GrAF document. These few files together are called the GrAF skeleton. Afterwards, the skeleton is used together with the PoS tagger output to create a complete GrAF document. This final GrAF is the output of the grafconverter_postagging web service.

A workflow has been designed to create GrAF documents with PoS tagging information using basic XCES documents as input. More detailed information about the Grafconverter will be presented in the workflows section 4.3.10 as well as in the GrAF section 7.

3.3.3 Soaplab wsdl validator

Soaplab wsdl validator is a web service designed to check whether other Soaplab web services are PANACEA Common Interface compliant or not.

It is based on a python script developed at UPF which connects to a web service and automatically verifies the mandatory parameters used against the CI parameters proposal. The input parameters to use this web service are the wsdl URL of the web service to be verified and its functionality (PoS tagging, tokenization, alignment, etc.)

It was also developed a workflow to help web service providers with a lot of web services to easily validate all of them with only one workflow execution. The workflow is presented in section 4.3.1.

All partners with web services used the web service or the workflow to validate their web services and the result of that task is presented in a report included in this deliverable[14]. There are only a couple of warnings reported by the validator script: 1) the bilingual crawler due to the fact that it's being validated against the CI of a monolingual crawler and obviously it fails for the two language parameters. 2) A parser web service which is only for Italian and the CI expects a language parameter that is not necessary in this case. To summarize it can be said that the report shows that all partners are following the CI for web services with a CI definition.

3.3.4 fc_freeling_text_2_conll_it

CNR has deployed a converter from the Freeling output to the Tanl format (the input format of the dependency parser for Italian, DeSR). This tool is used when a user needs an Italian corpus parsed at the syntactic dependency level, i.e. wants to execute the Freeling POS tagger and the

[14] In this deliverable: /reports/PANACEA-WP3-t22-CI-validation-results-UPF-ILSP-DCU-ILC-final.pdf

DeSR dependency parser. In Panacea myExperiment, a workflow for Italian has been uploaded chaining the Freeling POS tagger, the converter Freeling To DeSR and the DeSR parser. This workflow allows the user to obtain different levels of linguistic analysis (sentence splitting, tokenization, POS tagging, and dependency parsing). In the current version, the converter works on the proprietary data formats as it was developed before the definition of the TO, and because the conversion is specific to the individual tools due to tagsets idiosyncrasies and no 1:1 mapping.

Both the tagsets and the data formats are different in Freeling and DeSRr. DeSR uses the Tanl tagset and the Conll format. The converter takes in input the Freeling text output (up to the POS tag level) and performs these conversions:

1. It changes the Freeling format in Conll format (one token per line, attributes tab separated);

2.It translates each Freeling pos tag in Tanl pos. Where needed, we have implemented heuristics to solve translation problems. However, some information is lost. (The conversions table can be found at [URL of MyExperiment])

Documentation about the Conll format and the tanl tagset can be found here:

- http://nextens.uvt.nl/depparse-wiki/DataFormat
- http://medialab.di.unipi.it/wiki/Tanl_POS_Tagset

Work is ongoing for the development of a version of the converted through TO2 (GrAF), and for the development of the converter from the DESR output to TO2.

3.4 WP4 Web Services

All WP4 web services deployed and ready to be used in the platform are reported on D4.4 appendix E.

3.5 WP5 Web Services

WP5 aligners web services are listed in this section as well as their registry entries.

Sentential Alignment:

3.5.1 Hunalign

This web service was presented on section 2.1.1 of D5.2.

Its Registry entry is http://registry.elda.org/services/80

3.5.2 GMA

This web service was presented on section 2.1.2 of D5.2.

Its Registry entry is http://registry.elda.org/services/93

3.5.3 BSA

This web service was presented on section 2.1.3 of D5.2.

Its Registry entry is http://registry.elda.org/services/92

Sub-sentential alignment:

Giza++

This web service was presented on section 2.2.1 of D5.2.

Its Registry entry is http://registry.elda.org/services/78

Berkeley Aligner

This web service was presented on section 2.2.2 of D5.2.

Its Registry entry is http://registry.elda.org/services/70

OpenMaTrEx chunk aligner

This web service was presented on section 2.2.3 of D5.2.

Its Registry entry is http://registry.elda.org/services/74

Anymalign

This web service was presented on section 2.2.4 of D5.2.

Its Registry entry is http://registry.elda.org/services/91

3.6 The registry: sharing web services

The PANACEA registry of web services has been slightly modified. In particular, changes concern the layout of the registry. Indeed, the update of the last BioCatalogue version modified the interface layout in the list of available services, presenting now more web services on a single page than the previous version. For each web service, a summary is showed by default in a box, including its name, its type (Soap, Soaplab), its status (passed, warning, failed), its description and its provider. Simple view and detailed views of web services are also possible.

But the main modification is related to the addition of a link so as to access directly to the Spinet form. Thus, it allows user a faster access to the web service in the list of Soaplab web services. The Spinet link is only available for Soaplab web services.

Furthermore, the status monitoring frequency has been decreased to one check per day. This is mainly due to 1) reduce the server load and 2) the lower activity on the registry, since PANACEA partners have already registered their web services and then less services are registered per day.

Finally, a few minor bugs have been corrected. The main is related to the WSDLUtils (the service used by BioCatalogue to parse WSDL files into the BioCatalogue format) tool, which has been updated, thanks to Biocatalogue developers. The errors fixed were preventing to add some non Soaplab web services into the registry.

4 Workflows

4.1 MyExperiment: sharing workflows

4.1.1 Deployment

From the tools options to share research object, defined in D3.1, myExperiment[15] is the one PANACEA is currently using to provide a workflow catalogue. Following the first review report advice the PANACEA myExperiment portal can be used to easily find example workflows.

Likewise the registry, a PANACEA version of myExperiment has been installed on a local server at ELDA so as to share workflows combining different web services. We then conducted our own modifications and design and the PANACEA catalogue of workflows is independent from any other project and may follow its own development. The PANACEA myExperiment is available at http://myexperiment.elda.org. Workflows placed in the catalogue may then be available to other partners and reused as wished.

The installation and setup of myExperiment have been rather easy and we benefited from the experience of the registry. Indeed, the two applications are very close and developed by the same groups. The PANACEA myExperiment is considered as stable since no daemons have been restarted for several weeks, and workflows have been submitted in the meantime. Until now (t20), 19 workflows have been submitted from DCU, ELDA, ILC, ILSP and UPF.

After being registered, providers can submit a new workflow by uploading a Taverna file. Workflow providers can also add extra information such as tags, credits, sharing options, license, description, etc.

Users can browse the catalogue of workflows and search for workflows using key words, but also according to the type, tag, licence or workflow provider. A “runner” allows users to execute workflows. Unfortunately, for the time being, it only works on Taverna 1.0 workflows. Developers of myExperiment are working on a new version of the application which will be able to run Taverna 2.0+ workflows.

Unlike the registry, there is no monitoring system for workflows: web services used in workflows are checked within the registry.

In more details, the following functionality is available:

- List the available workflows;
- List the registered users;
- List user groups;
- List files;
- List packs (collections of different items listed above);
- Search in the different lists;
- Add and remove items of the different lists;
- Register as a user of the PANACEA myExperiment.

[15] http://www.myexperiment.org

4.1.2 Modifications

Likewise BioCatalogue, no specific documentation is available about its features for developers. However, PANACEA myExperiment was easier to deploy. Thus, configuration settings are similar to the registry and little time has been needed to install. For debugging or help, a “myexperiment-discuss” list is available for myExperiment users[16].

However, further database corrections have been made so as to fix some bugs in the catalogue of workflows. This was mainly due to compatibility difference of system or tool versions.

Main modifications are related to the design of the interface so as to adapt the layout (e.g. colors) to the PANACEA web site, the logos and the main page information.

4.1.3 Shared workflows

The list of workflows is presented in this section in the current version of the PANACEA MyExperiment (t20). At the same stage, six users are registered.

Name Type		**Provider**	**MyExperiment Number**
bilingual word aligner for crawled data	Taverna 2	UPF	9
Panacea Common Interface validation for Soaplab web services	Taverna 2	UPF	25
ILSP Basic NLP Tools	Taverna 2	ILSP	20
bilingual crawler output language splitter	Taverna 2	UPF	4
bilingual sentence alignment for crawled data	Taverna 2	UPF	7
WORD freeling tagging and stylesheet	Taverna 2	UPF	23
GrAF PoS tagging with Freeling for basicxces documents	Taverna 2	UPF	26
PDF sentence alignment	Taverna 2	UPF	21
Freeling tagging for crawled data	Taverna 2	UPF	5
bilingual sentence alignment for crawled data EN EL	Taverna 2	UPF	8
Freeling to Desr - From text cleaned to text parsed (with Tokenizer and Tagger Freeling, Dependency Parser Desr)	Taverna 2	ILC	24

[16] http://lists.nongnu.org/archive/html/myexperiment-discuss/

Merge list of errors to string	Taverna 2	UPF	27
List example 01	Taverna 2	UPF	3
PDF freeling tagging with panacea stylesheet	Taverna 2	UPF	22
IULA tagging for crawled data	Taverna 2	UPF	6
MEDAR test workflow	Taverna 2	ELDA	1
Wordalignment using GIZA++	Taverna 2	DCU	16
test01	Taverna 1	ELDA	2
bilingual sentence alignment (using GMA) for crawled data	Taverna 2	DCU	19

4.2 Taverna

In this section all the relevant topics about Taverna are presented. Taverna is the workflow manger for the 1st and 2nd version of the PANACEA platform. The needs of the project (large data, many input files, and long lasting processes) make it necessary to make use of all the functionalities that Taverna provides. These advanced features of Taverna will be used to design robust workflows for the 2nd version of the platform. These are the presented topics:

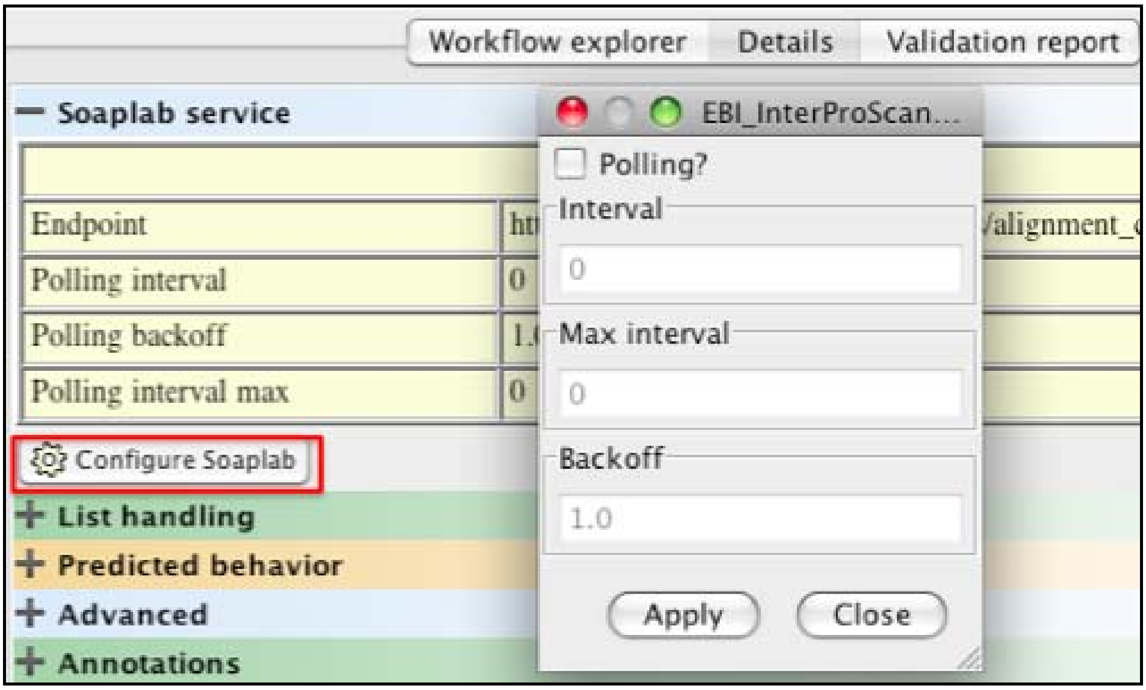
- Polling
- Retries
- Parallelization
- Taverna Server
- Taverna on Windows

4.2.1 Polling

As mentioned before, "polling" is a very interesting feature of Soaplab web services that allow the execution of long lasting processes without reaching the client's timeout. If Taverna calls a web service and it doesn't answer in before this timeout the call is cancelled and an error is reported. Thanks to Soaplab polling we will be able to skip this timeout by making periodic requests to the web service to check its status.

Workflow designers can avoid the timeout by creating a series of calls to the soaplab operation "getStatus" until the web service is finished and then use operation "getResults". This would perfectly work but it would create a much more complex workflow than what users in PANACEA are used to (compared to 1st version of the platform workfows).

On the other hand, if designers make use of the Taverna Soaplab plugin (it was also used in the 1st version of the platform) they'll be able to easily configure polling without making complex series of calls. The plugin will make them automatically. This is one of the reasons why it was important to use the plugin.



Endpoint	ht
Polling interval	0
Polling backoff	1.(
Polling interval max	0

□ Polling?

Interval: 0

Max interval: 0

Backoff: 1.0

Figure 1: Polling parameters with Taverna Soaplab plugin

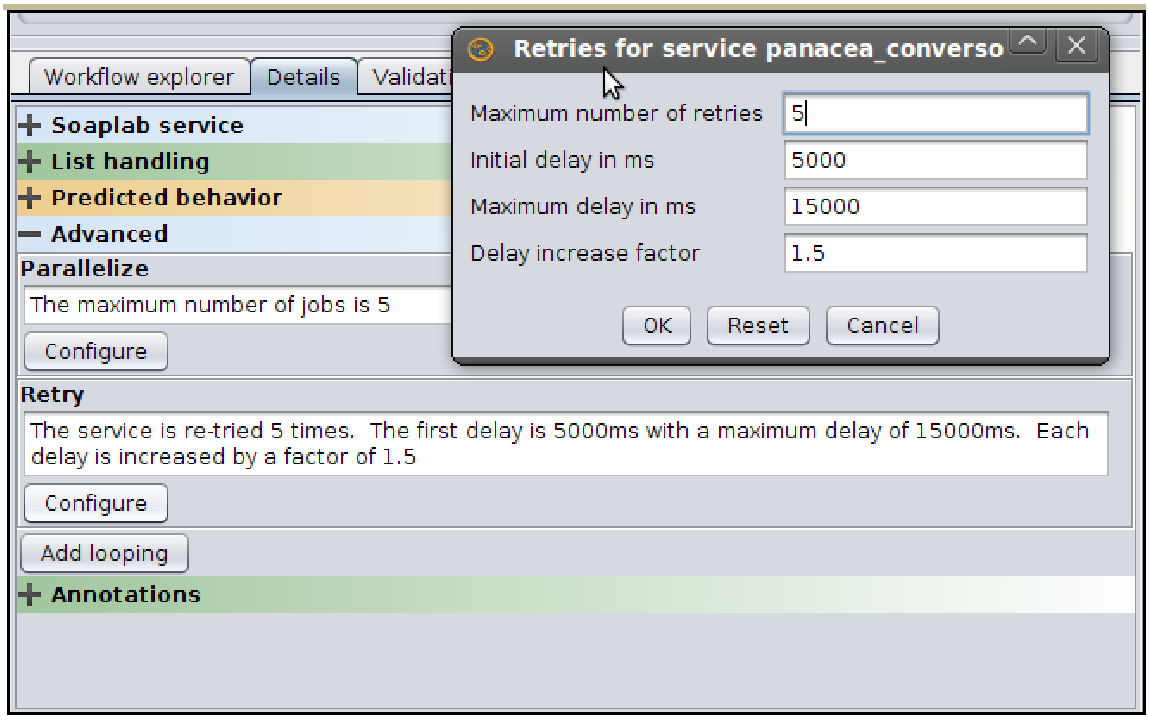
Figure 1 shows how to configure the polling in a Soaplab web service. “Interval” sets the initial time between requests, the “backoff” parameter specifies how “Interval” is increased every time and “Max interval” is the Max interval used between requests.

All this information can be found on the Taverna tutorial.

4.2.2 Retries

When a workflow has a few input files (it has a few iterations) if something goes wrong or one of this files fails at some point of the workflow there is always the option of running the workflow again.

However when there are a lot of iterations running the workflow again is a waste of time and resources. Taverna implements an automatic retry system that allows the designer to configure every web service call in a workflow.



Parallelize

The maximum number of jobs is 5

Retry

The service is re-tried 5 times. The first delay is 5000ms with a maximum delay of 15000ms. Each delay is increased by a factor of 1.5

Maximum number of retries: 5

Initial delay in ms: 5000

Maximum delay in ms: 15000

Delay increase factor: 1.5

Figure 2: retry parameters

In Figure 2 it can be seen how to configure retries for a web service and the parameters involved.

The Taverna tutorial has a link to a myGrid video which is very descriptive and helpful to understand how to use the “retry system” in Taverna.

4.2.3 Parallelization

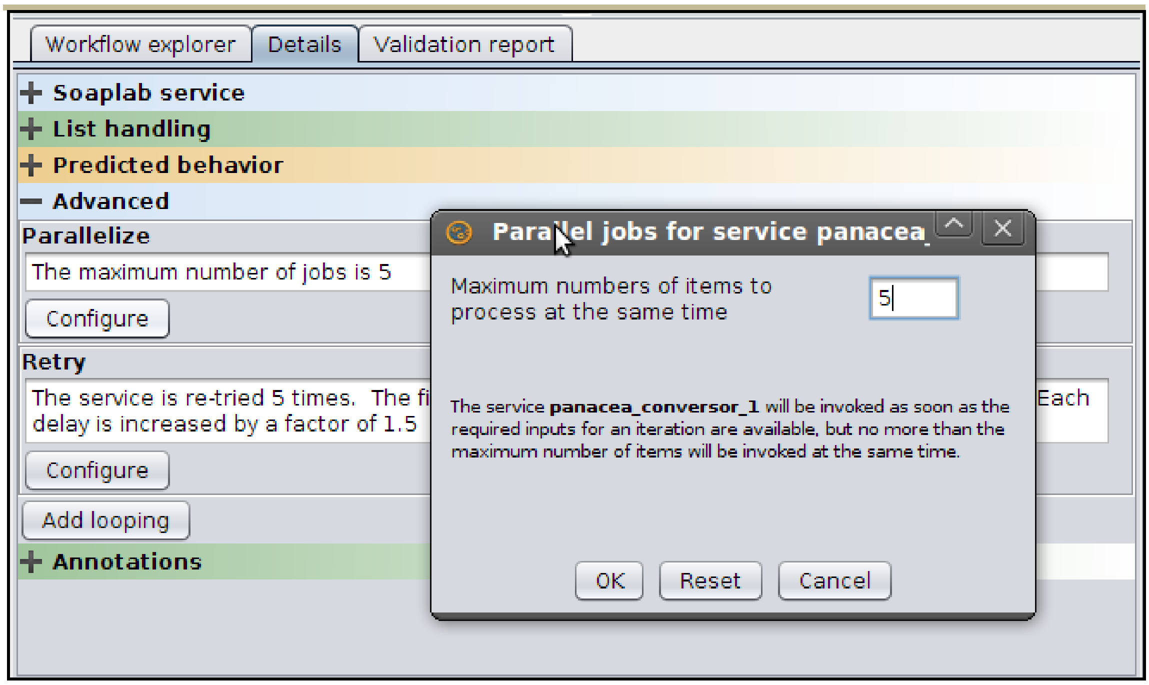
Taverna offers to possibility to make multiple calls to the same service in one workflow. This parallelization makes workflows with multiple iterations to finish earlier. The first simple test, carried out in UPF about parallelization demonstrated that simply doubling (x2) one web service in a workflow with only that service reduced the execution time in half.

Parallelization seems to be a great advantage but it has its drawbacks. Web services are run on machines with limited resources (processors, memory, etc.) which cannot handle infinite parallel calls to their web services. One problem is that most of those limits can only be measured empirically. Some web service providers offer information about the limits of their web services on the Registry.

A bad use of parallelization may cause the server to fail or to be very slow which is the opposite of the desired behavior.

Figure 3 shows how to use the parallelization parameter for a web service in Taverna.

The documentation about Parallelization can be found on the Taverna tutorial.



The maximum number of jobs is 5

The service is re-tried 5 times. The fi … Each
delay is increased by a factor of 1.5

Maximum numbers of items to process at the same time: 5

The service **panacea_conversor_1** will be invoked as soon as the required inputs for an iteration are available, but no more than the maximum number of items will be invoked at the same time.

Figure 3: Parallelization parameter

4.2.4 Taverna Server

The plan for PANACEA platform 2 was to have a Taverna Server where long lasting workflows could be executed and results would be obtained later. Having Taverna on a server would allow it to have a better internet connection and more resources (memory, faster hard drive, etc) than a personal computer. It would also allow users to shutdown their computers while the workflow is being executed on the server.

Taverna 2.2 server was tested but it didn’t fulfil PANACEA requirements of usability and security. It required a lot of development to make it ready for users. It was decided to wait for the Taverna Server 2.3 due to May 2011. Taverna release was delayed and now, according to the Taverna roadmap, Taverna server was supposed to be released on August 2011 leaving no time to include it in the platform version 2.

We expect Taverna Server 2.3 to be released soon and it would be tested to be used in the 3rd version of the platform.

4.2.5 Taverna on Windows

During the tests done on Linguatec using Taverna on Windows operating system it was found that Input parameters and read/write file capabilities had encoding problems. Taverna used the default encoding used by the operating system and broke some characters. These bugs were presented on D3.2 Section 5.4 and Linguatec developed an easy to use solution for read/write operations in Taverna 2.3.2. These bugs were reported to Taverna developers who fixed them for the new Taverna 2.3.0.

4.3 Workflows

Several relevant workflows are listed in this section regarding the 2nd version of the platform. The rest of the workflows are presented can be found on the PANACEA myExperiment portal.

4.3.1 Panacea Common Interface validation for Soaplab web services

This workflow, shown in Figure 4 in 14.3 "Workflow images", is a simple workflow designed to use the "Soplab wsdl validator" presented on Section 3.3.3. It allows Web Service Providers to validate the CI of all their web services in one workflow execution. The workflow and its documentation can be found on: http://myexperiment.elda.org/workflows/25.

4.3.2 Merge list of errors to string

This workflow, presented in Figure 5 in 14.3 "Workflow images", presents three different ways to show a list of results and its advantages and problems. The output "list_with_errors" is the tipical output in a workflow, it directly shows the output port of a processor. It presents a list of elements and if there is an error that element is marked in red. There is not an easy way to copy all results using this output but it contains all the information and never fails. The output "Merged_output_fails" is a single merged output where all results are presented in a single output. It's really user-friendly because the results list can be copied with a simple CTRL+C. Its drawback is that a single error on one element of the list makes this output fail. This is why it's always used with the direct to a port output. The final option is the "merged output without errors". This output stores every correct result in a temporary file and presents the list at the end. This option avoids the errors but it only provides the correct outputs.

This new way to provide the output was designed at UPF in collaboration of mygrid support team. The example workflow can be found in: http://myexperiment.elda.org/workflows/27?version=2.

4.3.3 WORD and PDF freeling tagging and stylesheet

These are two workflows designed to get the PoS tagging of Word and PDF documents in the PANACEA TO1 format. Using converters deployed at UPF, Word and PDF documents are converted to plain text and afterwards processed with Freeling.

The workflow to process Word documents is presented in Figure 6 of 14.3 "Workflow images and can be found on: http://myexperiment.elda.org/workflows/23. On the other hand, the workflow to process PDF documents is posted on: http://myexperiment.elda.org/workflows/22.

4.3.4 Freeling tagging for crawled data

This workflow, presented in Figure 7 in 14.3 "Workflow images", can process documents downloaded using the ILSP crawler in the Basic XCES format, process them with Freeling PoS tagger and present the results in the TO1 format.

The workflow is posted on http://myexperiment.elda.org/workflows/5.

4.3.5 Freeling to Desr - From text cleaned to text parsed

This workflow, developed at ILC and shown in Figure 8 in 14.3 "Workflow images", combines the PoS tagger of Freeling with a dependency parser.

It's documented and posted in http://myexperiment.elda.org/workflows/24

4.3.6 Bilingual sentence alignment for crawled data

This workflow, presented in Figure 9 of section 14.3 "Workflow images", can process crawled data in the Basic XCES format. Using the "europarl tools" and "hunalign", it can make the sentence alignments and present them using the TO1.

This workflow can be found on http://myexperiment.elda.org/workflows/7

4.3.7 Bilingual sentence alignment for crawled data EN EL

The following workflow, shown in Figure 11 of section 14.3 “Workflow images”, uses the ILSP web service to process Greek and allows using this language for sentence alignment (The previous workflow cannot process Greek).

This workflow is posted on http://myexperiment.elda.org/workflows/8

4.3.8 Bilingual sentence alignment (using GMA) for crawled data

This workflow is like the “Bilingual sentence alignment for crawled data” (previously presented on section 4.3.6) but instead of using “hunalign” it uses “GMA”.

This workflow is posted on http://myexperiment.elda.org/workflows/19

4.3.9 Bilingual word aligner for crawled data

This workflow, presented in Figure 11 of section 14.3 “Workflow images”, uses “hunalign” and “Giza ++” to make the word alignment for Basic XCES documents. The output is presented using the TO1 format.

This workflow can be found on http://myexperiment.elda.org/workflows/9

4.3.10 GrAF PoS tagging with Freeling for basicxces documents

This workflow, presented in Figure 12 of section 14.3 “Workflow images”, is an example to show how to use the two new web services designed to create GrAF documents. The input data for the workflow is Basic XCES (TO1) data crawled from the ILSP crawler. The data is processed using the PoS tagger of Freeling and finally a GrAF is presented (as a set of output documents).

This workflows is posted on http://myexperiment.elda.org/workflows/26

5 Complementary tools

5.1 GIT Server

Since PANACEA implies several developments (especially in WP3, WP4 and WP5), a source version control system has been deployed. In particular, it helps developers to work on application development, or resource building, in keeping track of successive modifications and go back on previous versions. In our case, it also aims at storing tools and resources of the project.

Several options exist like SVN, CVS or GIT for the most well-known. Contrary to SVN and CVS, GIT is a distributed system (against centralised) and allows the management of branches for specific tasks. It also seems the GIT performance is higher and it is the backbone technology used on the famous github[17] portal. Therefore, in PANACEA, we choose to use GIT as our source version control system.

[17] https://github.com/

During the last few months and since t14, GIT is used by PANACEA partners. For the time being, WP3, WP4 and WP5 have been the contributors. The activity can be summarized with 53 commits from 5 different users. Main activities have been observed during t14, t15 and t18.

To use PANACEA GIT, users have to install GIT, generate a public SSH key associated to a password and send the public SSH key to ELDA. Next, access to PANACEA GIT is then manually granted and users may pull and commit data.

6 Massive data

Handling massive data is one of the most challenging issues for the PANACEA platform. There are many variables involved in the performance result and some of them cannot even be controlled (like internet performance). The article "So you want high performance"[18] by Peter Lin show that achieving high results with internet web sites or web services is a complex task that requires a hard work in all aspects involved.

In this section we will present a list of variables involved in the overall platform performance and the first work done at the beginning of the second development phase about performance. The first report about data tests is presented as well as the second. Conclusions and future work about handling massive data can be found at the end of this section.

6.1 Involved variables

There are many aspects involved in the performance of a distributed software platform as PANACEA. Some of them have just been noticed by PANACEA developers while doing the tests, some are incontrollable, etc.

- **Internet**: local networks, WAN networks, Internet provider, etc. The network has a very big impact on the performance depending on the experiment. The more input data files (data is divided in many small files) the more network traffic generated. Thus, the performance becomes more dependent on the network.

- **The tools** : the tools being deployed as web services obviously have an impact on performance. Some of the tools consume a few resources and others lots of them. Some of them may have memory leaks[19] or non-optimal processes ending up with a lot of wasted machine resources.

- **The machine**: The hardware involved has an obvious impact on performance: number of CPUs, type of CPU's, memory (amount and speed), hard drive (amount and speed), etc.

- **The Operating System**: the operating system controls the different running processes, the read/write operations, etc. It also has an impact on the software versions that can be used.

[18] "So you want high performance" article can be found in this deliverable: references/performance.pdf

[19] Memory leaks: is a reserved portion of memory which is no longer used by the tool but is still reserved by error. No other applications can use it.

- **Web services variables**: there're many variables under this section. The application server (e.g. Tomcat), its variables, version, etc. The web service application (e.g. Soaplab) its configuration parameters, timeouts, temporary files management, memory usage, data transfer protocol, etc.

6.2 Handling massive data

After the first development phase it was time to benefit from some advantages within Taverna and Soaplab to help handle massive data and to begin working in that direction.

In this section, the first's movements towards handling massive data are described.

6.2.1 Using Taverna features

The first move about handling massive data was to make the already designed workflows more robust and able to handle more files and bigger files.

Most crawled data by the ILSP crawler, is presented in small files (not more than 100 kb). For this amount of data and for most of the deployed tools there are no timeout problems. But for larger files (1 mb) the 5 minutes timeout in Taverna could be a problem. To avoid this timeout the Soaplab "polling" technique is a great advantage. "Polling" (introduced in section 4.2.1) allows the execution of long lasting processes without reaching the Taverna timeout. However, this solution has its performance issues: polling implies periodically asking if the task is finished. If the time between request its two high the system can be waiting a too much for an already finished task. On the other hand, making requests too often can overload the server and the network.

To make workflows more robust all workflows were modified to add "retries" to all web services calls. Retries (introduced in section 4.2.2) allow avoiding some network errors or servers problems by calling a service again when it doesn't respond correctly.

To make executions faster, most workflows were modified to parallelize the calls to web services. Parallelization (introduced in section 4.2.3) improves performance but if abused can have a bad impact on performance (machines have limited resources). The problem is that its optimal use can only be measured empirically.

6.2.2 Soaplab

During some tests it was found that the "Soaplab Taverna plugin" does not use the optimal operation to gather results from the web services. It gathers the data and the URL reference to that data. If the data result is not very small it's always better to use only the URL because then the amount of data being transferred is minimal. Soaplab was modified to improve this situation and a patch (introduced in section 3.1.3) was shared among PANACEA service providers. The output size limit patch makes Soaplab send the URL and a warning message instead of the output data (the warning message is much smaller than the output data, improving the network usage efficiency).

6.2.3 Temporary files management

It was easy to predict that with more experiments and more files to be processed temporary files could become a problem. In section 3.1.6 it was introduced different options to periodically delete temporary files.

6.3 First tests report

The first experiments were designed to show and learn how to use some of the used options to handle massive data. The report[20], showed the impact of using "polling", "retries" and "parallelization".

Polling solves the timeout issue (test 1-A and 2-A) for larger files as expected and used with the adequate requests interval can be very useful.

Parallelization does not always mean a faster execution, when the maximum capacity of the machine is reached the parallelization is useless or it can even overload the server. It can also be seen that a better server can better deal with numerous executions and parallelization.

The tests also show a considerable variance between the same experiment execution time. It was seen that most execution were faster early in the morning due to de fact that the networks are less used and nobody else was running any process on the servers.

It was also important to notice that Taverna workbench required more memory in order to perform better and to be more robust against failures.

6.4 Second tests report

6.4.1 Scenario 1

This September it was time to run more tests. This time focused on testing only server capabilities and dealing with more files than on April's tests.

- Taverna is executed from a PC in UPF behind a proxy.
- The tested machine with deployed web services is **iula04v.upf.edu:**
    - Linux based 64 bits virtual machine (deployed on a machine called Isolde)
    - 1 CPU and 4GB of RAM.
    - Tomcat 6.0.29 and Soaplab 2.3.1 (with the PANACEA patch)

One workflow was chosen to make the tests: "Freeling tagging for crawled data" presented in section 4.3.4. This workflow is executed completely and solely in iula04v and it gathers the data from another server (originally ILSP data server).

The workflow has 3 processes, all executed on the iula04server:

1. **Panacea Converter**: from basicXces to plain text. Parallelization x5
2. **Freeling tagging**: PoS tagging. Parallelization x5
3. **Panacea converter**: from PoS to Travelling Object 1 (XCES). Parallelization x5

[20] The report can be found on reports/PANACEA-WP3-t22-Massive_data_tests-v02-deliverable.pdf

All three processes are executed 5 times in parallel which represents a total of 15 simultaneous processes per input file. This represents a big load for a virtual machine with only one processor assigned.

Surprisingly experiments reported more errors and were slower than the ones from April (test 7 of April report). For a few files it was only slower but for more files it didn't finish or reported errors.

There have been deployed more virtual machines on Isolde since April and this could be the reason for the web services being slow. Although virtual machines have reserved memory and CPU the hard drive access is the same for all the virtual machines.

It was also found with some tests that gathering data from ILSP was dramatically slow during a few periods of time causing unsustainable delays for the system.

The busy server and the slow network can explain a worse overall performance but Taverna not finishing the execution had to be something else. Taverna developers and support team were contacted and the problem was reported. In Figure 13: Never finishing execution (section 14.5) it can be seen the execution of 661 files, the "running" message and some iterations with the "waiting for data message". This last message usually appears when one iteration is being executed and the result of one web service has not been reported to Taverna yet. Unfortunately, in this scenario the answer from the web service never gets to Taverna.

Several tests were done and some showed this problem. All relevant logs (Taverna and Soaplab) showed no warning or error message on those concrete iterations with the "waiting for data" message. Taverna should be able to detect these situations but it does not and the worst problem is that results cannot be saved after the execution is cancelled. PANACEA is still working in collaboration with Taverna developers to find the cause of this bug.

On the other hand, other problems were detected. Some executions suffered bursts of errors, a big number of iterations with errors. Figure 14: errors burst (section 14.5) shows a series of errors in red in a correctly finished workflow execution. The study of the errors showed different causes: in some cases it was due to the network problems with ILSP servers and others showed that the machine or Tomcat run out of memory. The burst was due to the time it took Tomcat to recover. This is probably showing a memory leak problem since the use of memory should not rice because the programs are being run more times. A deep analysis of the memory usage done by Tomcat, Soaplab and the tools involved (Freeling and the converter) should be done. This is a complex task that takes a lot of time but it should be considered for the 3rd version of the platform.

6.4.2 Scenario 2

It was time to improve the virtual machine. More resources were assigned to iula04v and some updates were done:

- 2 CPUs instead of only one. However it must be taken into account that the host machine Isolde has only 8 CPUs for its internal use and 5 running virtual machines. This means that the CPUs may have to be shared anyway.
- 6 GB RAM instead of just 4.

- New Tomcat version 7.0.8

These improvements reduced the error bursts as expected but they didn't improved execution time and didn't solved the problem of unfinished executions due to the "waiting for data" iterations.

After a few research it was found that using the Apache Tomcat Native Library could improve performance:

Test results:

- **Tomcat 7.0.8 + libtcnative-1 1.1. 19** (lucid repositories). The library libtcnative-1 1.1.19 was downloaded from the lucid repositories (lucid is the Ubuntu version on the virtual machine). This combination was found very unstable. The server hung easily even with simple tests with Spinet web client.

- **Tomcat 7.0.21 + libtcnative-1 1.1.1 9** (lucid repositories). A newer version of Tomcat was installed to see if it worked better in combination with the Native Library. It kept being unstable but correct executions were much faster than before (without the native library). Converter executions, which used to vary from 10 to 20 seconds became more stable around 6 seconds. The changes log on the Tomcat website reported that version 1.1.19 could hung the Java virtual machine so it was decided to try find antoher version.

- **Tomcat 7.0.21 + libtcnative-1 1.1. 20-1** (maverick repositories). This is the actual combination on the iula04v machine. To install the new version of the Native Library the system was connected to another version of Ubuntu (maverick) for which the new library was existent. The system is much more stable now but it hangs sometimes. Performance has improved reducing execution time.

The repositories must be check periodically for new updates and more stable versions and probably some test should be done compiling the Native library directly on the iula04v machine instead of using a software package.

6.4.3 Scenario 3

With the new Tomcat and Native library installed and the considerably improved performance it was time to run the experiments with more data. However, it must be remembered that the non-finishing workflows is still an issue. However, Taverna developers run the experiment several times without being able to reproduce the "waiting for data" messages.

This could mean there's a problem in the local Taverna being used to run workflows in the PC at UPF. Taverna support team recommended using Taverna 2.3.0 and also try using the command line instead the graphical interface. One very important advantage of the command line tool to run workflows is that results are written in a directory during the execution so there's no need to wait until the end. This way, the problem with saving results in the graphical workbench is avoided.

New experiments with 2k, 3k, 5k and 10k documents (1k files is around 1M words) showed that Taverna must be used without the parameter "in-memory" activated for such large amount of data or it will consume all PC memory resources.

The experiments with 5k documents took between 110 and 200 minutes to be run. Unfortunately, it was always necessary to cancel execution for the workflow to finish. The "waiting for data message" appears on a few executions (1/10) leaving Taverna without knowing when the workflow ends and waiting for results that never arrive. We expect to solve this problem in collaboration with Taverna support team.

Experiments with 10k documents also suffered from the non-finishing execution problem. They took between 5 and 7 hours and were useful to find another situation that need to be solved. Some Linux file systems can only have 32k folders in one directory. Soaplab server stores all temporary data files in the same folder. If a workflow with three Soaplab web services deployed on the same server is run with 10k input files there are going to be needed 30k folders. If there were already some temporary files there the limit is really easy to be reached causing the systematic failure of all new executions.

To solve this 32k limit Soaplab developers have been contacted to improve the temporary files management. If there is not a solution soon a script to automatically erase temporary files when the 32k limit is near will be developed and deployed.

6.4.4 Scenario 3: DCU

The 5k experiment has been run (several times) from a Taverna installed on a DCU PC confirming that non-finishing workflow executions problem was a UPF PC problem. This means that now PANACEA can handle the execution of 5k documents (aprox. 5M) all on the same server (iula04.upf.edu) with a very demanding workflow with more than 15 simultaneous processes.

6.5 Conclusions and future work

From the experiments it can be concluded that achieving high performance and handling massive data is a very complex task. Developers are sometimes limited by the software tools being used and its versions, the machine resources, etc.

Handling massive data requires hard work on all the aspects involved. Assigning enough machine resources it's not always easy due to the expensive prices of powerful servers. However, it shows that with the sufficient economic resources more powerful machines can be bought or virtual machines can be rent to achieve higher performances.

Another aspect that needs to be taken care of is the system maintenance. Updated versions of the operating system and all the software involved can benefit performance. Compatibility issues may arise and can reduce performance and force unpredicted errors. Installing different versions and make several tests with a good report system can be very helpful and should be done from now on.

As expressed on the article, "So you want high performance", using XML for data transport (the SOAP messaging system) and for data storage (the TO) has a price paid in lower performance. Tomcat, Soaplab and Taverna need to be carefully parameterized for optimal results.

Network and users behaviour will always be unpredictable and tools should be robust to their actions and changes.

For the future improvement of massive data handling in PANACEA it's clearly important to go on with the collaboration with Taverna and Soaplab support teams which has been fruitful so far.

Tomcat and Soaplab should be tested in detail for memory leaks, performance, improved logging system and temporary files management. In case of errors or bugs they should be reported to their respective support teams and solutions should be designed. E.g. the system to delete temporary files when near the 32k limit.

Also the deployed tools should be checked and studied for memory leaks: Particularly, converters which are massively used and consume more resources than expected.

Reducing parallelization should also be tested. Taverna support team reported that is a very stressful use for the services and for Taverna itself. Test with less parallelization or none at all should be done and results compared. The last tests are being done with 15 concurrent processes which is a lot for the iula04v virtual machine.

Further experiments will be done to test with 10k documents and the problem with Taverna deployed at UPF PC will be addressed. Running the experiments from several locations will be tested and more complex workflows with distributed tasks will tested as well.

In the end, it seems that massive data requirement will be fulfilled but it will require more effort and hard work for service providers if we want to achieve more than 10k documents per experiment.

7 GRAF

In this section the discussion about the new TO is presented. All partners presented ideas and contributions which end up in a few proposals and options. Finally, it was decided to choose the GrAF standard because some if its advantages.

Many tools outputs are not ready to be presented in a stand-off format. In this section we're also going to present the modifications needed to make tool outputs ready to be converted to a stand-off format.

The GrAF section is presented with these topics:

- Stand-off analysis
- The GrAF format
- Preparing tools: output modification
- GrAF Converter
- Travelling object document path

7.1 Stand-off analysis

Different options have been analysed to adopt one stand-off format to be used as TO. Improving the original TO XCES was discarded because the aim is to adopt an already existing format (as standard and widely used as possible). This format needed to use stand-off annotation and be as flexible as possible due to the multiple in-house formats used by the tools.

ILC developed a study of the KAF[21] format and the feasibility of using such format in PANACEA. The study[22] was very useful and brought much information and ideas to the discussion. The report pointed out that KAF needed some adaptation before it could be used for parallel corpora and it does not fulfil the complete flexibility requirement.

The Graph Annotation Format (Ide and Su-dermam, 2007) is the XML serialization of LAF (ISO 24612, 2009). It can be seen as a flexible container for much different kind of annotations making it a good option to adapt the numerous in-house formats used by the tools in PANACEA. GrAF only specifies how to make annotations but not their "names" or their content. It provides a logical framework rather than semantic.

Another advantage of GrAF is that it uses CES headers and that it's compatible with cesAlign documents already being used in TO1. Moreover, GrAF is being used by the American National Corpus (ANC) and converters to use GrAF in UIMA and GATE can be found on their website.

7.2 The GrAF format

Using the GrAF schema files and examples found on the Mini-MASC[23] corpus of the ANC a GrAF model was build to be used as an example for PANACEA converter developers. This example can be found in a ZIP file on this deliverable[24].

A document represented in GrAF is divided in different files:

MAIN FILES:

- **<doc-name>.anc**: it's an XCES header with provenance and general information. It also has information about the location of the primary text file and all associated annotation files.
- **<doc-name>-original.txt**: It contains the original document.
- **<doc-name>-plain.txt**: It contains only the text without HTML markup, etc.
- **<doc-name>-layout.xml**: It contains the HTML markup.

ANNOTATION FILES (A few examples since there can be various)**:**

- **<doc-name>-seg.xml**: It contains basic segmentation of the original document like paragraphs, etc.

[21] KYOTO annotation framework

[22] ILC Travelling Object 2 proposal / study (PANACEA-WP3-t22-ILC-TO2-aboutKAF.pdf on this deliverable)

[23] http://americannationalcorpus.org/MASC/Download.html

[24] TO/GrAF/graf-example.zip

- **<doc-name>-s.xml**: sentence boundaries region annotations.
- **<doc-name>-pos.xml**: PoS annotations
- **<doc-name>-chunk.xml**: Chunker annotations
- **<doc-name>-dep.xml**: Dependency parser annotations

7.3 Preparing tools: output modification

Not all tools outputs can be represented in stand-off annotations. Tools which introduce changes to the text segmentation must provide a reference between the new and the original in order to create correct stand-off annotations. To this aim, tools like Freeling and the IULA tagger have been modified making their outputs ready to stand-off annotations.

UPF developers modified the Freeling source code so for every token there's a reference to the characters in the original text. If the following text is processed:

```
cómetelos
```

With the original Freeling PoS tagging the result is:

```
come      comer        VMM02S0      1
te        te           PP2CS000     1
los       los          PP3MPA00     1
```

It can be seen that the segmentation of the original word "cómetelos" is different to "come te los". Thanks to the modified output with character reference to the original text the new segmentation can be referenced to the original text.

```
come      comer        VMM02S0      1      0      4
te        te           PP2CS000     1      4      6
los       los          PP3MPA00     1      6      9
```

With the tools output modified and ready to be used to create stand-off annotation it was time to design converters.

7.4 GrAF Converter

A GrAF converter has been developed at ILC based on the GrAF example, the TO1 crawled data from ILSP and the the modified Freeling PoS tagging output.

The converter is a C++ tool which is divided in two main functionalities depending on input process parameter: 1) Creating the skeleton and 2) converting a tools output to the GrAF annotations format. The first part consists of creating the main files of a GrAF using an XCES crawled document: the header, the layout, the original and plain. The second uses the Freeling output to create the GrAF annotations and updates the GrAF header.

Both of the processes take one xml file as input parameter (the second one only for updating the header file), so the converter use Xerces-C++ libraries which is a simple and easy to use XML parser written in a portable subset of C++; also Xerces-C++ has intrinsic support for many type of encoding including UTF-8 and ISO-8859-1. Because it did not interest us build the document tree in memory and vice versa were interested in a simple and efficient access to an

XML document, our needs have been hijacked on event-based APIs (SAX is the best known example of such an API).

The converter has been modified at UPF to handle the IULA tagger output and make it ready for the stand-off annotations.

7.5 GrAF Travelling Object Document path

With the GrAF converter tool already developed it was deployed as a web service at UPF. In fact the converter was deployed as two different web services which were presented in section 3.3.2. It was also developed a workflow (presented in section 4.3.10) which can help understand the whole process from XCES crawled data to GrAF document with PoS tagging.

The graphic in section 14.4 shows the whole data path and shows all the files and input/output parameters. It can be seen that crawled data is processed with the graf_converter to obtain the so called skeleton (header, layout, original and plain). Afterwards, the plain text is used as input for the processing tool. Finally the output of the processing tool (in most examples, Freeling) is used together with the header to create an updated header, and the annotation files (pos, segmentation and sentence). All this output documents combined are the final GrAF document representation of the original Basic XCES document.

8 Other technologies

As explained in D3.2, UPF has deployed web services using Axis (for other projects). Most of those web services have been migrated to Soaplab due to its maintenance advantages. The few remaining web services will be further studied and may be also migrated.

However, Axis and other web services frameworks should be studied, as a fallback position or to be used on very specific situations when Soaplab cannot fulfil all requirements. If there're resources enough it could be interesting to deploy some web services using REST (described in D3.1) or developed using other frameworks. For example, a few web services using authentication could be an interesting example for professional web service providers and for the business models development.

9 Security

In a public infrastructure like the PANACEA platform many security issues arise. Having a lot of public web services presents numerous vulnerabilities that must be addressed. In this section, we're going to explain some ways to make our servers more robust against massive usage, hackers, etc.

9.1 Virtualization

The first option proposed and already being used by some partners is virtualization. Installing all the software and doing all the maintenance for a server has a high cost. A hackers attack or a malfunction may require the reinstallation of all the software; it may cause the loss of data, etc.

Virtualization lets the service provider to create a virtual machine (or many) inside a real machine. The virtual machine can be stopped, run, rebooted, etc. These virtual machines are the ones hosting the web services and the ones which may suffer problems due to its public access. Thanks to the virtualization, if one machine has a problem, the service provider can easily turn it off, restore a copy and reboot it.

This system also allows to separate web services in groups. Each group can be hosted in one virtual machine. If one web service has a malfunction or it requires maintenance only its virtual machine is affected, leaving the rest of virtual machines completely operational.

There are several software solutions for virtualization like vmware[25], virtualbox[26], KVM[27], etc. Some solutions are professional expensive solutions while others are free. UPF is using KVM to host several virtual machines in one server. One of these machines hosts the UPF web services and a new virtual machine is planned to host other web services.

Once a month, the virtual machines are automatically stopped and a full backup is created. The experience has been successful so far although it must be taken into account that the real machine hosting these virtual machines must be a considerably powerful server:

1. **Fast hard drives** : having one operating system (OS) reading and writing in one hard drive requires a less fast hard drive than, for example, 6 OS reading and writing on that same hard drive.

2. **Memory**: the total amount of memory must be divided between the real server and the virtual machines.

3. **Number of processors**: For a good performance it's necessary to have at least one processor for each virtual machine and one for the real machine.

In the end, from the UPF experience, it can be said that using virtualization has been very useful and it will continue being a good solution. Now there's also the possibility to rent a virtual server to host services. Amazon[28] offers very flexible options to rent their virtual servers. This kind of solution offers numerous advantages. For instance, there's no need to learn how to use the virtualization software which is complex. Even more, a high-tech company (or a company with a lot of resources) will have the latest software solutions and will be able to benefit from all the possibilities offered by the virtualization software. For example, complex virtualization can automatically detect a growing number of requests and automatically turn-on several virtual machines to fulfil this requests. Afterwards it will stop these virtual machines. This system optimizes the use of the machine resources and helps the service provider to avoid worrying about request peaks which are not easy to handle with simple virtualization solutions.

[25] http://www.vmware.com

[26] http://www.virtualbox.org

[27] http://www.linux-kvm.org

[28] http://aws.amazon.com/ec2/#pricing

9.2 Limiting Web Services

One way to protect from too many requests or abuses is to limit the amount of them. Ideally, it would be nice to monitor "who" is making requests, "how many" requests is he/she doing, "what" requests are being done.

There is sophisticated and expensive software to do all this monitoring and to block requests from users making too much requests. This software requires high cost maintenance and is usually used by a team of system administrators in charge to ensure the correct use of the web services.

UPF developers contacted with the European Bioinformatics Institute (EBI[29]) due to its vast experience as a web service provider in the bioinformatics field, its relation with myGrid and that some of their web services are in fact, Soaplab web services. EBI has a strict fair share policy for their users: there is a maximum requests per user policy, etc.

EBI support team reported us with some of the software used to monitor the use of their web services and their users and also made some recommendations:

- **Web load-balancer (Zeus ZXTM[30])**: Request rate limitation for the web site. Specific request rate limits for some heavily used services. Specific black-listing of IP addresses from which badly behaved requests are coming.
- **Batch queue system (Platform LSF[31])**: A fingerprint is generated per user and used to regulate the number of concurrent jobs running per-user, excess jobs are queued. Additionally resource limits are placed on jobs to ensure no one jobs hogs resources, for example memory and run time limits.
- **Application fram eworks**: The tool service frameworks used include additional methods for black-listing users and restricting the types of jobs which can be submitted. Mostly this consists of restricting the amount of input (most of the services accept only one input sequence for which there is often a size limit).
- **Valid e-mail address**: An important part of this is requiring that systematic users of the web services have to provide a valid e-mail address, so in the event of problems, either with the service or their usage, there is a way of getting in contact with them.

Some of these solutions are expensive professional solutions very well suited for EBI or other service providers with many requests. However, at the actual stage of the PANACEA platform and with the actual low number of users these solutions are too expensive (price and human). Nevertheless this is valuable information for the future of the platform and for service providers with growing demand.

[29] http://www.ebi.ac.uk

[30] http://www.zeus.com/products/load-balancer

[31] http://www.platform.com/workload-management/high-performance-computing, http://en.wikipedia.org/wiki/Platform_LSF

EBI support team also recommended limiting the size of the input data sent to web services. DCU developed a technique for limiting web services not only the size but the amount of concurrent running jobs. This work was presented in Section 3.1.5. DCU technique represents a low cost solution very well suited for PANACEA web service providers and represents a good start for limiting web services.

10 The previous evaluation

In D7.2 section 2.7 it was presented the "lessons learnt" from the first evaluation of the platform. Different tasks and efforts have been done to improve the platform and will be presented in this section.

1. In the lessons learnt it was pointed out that it was not easy to reach the Spinet web client to test web services from the Registry. This usability problem has been solved by adding a "Spinet link" in the main page for every Soaplab web service.

2. Service providers are encouraged to include as much detailed information about their web services as possible. The registry has a very user-friendly interface to facilitate this task. Also two disclaimers have been prepared regarding usage conditions which can be used for all web service providers. These disclaimers were presented in section 2.4.3.

3. There have been prepared new manuals and specially videos which represent a very useful tool. It has also been prepared a document listing all the relevant manuals and tutorials which is very useful and has links to non-PANACEA documentation. For example, mygrid has a particularly good Taverna tutorial which has been used for PANACEA developers to learn how to build workflows. All this documentation was presented in section 2.4.

4. It was necessary to have a place to find workflows. The PANACEA myExperiment portal fulfils this requirement perfectly and it was presented on section 0. Users can easily find example workflows, share their own workflows and files creating a community that benefits from all its users contributions.

5. The Common Interface required more documentation than only what was documented on D3.1. This necessary extra documentation was explained on section 2.4.1 and can be found on the PANACEA website.

6. The PANACEA Registry has some improvements, as mentioned before as well much more annotations done by the web service providers. The PANACEA Registry work was presented in section 3.6.

11 Workplan updates

The workplan updates are listed and described in this section.

Web Services

WS-GR-1: Temporary files management. This task was originally due to the first version of the platform. It was postponed for t22 because it made more sense developing it together with the massive data tests.

The Temporary files deletion system was presented in section 3.1.6 of this document. However, during the massive data tests it was seen that this task cannot be considered closed. Massive data has considerable development implications with the growing number of processed files and number of experiments.

The 3rd version of the platform must have an adapted temporary files management system.

WS-GR-02: Provenance . While the provenance information stored in the headers of the Travelling Object remains useful and being used in the workflows, the provenance regarding the execution times, processes, etc. is still not easy to get. ILC developed a task to get that provenance from the Taverna database. However, considering that new Taverna was supposed to be released on May the task was postponed expecting considerable improvements on the Taverna provenance usability. Due to the fact that Taverna 2.3.0 was released on August with no time enough for a deep analysis this task is postponed. However, basic Taverna provenance can still be obtained and copied from the graphical interface: execution time, iterations, time per iteration, etc. which is usually enough.

Common Interfaces

WS-CI-02: Controlled vocabularies. The idea of this task was to have controlled vocabularies for some specific situations during the CI definition if necessary. It was also considered as a possible closed set of tags for the Registry metadata used to describe services. There has not been this necessity so far therefore this task is postponed to the 3rd version of the platform in case it is necessary.

Axis

All Axis tasks are deprecated and Axis and other frameworks for web services will be considered fall back positions or used only in very specific situations. Soaplab has proven to be useful and scalable.

Web Services alternatives

These tasks will be compacted as a one task with a specific framework for the 3rd version of the platform in case there are enough resources (basically time).

Alternatives Test

Tests with grid technologies, UIMA or GATE have not been carried out since massive data tests have consumed a lot of resources. These alternatives will be studied and tested in case PANACEA needs it.

12 Conclusion and future work

The second version of the platform is working and it can be used. Service providers deployed WP4 CAA, WP aligners, converters and other web services. The Registry and the PANACEA myExperiment portal can be used to find and share web services and workflows. Improved documentation and videos have been shared in the PANACEA website to assist new users. New workflows have been developed to handle larger amounts of data and massive data solutions have been implemented. Finally a new stand-off Travelling Object has been introduced to the platform. GrAF format will be used for the new tools and workflows coming for the 3rd version of the platform which require a stand-off format.

All tools are being used among partners and have proven to be useful and user friendly. Soaplab is used by service providers to easily deploy their tools as web services and Tavern allows developers to design complex workflows with a friendly graphical interface. The Registry and myExperiment are being used and represent a perfect point of entry to the platform.

The platform needs to handle massive data are numerous and require a lot of focus and resources (human and machine). From the service provider point of view maintenance becomes a fundamental issue to have a good quality service. Tomcat, Soaplab, temporary files management, etc. all need special care and to handle even more data it will require improved parameterization and better scripts which take machine resources into account. Experiments with 5k documents have been carried out successfully and larger amounts will soon be considered.

The bug errors found during the massive data tests must be addressed. Service providers will be required to carry a massive data test in their servers to detect Tomcat, soaplab, temporary files management and Taverna problems. It will also be used to detect the maximum capacity of the servers. It must be taken into account that the massive data tests were carried out with a very stressful parallelization. Also running experiments from different locations (different Taverna instances) will be tested.

Collaboration with Taverna and Soaplab support teams has been productive and is expected to be helpful during the rest of development. All software support teams' support is fundamental when dealing with massive data. Using a non-optimal version of Tomcat, or some incompatibility with java or precompiled packages has a huge impact on massive data experiments. To this aim service providers should have a strict control of their software versions, logs, be well informed about software updates and report problems as soon as possible with all relevant information and logs.

The new and numerous web services to be deployed for the 3rd version of the platform as well as their combination in workflows will require a considerable effort for all service providers. Common Interfaces will have to be designed if necessary and collaboration from all partners will ne fundamental to meet the schedule.

Massive data is still the most challenging aspect for WP3. Experiments with more than 5k documents and more complex workflows will be addressed. Using the optimal software version combination of all programs and improving the servers' maintenance system will considerably improve the capacity of servers and will help PANACEA handling massive data.

13 Bibliography

[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] Poch, Marc, Prokopis Prokopidis, Gregor Thurmair, Carsten Schnober, Riccardo Del Gratta, and Núria Bel. 2010. *D3.1 - architecture and design of the platform*. Confidential deliverable, The PANACEA Project (7FP-ITC-248064).

[GrAF] Nancy Ide, Keith Surderman. 2007. "*GrAF: A Graph-based Format for Linguistic Annotations*". In Pro-ceedings of the Linguistic Annotation Workshop (June 2007), pp. 1-8.

[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.

14 Annex

14.1 Web Services Disclaimers

14.2 Usage conditions

These disclaimers can be used in any kind of documentation for a web service and are being used in the "Usage conditions" metadata field of the PANACEA registry.

14.2.1 Temporary files deletion

Temporary files may be generated by the various processes for their needs and operations.
Temporary files will not be used by anyone but the actual user of the input data that generated them. This is part of our data protection policy aimed at safeguarding the owner rights on the data travelling through the web services.
Temporary files will be automatically deleted from the system **after N days**, even if they are not accessible to anyone but the actual user.
It is the sole responsibility of the input provider to check and ensure that (s)he has the right to use the input data provided to the platform.
No access or use of the temporary files will be allowed other than estipulated in this disclaimer.

14.2.2 Fair Share Policy on Parallel Process Running

Users are kindly asked not to submit **more than N processes**/requests in parallel. This is part of the fair share policy implemented so as to allow all users to benefit from the web services offered by the PANACEA platform.
If this policy is not complied with in a way that prevents other users from using the web services, users concerned may be prevented from submitting processes/requests, their exceeding processes may be killed and they may be black-listed for future use.

```
In the event of an exceptional need to use the platform in a manner not
covered by this disclaimer, users are kindly adviced to address the contact
point of the web service(s) required so as to study the possibility of
establishing an exceptional usage for those web services.
```

14.3 Workflow images

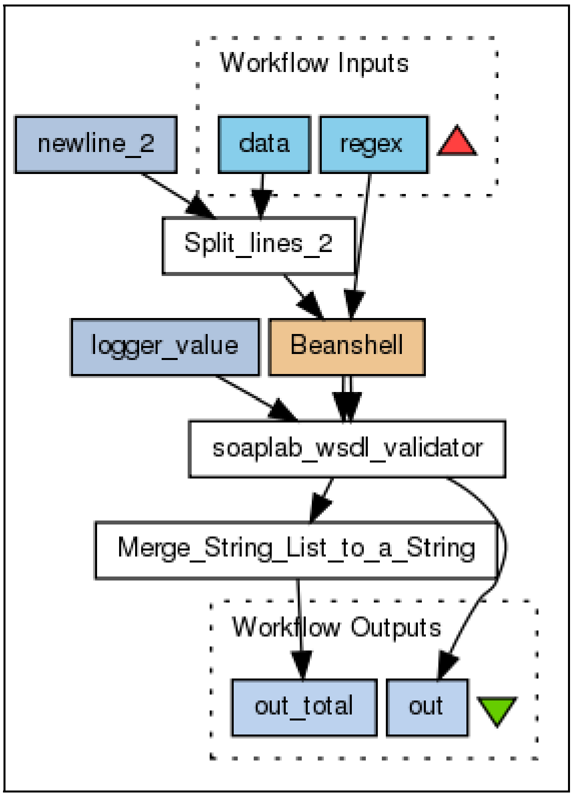


Figure 4: Panacea Common Interface validation for Soaplab web services

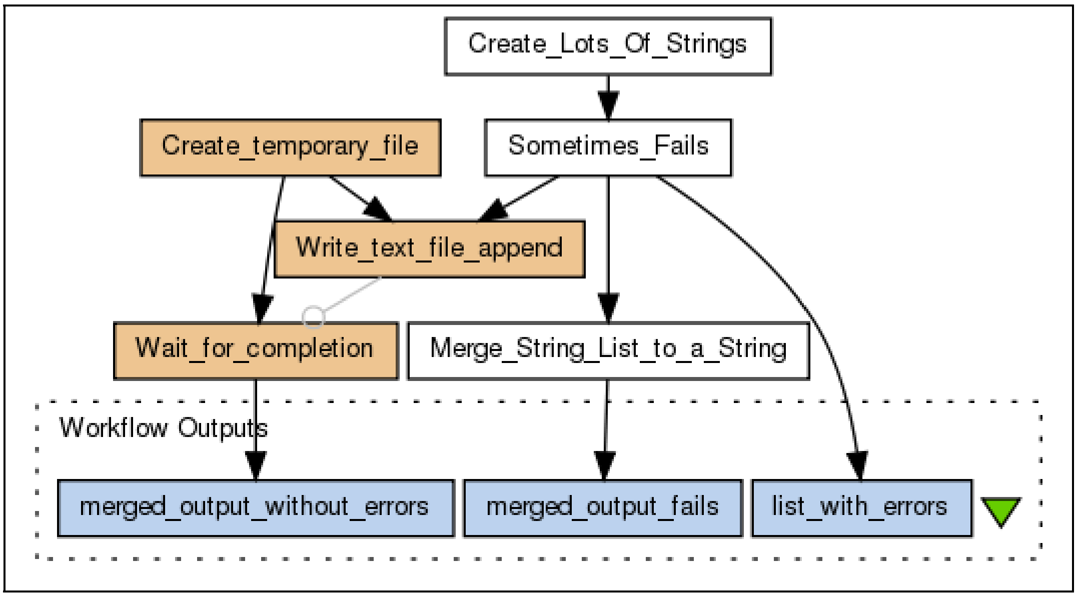


Figure 5: Merge list of errors to string

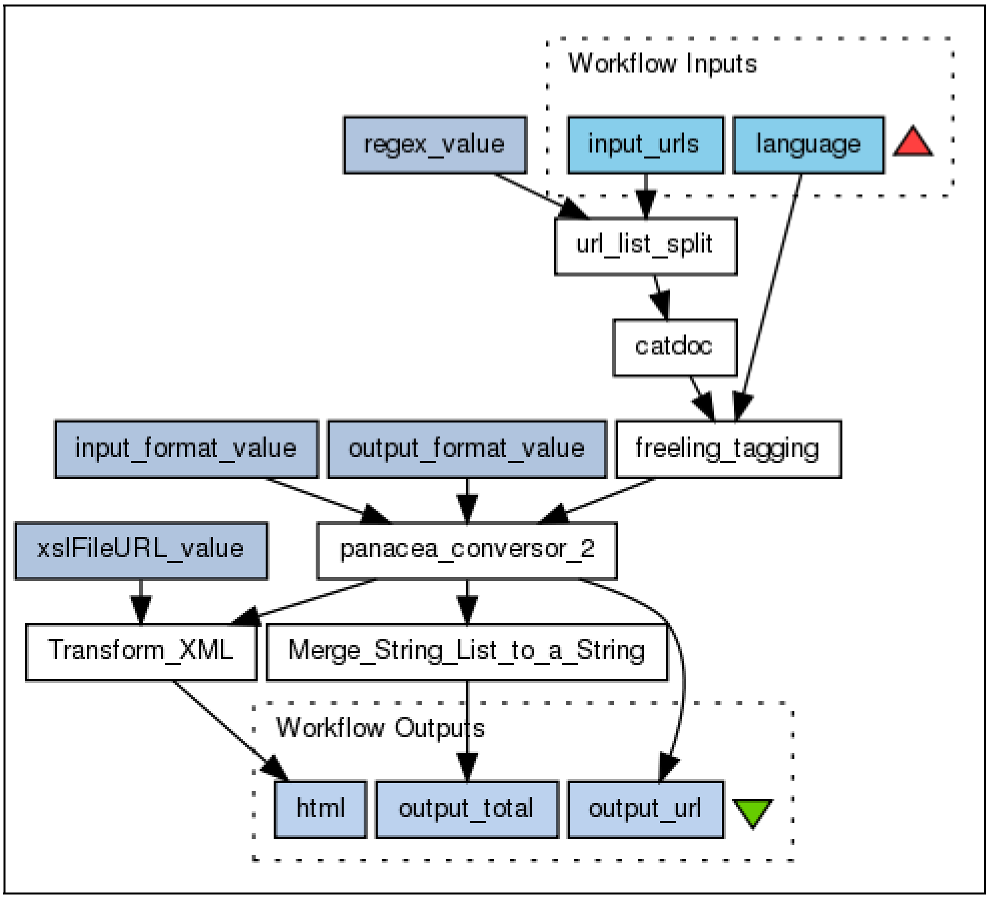


Figure 6: WORD freeling tagging and stylesheet

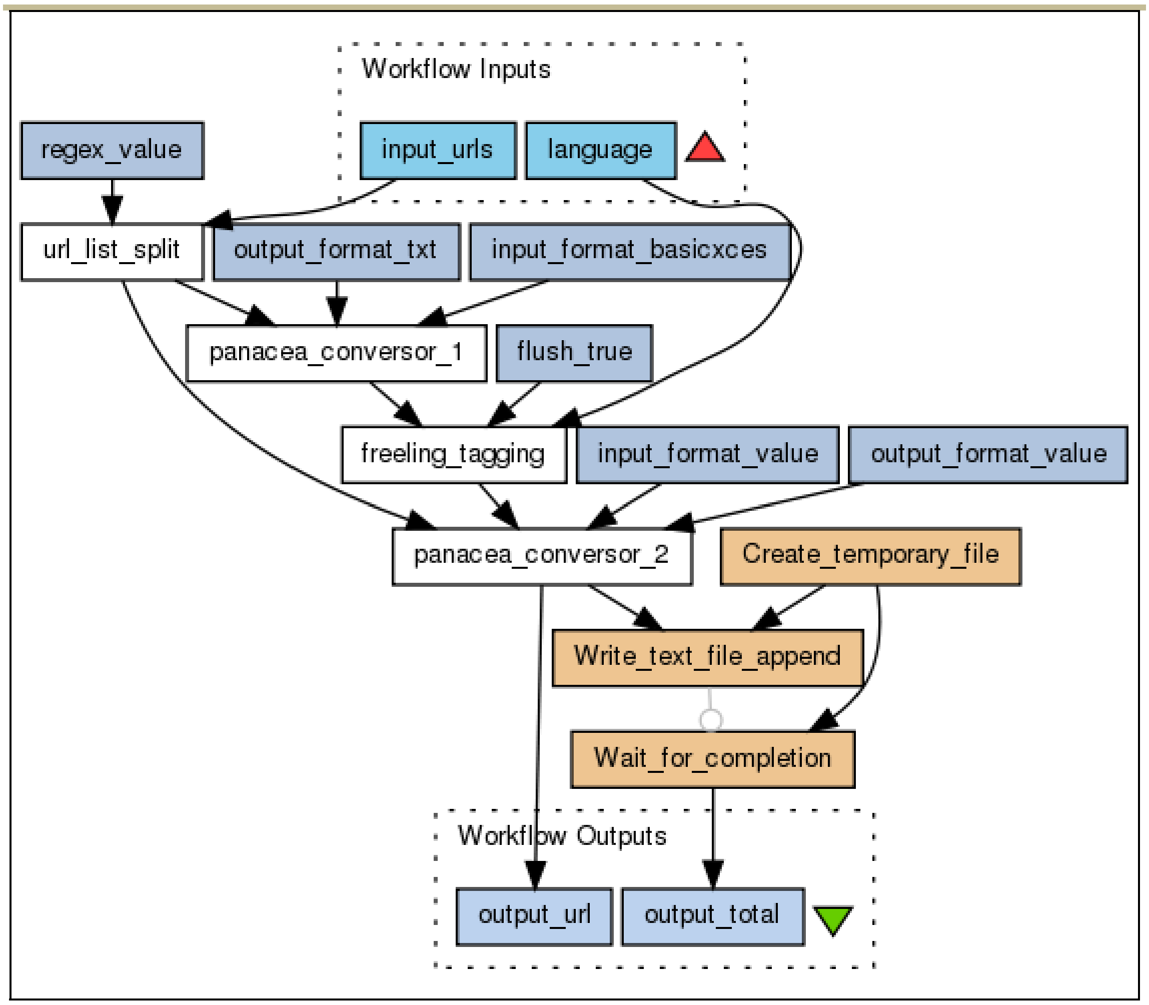


Figure 7: Freeling tagging for crawled data

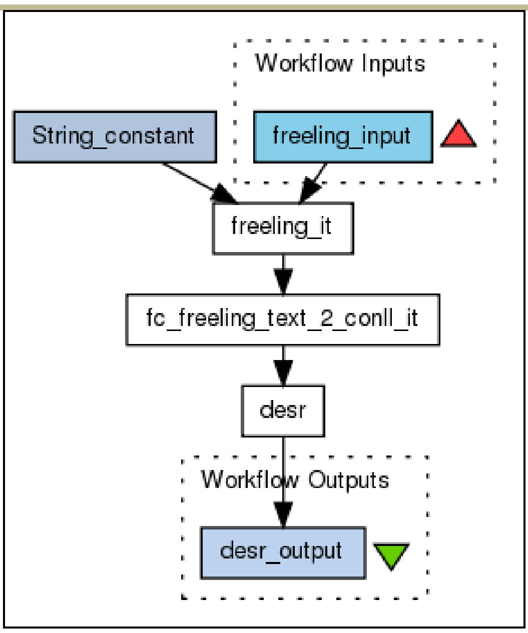


Figure 8: Freeling to Desr - From text cleaned to text parsed

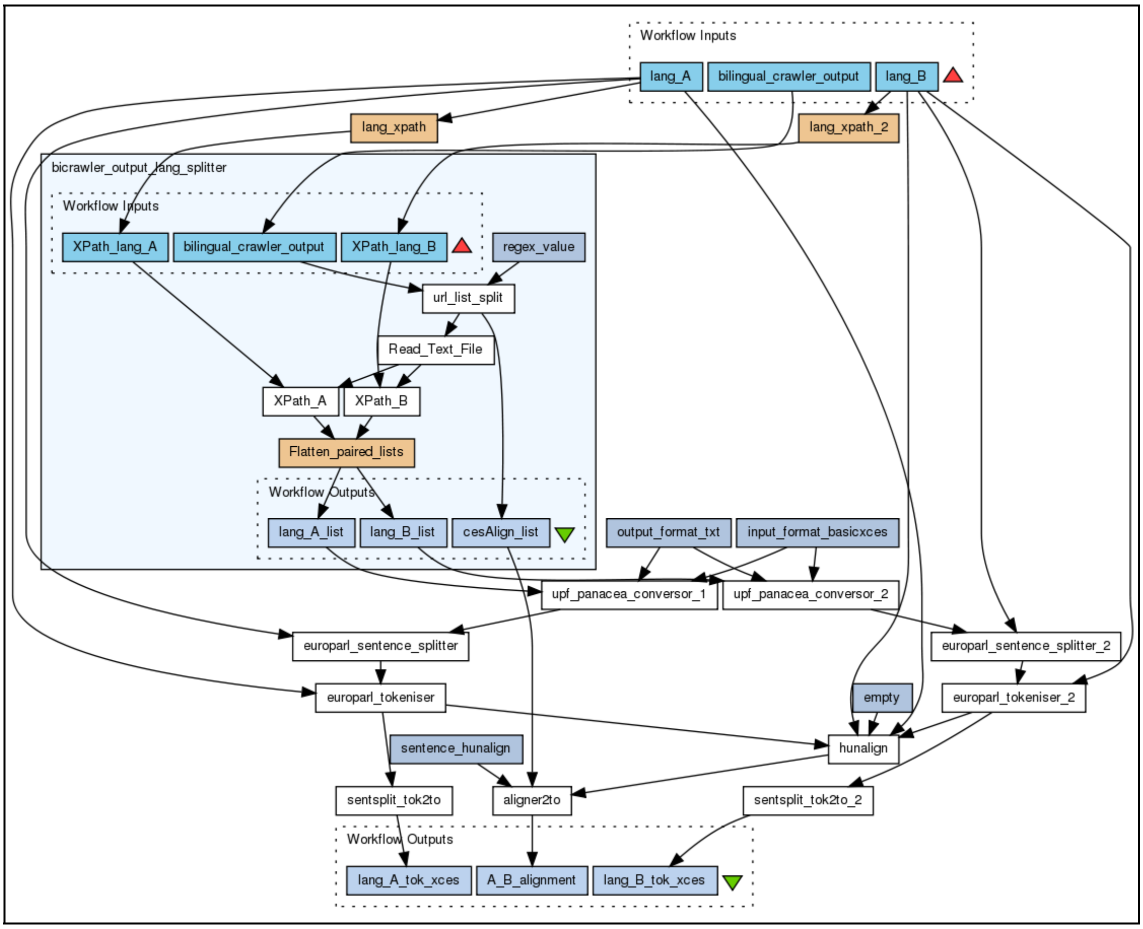


Figure 9: bilingual sentence alignment for crawled data

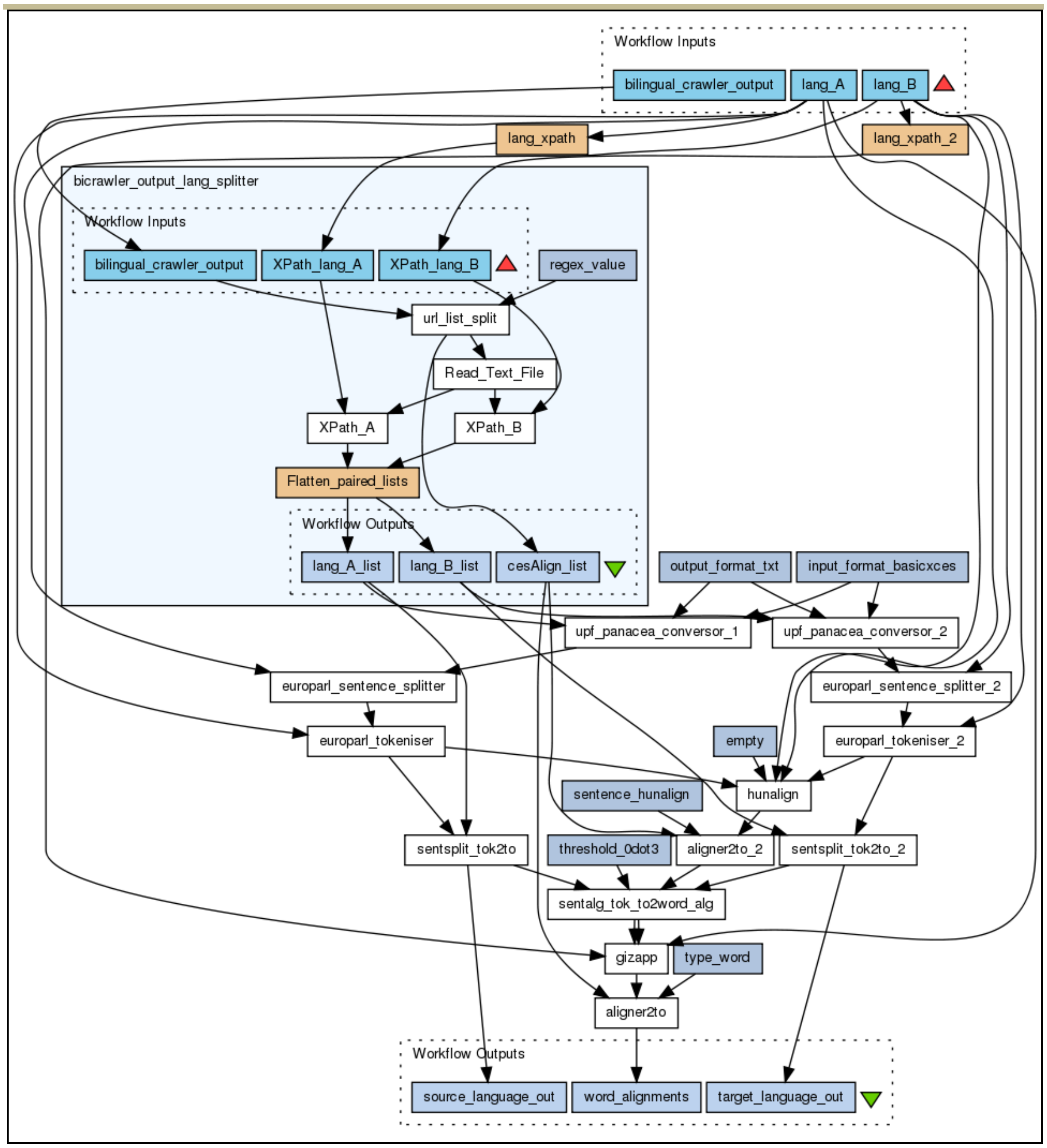


Figure 10: bilingual word aligner for crawled data

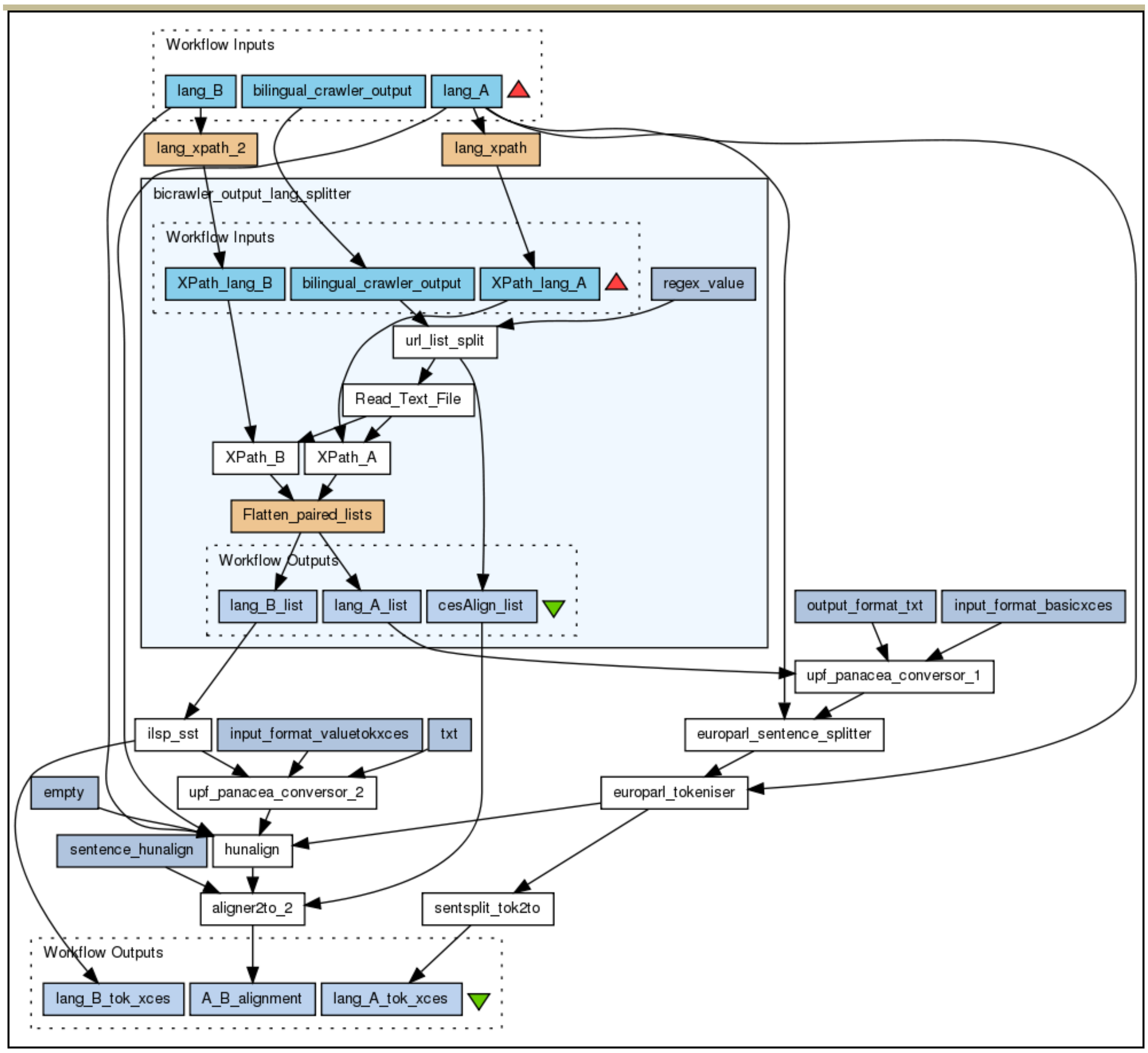


Figure 11: bilingual sentence alignment for crawled data EN EL

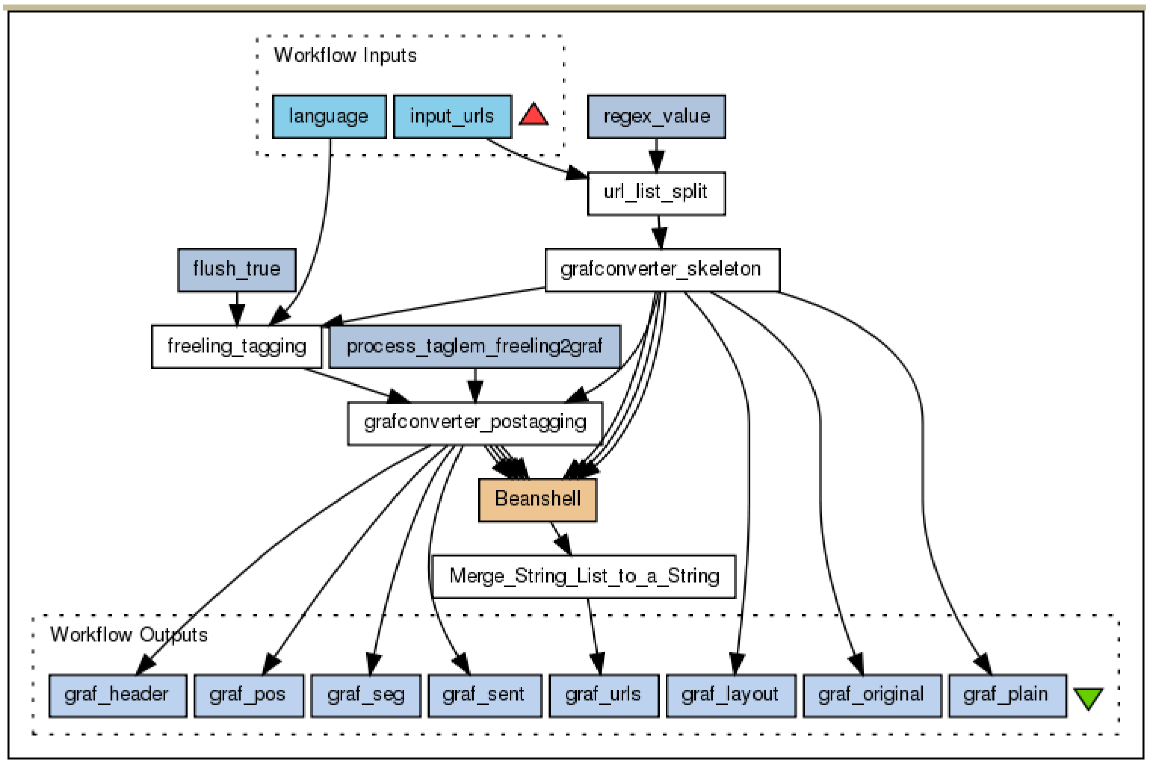
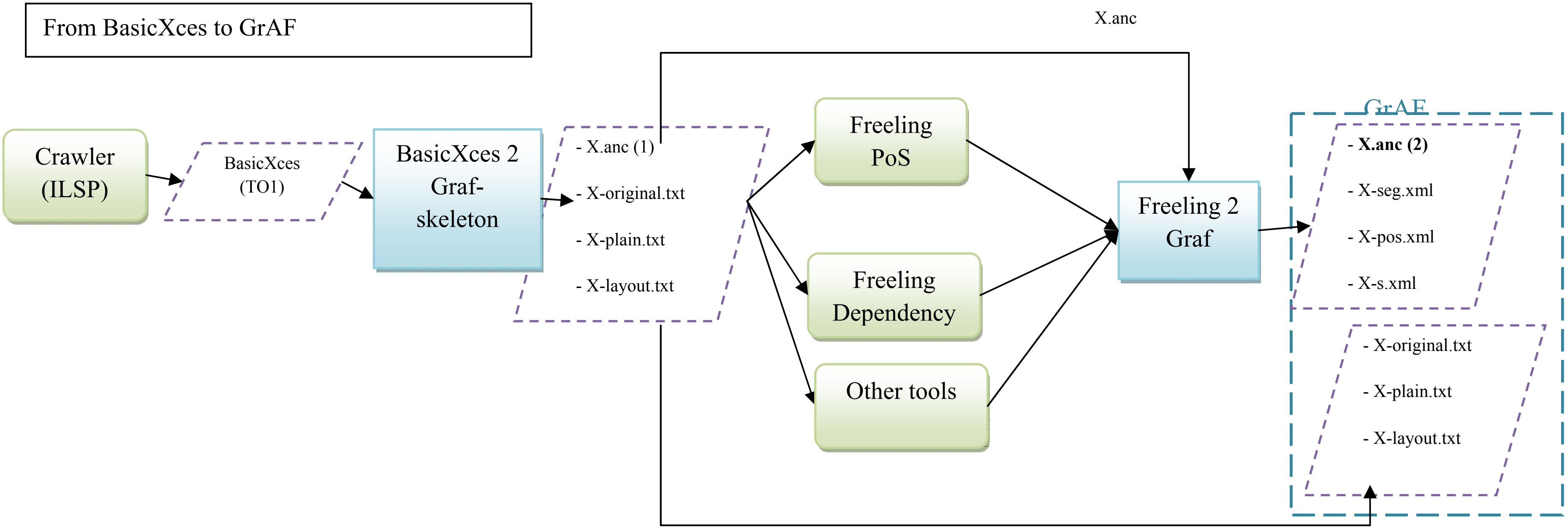


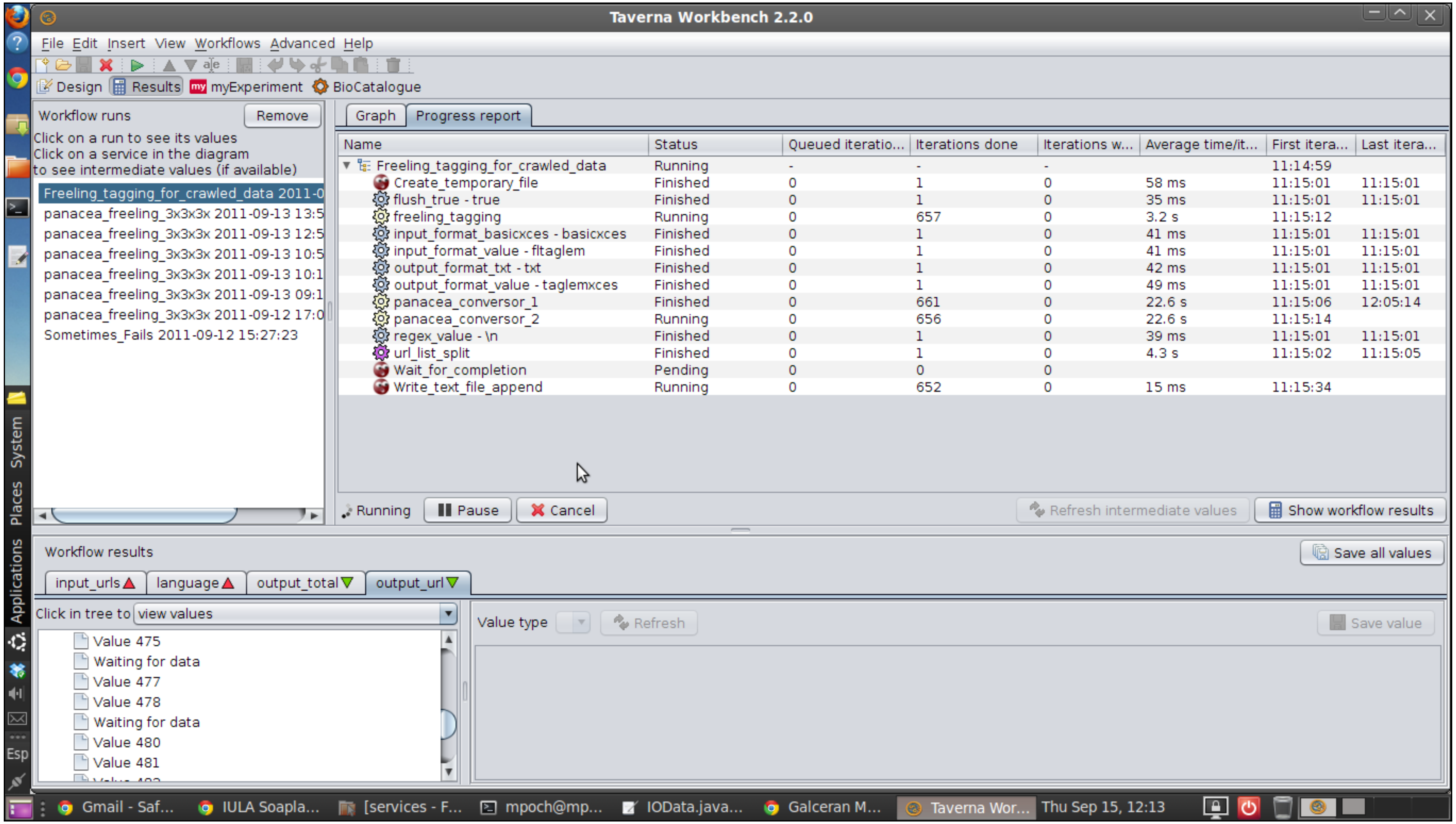
Figure 12: GrAF PoS tagging with Freeling for basicxces documents

14.4 From BasicXces to GrAF

This graphic shows the data path from Basic Xces data to Freeling or other tools processing and finally the resulting GrAF document.



14.5 Taverna captures



Click on a run to see its values
Click on a service in the diagram
to see intermediate values (if available)

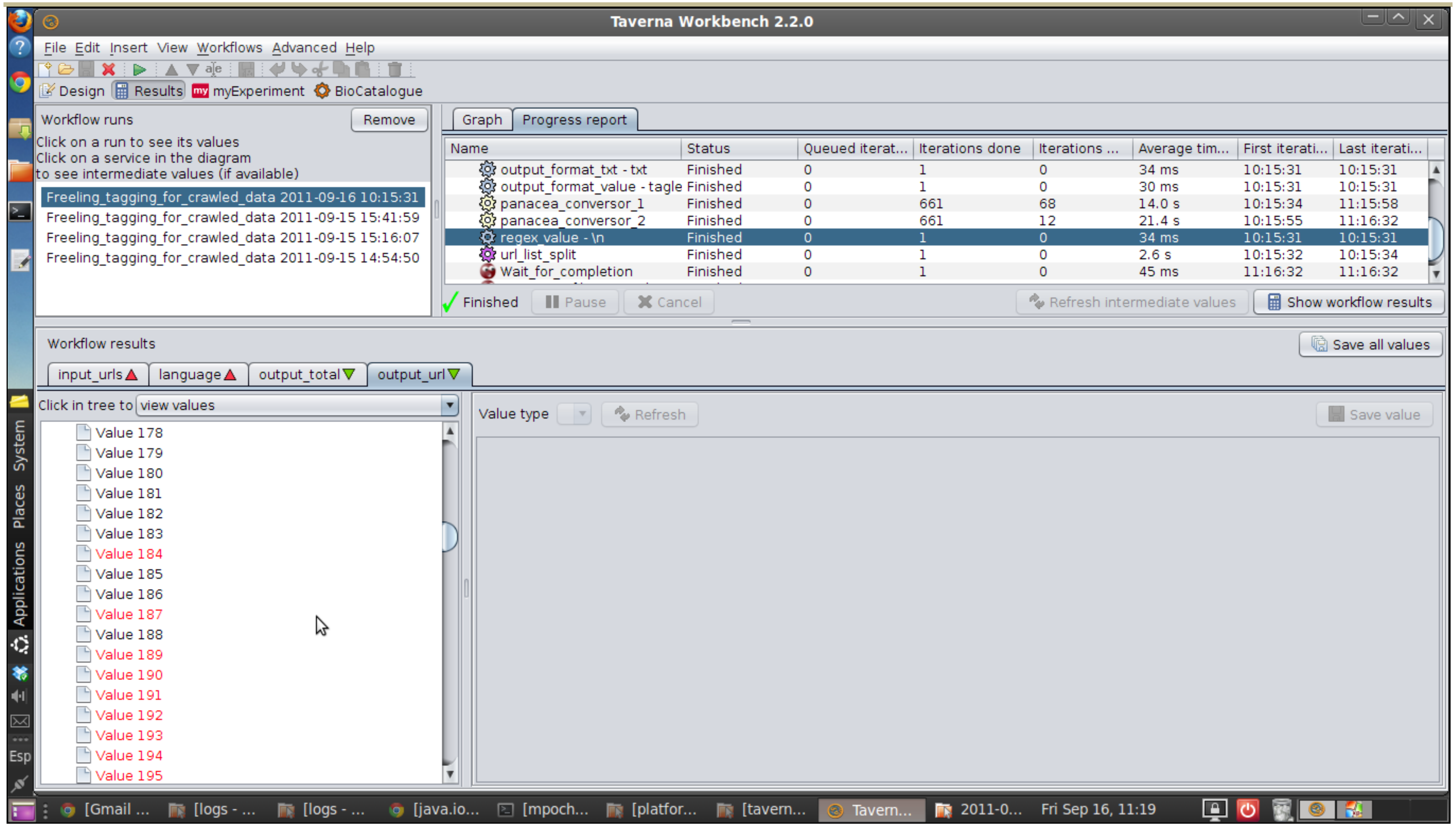
Freeling_tagging_for_crawled_data 2011-0
panacea_freeling_3x3x3x 2011-09-13 13:5
panacea_freeling_3x3x3x 2011-09-13 12:5
panacea_freeling_3x3x3x 2011-09-13 10:5
panacea_freeling_3x3x3x 2011-09-13 10:1
panacea_freeling_3x3x3x 2011-09-13 09:1
panacea_freeling_3x3x3x 2011-09-12 17:0
Sometimes_Fails 2011-09-12 15:27:23

Name	Status	Queued iteratio...	Iterations done	Iterations w...	Average time/it...	First itera...	Last itera...
▼ Freeling_tagging_for_crawled_data	Running	-	-	-		11:14:59	
Create_temporary_file	Finished	0	1	0	58 ms	11:15:01	11:15:01
flush_true - true	Finished	0	1	0	35 ms	11:15:01	11:15:01
freeling_tagging	Running	0	657	0	3.2 s	11:15:12	
input_format_basicxces - basicxces	Finished	0	1	0	41 ms	11:15:01	11:15:01
input_format_value - fltaglem	Finished	0	1	0	41 ms	11:15:01	11:15:01
output_format_txt - txt	Finished	0	1	0	42 ms	11:15:01	11:15:01
output_format_value - taglemxces	Finished	0	1	0	49 ms	11:15:01	11:15:01
panacea_conversor_1	Finished	0	661	0	22.6 s	11:15:06	12:05:14
panacea_conversor_2	Running	0	656	0	22.6 s	11:15:14	
regex_value - \n	Finished	0	1	0	39 ms	11:15:01	11:15:01
url_list_split	Finished	0	1	0	4.3 s	11:15:02	11:15:05
Wait_for_completion	Pending	0	0	0			
Write_text_file_append	Running	0	652	0	15 ms	11:15:34	

Click in tree to view values

Value 475
Waiting for data
Value 477
Value 478
Waiting for data
Value 480
Value 481

Figure 13: Never finishing execution



Click on a run to see its values
Click on a service in the diagram
to see intermediate values (if available)

Freeling_tagging_for_crawled_data 2011-09-16 10:15:31
Freeling_tagging_for_crawled_data 2011-09-15 15:41:59
Freeling_tagging_for_crawled_data 2011-09-15 15:16:07
Freeling_tagging_for_crawled_data 2011-09-15 14:54:50

Name	Status	Queued iterat...	Iterations done	Iterations ...	Average tim...	First iterati...	Last iterati...
output_format_txt - txt	Finished	0	1	0	34 ms	10:15:31	10:15:31
output_format_value - tagle	Finished	0	1	0	30 ms	10:15:31	10:15:31
panacea_conversor_1	Finished	0	661	68	14.0 s	10:15:34	11:15:58
panacea_conversor_2	Finished	0	661	12	21.4 s	10:15:55	11:16:32
regex value - \n	Finished	0	1	0	34 ms	10:15:31	10:15:31
url_list_split	Finished	0	1	0	2.6 s	10:15:32	10:15:34
Wait_for_completion	Finished	0	1	0	45 ms	11:16:32	11:16:32

Value 178
Value 179
Value 180
Value 181
Value 182
Value 183
Value 184
Value 185
Value 186
Value 187
Value 188
Value 189
Value 190
Value 191
Value 192
Value 193
Value 194
Value 195

Figure 14: errors burst