IST FP7 231507

D4.4 Release Notes

File name	PuppyIR-D4.4-ReleaseNotes-v1.0
Author(s)	Richard Glassey (UGLW) Leif Azzopardi (UGLW) José Miguel Garrido (ATOS) Jeldrik Schmuch (ATOS) Paul Moore (ATOS)
Work package/task	WP4
Document status	Final
Version	1.0
Contractual delivery date	30 November 2010
Confidentiality	Public
Keywords	Implementation and Framework Release
Abstract	Release notes to accompany the initial release of the PuppyIR Framework (v1.0)

Table of Contents

Executive Summary

This report accompanies the release of the first version of the PuppyIR framework. The release notes provide an overview of the framework development status, the features that the framework currently supports, the requirements and installation guide, known issues, and an outline for the development activities ahead of the second version of the framework.

This deliverable builds upon the previous WP1 and WP4 deliverables:

- D1.2 – Agreed User Requirements and Scenarios **[2]**
- D1.3 – Agreed Technical Requirements **[3]**
- D3.1 – Report on Data Pre-processing **[4]**
- D4.1 – Specification Report **[5]**
- D4.2 – Design Report **[6]**
- D4.3 – Report on Implementation and Documentation **[7]**

The source code and documentation can be accessed from the project's SVN repository hosted on Sourceforge at http://sourceforge.net/projects/puppyir/.

The framework can also be downloaded directly using these links:

Framework: http://sourceforge.net/projects/puppyir/files/release/version-1-alpha/puppyir-1.0.tar.gz/download

Multitouch Prototype: http://sourceforge.net/projects/puppyir/files/release/version-1-alpha/Multitouch_protoype.zip/download

1 Introduction

The PuppyIR Project aims to facilitate the creation of child-centred information services, based on the understanding of the behaviour and needs of children **[1]**. The goals of this report are to:

1) Provide an overview of the framework status
2) Outline the features that are currently supported
3) Explain the requirements for installation
4) Describe known issues
5) Provide a roadmap for the development of the second version

This report serves 3 key purposes:

1. Provide release notes to accompany the source code for the first version of the framework for project developers.

2. Describes the current framework feature set, noting changes from previous deliverables.

3. Communicate to all project members and stakeholders how the project is progressing from the initial release towards more mature iterations.

1.1 Implications from D4.3 – Report on Implementation and Documentation

In **D4.3 – Report on Implementation and Documentation [7]**, the state of the framework development was described, discussing the rationale for the technology choices that were adopted, detailed the prototypes that have been produced using these technologies, and finally how the framework was progressing in terms of its implementation. As that report covered the main aspects of framework development, this report will be necessarily brief, covering the important aspects of the framework without delving into too much detail and repetition of earlier reports. The primary component of this deliverable is the source code for the framework, currently hosted on Sourceforge at this address:

http://sourceforge.net/projects/puppyir/

As the framework is still under development, there will be changes made to the specifications and design as required, and these will be reported as part of the future deliverable, **D4.5 – Report on Specifications and Design Changes**.

1.2 Structure

The report is divided into the following major sections:

- **Section 2 – Framework Features**
- **Section 3 – Requirements and Installation**
- **Section 4 – Known Issues**
- **Section 5 – Future Plans**

2 Framework Features

The initial version of the framework has reached its core functionality. As stated in **D4.2 – Design Report (section 7) [6]**, the framework 1.0 is implemented in Python and divided in five major components:

- Service
- Model
- Query
- Search
- Results

A description of the component functionality can be found in the **Section 4** of Deliverable **D4.2**.

2.1 Global features of 1.0

The main features of the framework are provided by each of these five components, and their interaction with each other. This initial version of the framework assists a developer by providing a unified and extensible interface to multiple search engines, a common data model that abstracts over the differences in search results returned by each separate search engine, and a set of information processing components to help build a PuppyIR service for children.

- The framework can search using several search engines, including Yahoo!, Google, Bing or any OpenSearch compatible engine. The framework offers an abstracted interface for the different engines, so the developer does not need to be aware of the different search engines interfaces. It is possible to add new search engines if required.

- The search engine wrapper adds an abstraction layer over low level details, as service addresses, proxies, IDs

- It is possible to add, delete or change the search terms automatically, using an open architecture of filters. Several filters can be organized like a pipeline, modifying the terms step by step.

- The search can also be decorated with suggestions.

- The search results can also be supervised, using a different set of filters.

- The search results are converted to the Open Search standard for search engine results, regardless of which search engine is used.

2.2 New features in 1.0

Most part of the features, the modules and the API of the framework were previously described in detail in Deliverable **D4.2 section 4**. There are no changes in the elements described in **D4.2**; therefore we will not repeat this description here. So, only new features added after **D4.2**, will be described.

2.2.1 REST interface

To support multiple client devices, beyond a simple desktop and web browser scenario, the framework exposes a REST (Representational State Transfer) interface. The purpose of the interface is to expose the most important interactions (e.g. querying, making suggestions, and

returning results) that a PuppyIR service would be expected to support. Using the REST interface, a service can be accessed from most client devices.

In order to call the framework, it is only required to call a REST web service

```
http://127.0.0.1:8000/queryrest/?query=elmo
```

The response from the framework is an OpenSearch-compatible XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:opensearch="http://a9.com/-
/spec /opensearch/1.1/">
<title>Yahoo!: elmo for kids colouring book</title>
<link>*** missing from Response ***</link>
<description>Search results for Search results for 'elmo for kids colouring
                          book ' at Yahoo!</description>
<opensearch:totalResults>101391</opensearch:totalResults>
<opensearch:startIndex>1</opensearch:startIndex>
<opensearch:itemsPerPage>10</opensearch:itemsPerPage>
<opensearch:Query role="request" searchTerms="elmo for kids colouring book"
                          startPage="1" />
<entry>
       <title>Kids Coloring Books - Compare Prices...</title>
       <link>http://www.nextag.com/kids-coloring-books/products-html </link>
       <content type="text">
              ... Herve Tullet, The Coloring Book, Toddler and Kids…
       </content>
</entry>

<entry>
       <title>Free Coloring Pages</title>
       <link>http://www.coloring.ws/coloring.html</link>
       <content type="text">
              Offers downloadable pictures for kids to color in.
       </content>
</entry>
<entry>
       <title>22 Printable Elmo Colouring Pictures Colouring Book</title>
       <link>http://www.colouringbookpages.co.uk/elmo/ </link>
       <content type="text">
       Elmo Colouring Books. Below is a selection of 22 Elmo colouring pages
       for kids. ... Web Design ┬® 2010 Free Colouring Book Pages and
       Childrens Printable ...
       </content>
</entry>
</feed>
```

From the implementation point of view, the REST interface is made using the “django-rest-interface” **[10]**, taking into account that the framework will use Django **[9]** for the configuration web pages. The REST interface makes use of the same server as Django, so it is needed to start the Django server to deliver the REST web services responses.

To simplify integration, a customized file similar to the standard `manage.py` has been created. This way, developers do not have to adjust environment variables such PYTHONPATH. The Django server is started now with the command:

```
python django_server.py runserver
```

2.2.2 Wrapper for Lucene/Solr search engine

A new search engine wrapper for Lucene has been created. Lucene is an open source library for searching structured contents. Although Lucene is developed in Java, and therefore not as simple to integrate into a Python, the Solr project includes the Lucene library integrated with a web service interface that is programming language agnostic.

It should be noted that Solr is not a web search service in the same sense as Yahoo! or Google. In the case of Solr, there is no single entry point or a single response format. Developers are free to create their own customized search application, according to their needs, so the wrapper for Solr will also need to be tailored for specific applications.

2.2.3 Additional filters

There are three new filters. These filters are useful for deleting obscene words or terms not suitable for children. The list of forbidden terms is passes as a parameter to the filters.

- **BlackListFilter.py**:
    - This is a query filter; it erases the inappropriate terms provided as queries.
- **BlackListResultFilter.py**:
    - This filter is quite similar. It erases inappropriate words but from the result set. The bad words in the title and abstract for a result are substituted by asterisks. This approach has a setback: the original web page can be accessed unchanged using the provided link.
- **ExclusionFilter.py**:
    - This filter deletes the results containing inappropriate words in the title or the abstract. These results are deleted from the results lists, so the user does not see these results.

2.2.4 Basic test suite

There are two new filters intended for testing and debugging purposes:

- **TestEqualFilter.py**: this filter works like an “assert” statement in the language Java or Python.
- **EchoSearch.py**: It is a false search engine for debugging, it returns the search terms as a response.

There are examples of the use of these modules in the directory **test**

2.2.5 Logging

There is a new functionality for basic server side logging in the version 1.0. It uses the standard logging library from Python 2.X, so there are no new dependencies added for the logging feature. The format of the logging file is a line for individual query or result, with a fixed schema:

```
QUERY user queryID date/time query-terms query-filters

RESULT user resultID:queryID date/time result-link result-filters
```

3 Dependencies and Installation

At this stage of the project, the exact requirements of the deployment environment are still changing with progress in development. This results in the requirement that using the early version of the framework requires competence with system setup and installation of packages.

3.1 Core Software Dependencies

The core requirements for the framework are as follows:

- **Python 2.6 (not version 3.0) [8]**
    - It should be noted that at present we do not support Python 3. Although this is the latest version of the programming language, most of the major libraries that the framework incorporates, or builds upon have not yet released a 3.0+ compliant version. It is recommended that developers use 2.6 (or later 2.X versions) until this situation changes. Multiple versions of Python can be supported on a single system, so it should not be too much of an issue to make a 2.6 version available on any system.
- **Django 1.02 (and above, recommended 1.2) [9]**
    - Django is under constant development, and crucially this development incorporates important security and bug fixes. It is safe to use the latest branch of the Django framework, although caution is advised for major version changes. The development of the PuppyIR framework will always endeavour to use the latest version of Django to ensure compatibility.

3.2 Additional Requirements

The framework requires a collection of extra libraries. This list is currently changeable, and some libraries may not be required, or might be incorporated into the framework source code for developer convenience. For instance, running certain prototypes require different XML processing libraries to be installed, whilst the multitouch client prototypes require MT4J, but this is already included in the source code. As future releases are delivered, this list should stabilise.

- **Django-REST-interface [10]**
    - Used in the framework for the REST interface.
- **BeautifulSoup - Python Library for XML processing [11]**
    - Used in the framework branch
- **Lxml – Python Library for XML processing [12]**
    - Used in the prototypes branch
- **JQuery – Javascript Library for AJAX/Web Interface [13]**
    - Used in the prototypes branch
- **MT4J – Multitouch for Java [14]**
    - Used in the multitouch interface branch

At this moment, Django REST Interface is only distributed from its Subversion Repository, it is not directly downloadable. In order to make the installation process easier, it is distributed included with PuppyIR by now.

3.3 Installation Notes

Full installation notes for each dependency can be found on their project websites. Notes on installing the framework from the Sourceforge repository are available from two locations. **D4.3 – Report on Implementation and Documentation [2]** contains a "Getting Started" guide and a tutorial on "Building a Basic PuppyIR Service". A short guide is currently hosted on Sourceforge and contains an installation guide for requirements, the current API for the framework (auto-generated from source code comments), a guide to using the SeSu (search and suggest) prototype, and creating custom interface themes for puppy services:

http://sourceforge.net/apps/trac/puppyir/export/98/trunk/prototypes/sesu/doc/build/latex/sesu.pdf

4 Known Issues

The framework is currently in its early stages of development, and as a matter of course will have issues in need of addressing. The major issues of this version of the framework are summarised below.

4.1 Integration

The largest issue with the framework is the current level of integration with Django. As much as possible the development of the PuppyIR Framework is loosely coupled from the web application component that Django provides.

The reason for this is twofold. Firstly, by separating out the framework from the web application component, it can be run in a simple command line only mode of operation that does not require a web or multitouch interface. Secondly, the effort required to test the framework is simplified by not requiring extra components to be taken into consideration, i.e. we test the PuppyIR source code rather than other 3rd party projects over which we have little control or resources to make improvements.

The outcome of this issue is that at present, the developer is required to put more effort into using PuppyIR with Django. Although this issue is common (merging independent frameworks) it does mean that PuppyIR could be integrated into other frameworks if required, and that we are not entirely dependent on the availability and stability of the Django project.

4.2 Testing Framework

At this stage, there is only minimal testing of the framework code. This issue will be addressed for the second and future releases as the development of the framework stabilises. The main testing strategy will be to ensure that core modules have sufficient testing coverage using standard unit testing, and that integration testing of these components will be achieved by creating a small suite of mock-services and testing them with known query, data, and search results.

4.3 Client Interfaces

Related to the first issue of integration, a second level of integration occurs at the client interface layer. Whilst the design of the framework has ensured that all clients communicate with a service using common HTTP methods over TCP/IP, documentation to support a developer who wants to build a web or multitouch interface to a service should also be provided.

4.4 Configuration

At present, the configuration of the services is manual and requires the editing of source code, instead of configuration files. This is sufficient for the early stages of the project and initial prototypes, but will need to be made user friendly in future releases.

4.5 Documentation

As the development of the framework is ongoing and significant changes are occurring with some regularity, it is inevitable that the documentation will become out of sync. To mitigate this, documentation is generated directly from code comments, but this process could be incorporated into the release cycle as well as having a review process prior to major releases to ensure consistency, compressibility and technical quality.

5 Future Plans

The following areas of interest for development will factor into the release of the second version of the framework, and will be reported in the next deliverable, **D4.5 – Report on Specification and Design Changes**.

- **Web-based Service Configuration:**
    - At present, configuring a service requires manual editing of source code, which is sufficient for this initial release, but prohibitive to users outside the project development team. The next release will incorporate a web-based interface that administrators of services can access securely, and make modifications to the service configuration. These modifications include defining and parameterise query and result pipelines, as well as the search engine and levels of logging to use.
- **Different profiles for the REST interface**:
    - Currently, the query filters, result filters and search engine are fixed for users of the REST interface. In the future, it will be possible to choose between different profiles for a search, each one with a different set of filters. This will be linked to the web-based service configuration feature, mentioned above.
- **Investigate User Profile Support:**
    - Whilst it is difficult to anticipate the needs of a service in terms of user profiles, nevertheless, we shall investigate integrating a simple user profile management into Puppy using Django's built-in user account features. This may be an optional feature that has to be enabled, rather than on by default.
- **Service Logging:**
    - A more advanced logging module, customizable and using different files for queries, results and other relevant data. Furthermore, logging information sent from client devices will be investigated, to capture relevant interaction data, such as click-through actions.
- **Improved Search:**
    - Currently the results are retrieved page by page, limited by the maximum number of results a search engine provides. A better and more flexible solution is to allow arbitrary amounts of results to be requested, adopting an iterator-based approach (see: SuSe prototype source code for a similar idea)
- **Search wrappers for Terrier and Lemur**
    - Most of the framework wrappers have integrated REST-ful search web services (e.g. Bing), which return results as either XML or JSON for HTTP requests. However, some users of the framework may wish to use alternative search engines that do not provide the same interface.

6 Summary

This report has provided an overview of the framework development status to accompany the release of the first version of the PuppyIR framework. Whilst several core components have had their initial implementation, the framework is still developing. Significant known issues have been reported and together with the future plans, will feed into the development of the second release of the framework. The next deliverable, **D4.5 – Report on Specification and Design Changes** will report the progress made on these implementation improvements ahead of the second release of the framework.

References

[1] Annex 1: Description of Work, PuppyIR, 2008

[2] D1.2 – Agreed User Requirements and Scenarios, PuppyIR, 2009

[3] D1.3 – Agreed Technical Requirements, PuppyIR, 2009

[4] D3.1 – Report on Data Pre-processing, PuppyIR, 2009

[5] D4.1 – Specification Report, PuppyIR, 2010

[6] D4.2 – Design Report, PuppyIR, 2010

[7] D4.3 – Report on Implementation and Documentation, PuppyIR, 2010

[8] Python (Programming Language): http://www.python.org

[9] Django Project (Web Application Framework): http://www.djangoproject.com/

[10] Django-REST Interface: http://code.google.com/p/django-rest-interface/

[11] Beautiful Soup (Python library for XML): http://www.crummy.com/software/BeautifulSoup/

[12] lxml (Python library for XML): http://codespeak.net/lxml/

[13] jQuery (AJAX Library): http://jquery.com

[14] MT4J (Multitouch for Java Framework): http://www.mt4j.org