European Commission Seventh Framework Programme
Call: FP7-ICT-2007-1, Activity: ICT-1-4.1
Contract No: 216267

Integrated Prototypes Progress Report V3

Deliverable No: D6.11

Version 1.0

Editor:	EA
Work Package:	WP6
Status:	Final
Date:	M36
Dissemination Level:	PU

Project Overview

Project Name: LiWA – Living Web Archives

Call Identifier: FP7-ICT-2007-1

Activity Code: ICT-1-4.1

Contract No: 216267

Partners:

1. Coordinator: Universität Hannover, L3S Research Center, Germany
2. European Archive Foundation (EA), Netherlands
3. Max-Planck-Institut für Informatik (MPG), Germany
4. Computer and Automation Research Institute, Hungarian Academy of Sciences (MTA SZTAKI), Hungary
5. Stichting Nederlands Instituut voor Beeld en Geluid (BeG), Netherlands
6. Hanzo Archives Limited (HANZO), United Kingdom
7. National Library of the Czech Republic (NLP), CZ
8. Moravian Library (MZK), CZ

Document Control

Title: Integrated Prototypes Progress Report – Version 3

Author/Editor: Radu Pop (EA),
Mark Williamson (HANZO)

Document History

Version	Date	Author/Editor	Description/Comments
0.1	Jan 06, 2011	Radu Pop	Outline
0.2	Jan 20, 2011	Mark Williamson	Hanzo Web prototype
1.0	Jan 31, 2011	Radu Pop	Update and final version

Legal Notices

Table of Contents

1. Introduction 4
2. Integrated prototype for European Archive V2 5
2.1. Updates on the Rich Media Capturing Module 5
2.2. Updates on the Temporal Coherence Module 6
2.3. Integration of the Spam filter 8
2.4. Integration of the Query Reformulation module 10
3. Integrated prototype for Hanzo Web platform 12
3.1. Rich Media Capturing Module 12
3.2. Link Extractor Module 13
3.3. Adversarial Module 13
3.4. Crawl-time Temporal Coherence Module 13
3.5. Post Processing 13

1. Introduction

This document describes the last version of the integrated prototype for the European Archive platform, as well as the integrated prototype prepared for the Hanzo platform (as required for D6.9). It presents the integration status for different components developed by the technical work packages in LiWA (Wp2 – Wp5) and focuses on the integration of LiWA components that have been selected for the last version of the prototypes.

The test-bed description and building configuration were presented in the first version of this report (D6.3). The second version of the report (D6.7) described the integration prototypes that become ready in month 23 of the project (see D6.6 and M6.1). This document focuses on the integration activities that were carried on in the third year of the project.

Section 2 represents an update on the last LiWA components integrated with the European Archive platform, for the last version of the prototype: complementary scripts developed for the Rich Media Capture, updates on the Temporal Coherence analysis, the integration of the spam filter module and the query expansion. Section 3 presents the LiWA modules that have been selected for the Hanzo platform.

The Hanzo architecture differs slightly from the European Archive architecture and this document describes the ways in which the various components integrate into that architecture. Note that since the Hanzo Archicture is proprietary, the configuration steps are not described.

2. Integrated prototype for European Archive V2

This section describes the list of the LiWA modules that have been updated or added to the European Archive's platform for the final version of the integrated prototype.

We describe in the following a brief user guide for each component, as well as several implementation updates since their latest released version.

2.1. Updates on the Rich Media Capturing Module

The module was used for capturing the video content for the streaming application. Based on a crawl engineering analysis, several updates were necessary for handling the particularities of the selected Web sites.

The main issue consisted in the detection of the real URLs of the video files. The complexity of this stage is given by the number of hops to perform, before reaching the real URLs. They are embedded in XML configuration files dynamically loaded by the player.

We briefly mention here the additional steps involved in the analysis of the www.pauwenwittemanblog.nl example:

- the html page containing the player use the following URL pattern:

http://pauwenwitteman.vara.nl/Archief-detail.113.0.html?&tx_ttnews[tt_news]=17307&tx_ttnews[backPid]=116&cHash=c9153b3923

- inside the html content we identify the player parameters:

so.addParam("flashvars","config=Archief-detail.113.9010.html...

html%3F%26tx_varaflashplayer_xmlgenerator%5Bconfig%5D%3D6332%26tx_varaflashplayer_xmlgenerator%5Btstamp%5D%3D1279551911%26cHash%3Da05d520f1c

- an intermediary step consists in replacing the special characters: %3F = ?, %26 = &, %5B = [, %5D =], %3D = =
- using this URL, we download the XML configuration file:

http://pauwenwitteman.vara.nl/Archief-detail.113.9010.html?&tx_varaflashplayer_xmlgenerator[config]=6332&tx_varaflashplayer_xmlgenerator[tstamp]=1279551911&cHash=a05d520f1c

- the configuration file redirects to a playlist:

http://pauwenwitteman.vara.nl/index.php?id=113&type=9010&tx_varaflashplayer_xmlgenerator[playlist]=6332&tx_varaflashplayer_xmlgenerator[tstamp]=1279552088&cHash=f7a7d943e8

- finally, the playlist file contains the real URL of the video content:

Error! Hyperlink reference not valid.

All the intermediary steps described in the example above are implemented by an additional python script, which prepares the video URLs for the downloading module.

2.2. Updates on the Temporal Coherence Module

The Temporal Coherence Module was released in open-source on LiWA's community platform. One of the important improvements brought to the deployment of the module relates to the database system. We replaced the initially used Oracle database with a PostgreSQL database, for a more simplified configuration and administration with the European Archive's infrastructure.

We briefly mention here several deployment steps for the latest version of this module:

- **Download the software from the repository:**

```
$ hg clone https://liwa-technologies.googlecode.com/hg/ liwa-technologies
```

Note: If you get an error like: "abort: HTTP Error 404: Not Found", it might be a configuration problem for https. Try instead http:

```
$ hg clone http://liwa-technologies.googlecode.com/hg/ liwa-technologies
```

- **Change to the downloaded directory** and define the **classpath** of Heritrix3 with the libraries to run the software:

```
$ cd liwa-technologies/temporal-coherence/Heritrix3
$ CLASSPATH=`echo $(ls lib/*.jar)":heritrix-3.0.0/bin" | sed -e 's/ /:/g'`
```

- **Start Heritrix3 with the temporal coherence module:**

```
$ java -Xmx512M -XX:+UseConcMarkSweepGC -XX:+CMSClassUnloadingEnabled -
XX:+CMSPermGenSweepingEnabled \
-classpath $CLASSPATH org.liwa.coherence.HeritrixLauncher \
-a password -p 8843 >& heritrix.log
```

This starts the Heritrix crawler, and starts the Web interface to it. One may access the interface with a favorite browser at http://localhost:8443/engine

Note: You might need to check first if the port is not used by another application:

```
$ netstat -l | grep 8443
```

- **Configuring a Job**

The Heritrix3 with coherence module is ready to start crawl jobs. The Heritrix crawler with the coherence module is a extensively configurable software: every crawl job comes with a configuration instructing Heritrix how crawling will be performed. Creating a configuration involves two steps: (i) creating a job from the Web interface with a help of a Web browser, and (ii) changing the parameters of the configuration by editing the configuration file.

- **Create a Job from the Web Interface**

To create a job open the Web interface to Heritrix3 at https://localhost:8443/engine with your favorite Web browser. The Web page lists all the jobs and configuration profiles available. The selective profile is the default configuration profile for the temporal coherence module and will be used to create new jobs. Make a job based on this profile in the following way: (i) click on the selective profile (opens its Web page) and (ii) enter the name of a new job (e.g., test-job in the field copy job to, press button copy. This creates a configuration directory in the software package:

```
liwa-technologies/temporal-coherence/Heritrix3/jobs/test-job
```

- **Configuration of a Job**

Configuration of the jobs is done by editing the file:

```
liwa-technologies/temporal-coherence/Heritrix3/jobs/test-job/crawler-
beans.cxml
```

One may also edit the file in the Web interface (follow the edit link at the beginning of the Web page of the job). The file is a Spring configuration file and all the configuration objects are expressed as Spring beans and their properties.

These three settings comprise the minimum that needs to be changed to configure the crawl:

1. Metadata settings
2. Seeds settings
3. Database settings

Metadata Settings

Two settings needs to be changed in the bean simpleOverrides, for the following properties:

- metadata.operatorContactUrl should point to the contact URL of the crawling operator.
- metadata.jobName must be the same as the name specified when the job was created

For example, one may set the values to the properties in the following way:

```
metadata.operatorContactUrl=http://www.mpi-inf.mpg.de/departments/d5/
metadata.jobName=test-job
```

Seeds

The coherence module of Heritrix crawls inputs the list of sitemaps, and crawls all URLs from each sitemap. The input sitemaps are set in in the sitemaps.textSource.value of the longerOverrides bean. Each sitemap URL must be placed on a separate line. Here is an example:

```
<prop key="seeds.textSource.value">
        # URLS HERE
        http://www.cnn.com/sitemap_index.xml
        http://www.cnn.com/sitemap_static.xml
</prop>
```

Database Settings

The temporal coherence module writes the crawl data into a PostgreSQL database with a specific DB schema. The database schema can be creating using the SQL commands provided in temporal-coherence/sql-scripts/liwa-coherence.sql file.

The connection settings to the PostgreSQL DB are set in the connectionPool bean. One must set the server (name or IP address), the user, the password, and the DB name. The other (four) parameters define the connection and the pooling mechanisms and is safe to leave the default values. Example of the DB configuration is given below:

```
<bean id="connectionPool"
        class="org.liwa.coherence.db.ConnectionPool">
        <property name="serverName" value="serverName" />
        <property name="user" value="user" />
        <property name="password" value="password" />
```

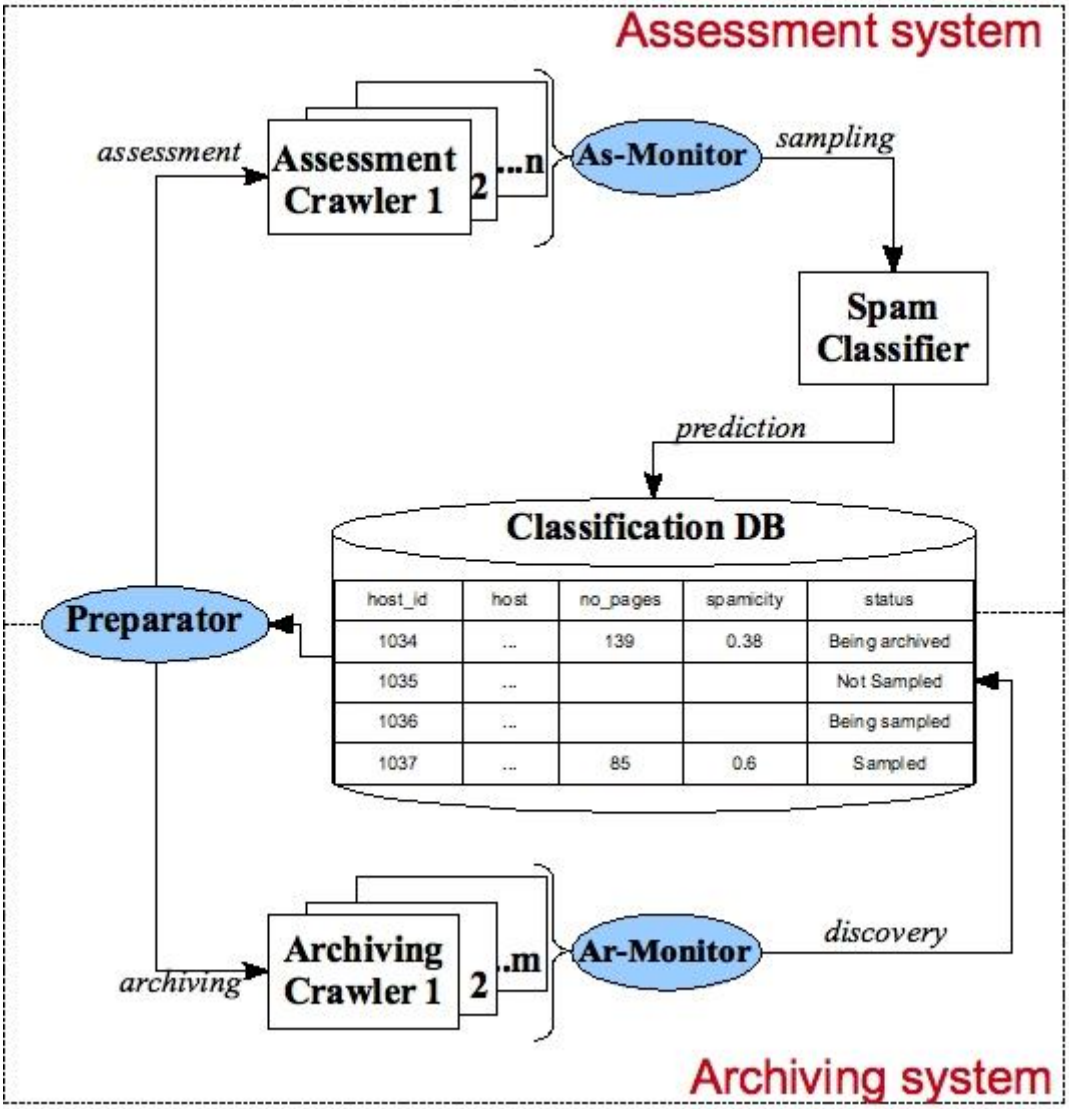
```
        <property name="databaseName" value="liwa_coherence" />
        <property name="dataSourceName" value="Coherence Data Source" />
        <property name="maxConnections" value="200" />
        <property name="dataSourceClass"
value="org.postgresql.ds.PGPoolingDataSource"/>
</bean>
```

2.3. Integration of the Spam filter

We can view the crawling system, with the Spam filter integrated, as two different and independent sub-systems, with one component in common, the database:

- Assessment system
- Archiving system

The general architecture for the integration of the Spam filter is given by the following diagram:



The database consists of a single table that holds informations for each host. The information kept for a host is: the host name, it's spamicity, the number of pages that the spamicity was computed on and the status.

The status of each host may have one of the following values:

- **not sampled** - the host has not passed through the spam module yet
- **being sampled** - the host is in the process of being archived and then passed through the spam module
- **sampled** - the host has been sampled
- **being archived** - the host is being archived (can reach this status only if it's spamicity score is below an imposed threshold)

Each host in the database has to go through the "assessment system". Then, if it meets some conditions, it can be archived.

Assessment system

This system is composed of a series of scripts, the crawlers used to gather the data needed by the spam module, and the spam module itself.

The "**Preparator**" module takes a batch of hosts marked as "not sampled" from the database, changes their status to "being sampled", prepares an order of crawling, and sends it to the crawlers.

The crawlers of this system, called "**assessment crawlers**", are configured to gather the information needed by the spam module (a maximum of 200 text resources for each host). When the crawler is done working, the "**Monitor**" script takes the resulted ARC files, sends them to the spam classifier and then updates the database with the results. At this stage a host will have the "sampled" status.

Archiving system

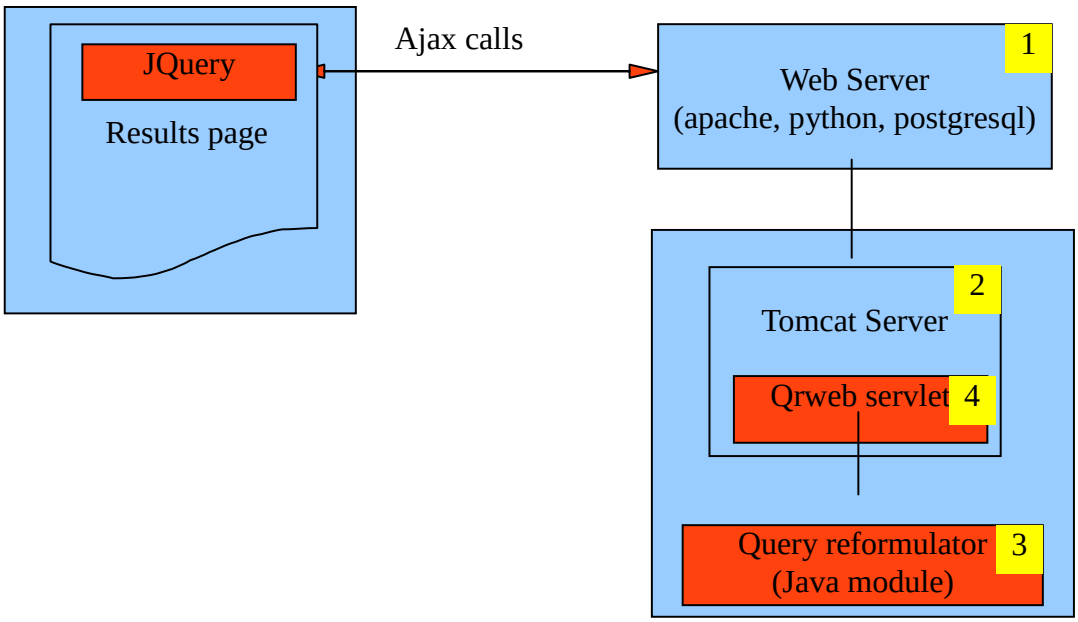
This system is composed of a series of scripts and the crawlers used to do the actual crawling. The "**Preparator**" does the same thing as the one used in the assessment system, with the sole difference that it takes hosts that are marked as "sampled" and have the spamicity below a certain threshold, and changes their status to "being archived".

The crawlers in this system, called "**archiving crawlers**", are standard crawlers configured to stay within a domain. The "**Monitor**" script in this system takes the out-of-domain sites discovered by the crawlers and adds them to the database with the "not sampled" status.

2.4. Integration of the Query Reformulation module

The Query Reformulation module is integrated to the full text search utility provided in the Web user interface of European Archive's platform.
To achieve its integration, the QR module was encapsulated in a Tomcat servlet which runs using a Tomcat server on a separated machine.

The functional overview of the module is depicted in the diagram below:



We briefly mention here several technical details for each component in the diagram:

[1] Web Server:
An Apache Web server running on Debian Lenny 64.

[2] Tomcat Server:
A standard Tomcat 5.5 which comes with Debian Lenny 64 with 4GM of memory and 4x1.6GHz. Java virtual machine that runs the Tomcat server was given all the 4GM of memory.

[3] Query Reformulator:
MPIITE tool, provided as a Java Module. It includes a jdb database which is about 3GB of data.

[4] QR interface.
QR is queried using 4 parameters:

- query string
- reference date
- target date
- k (number of results)

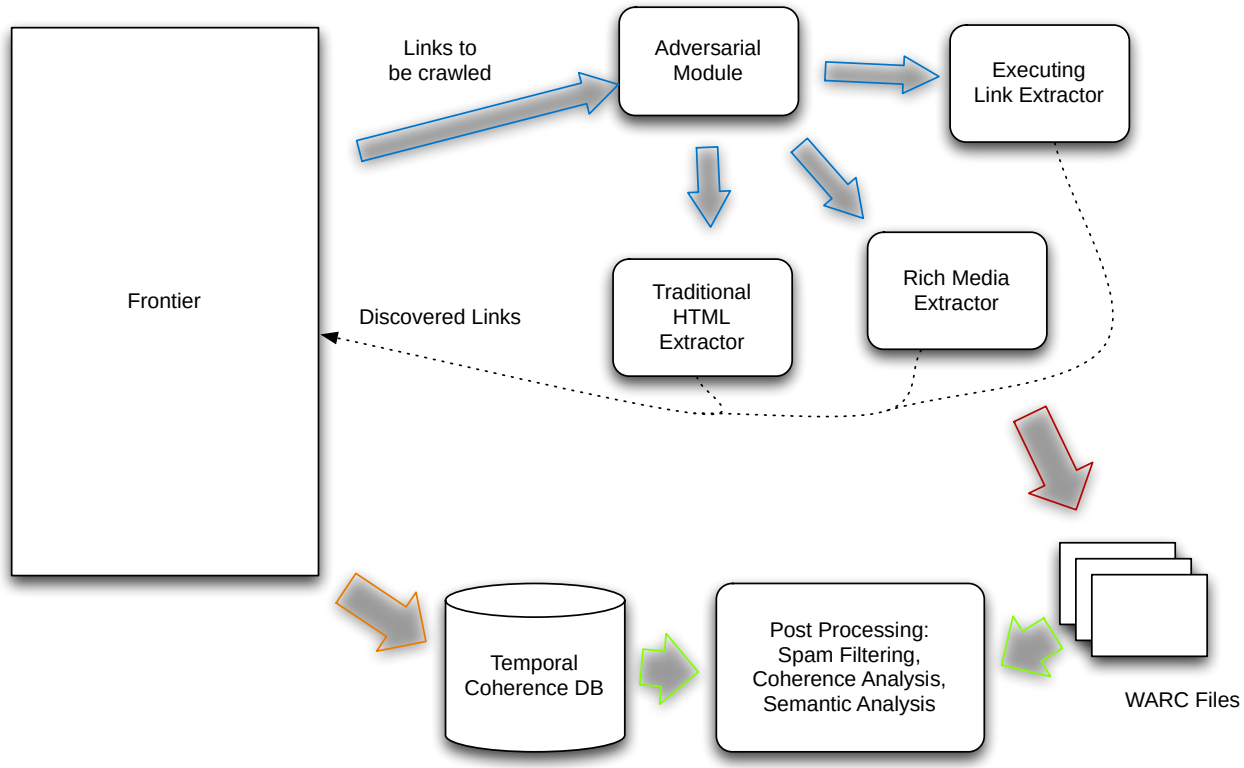
Global remarks:

- QR performs heavy computations, therefore a request could take up to one minute to return (on our server described above).

- QR use its own database which is build against the New York Times Archives. This database includes data going up to year 2007. This is the reason why we did not introduce query with reference-time/target-time greater than 2007.

3. Integrated prototype for Hanzo Web platform

The Hanzo architecture uses different modules to collect different types of content. The adversarial module (from WP2) selects which module is used to collect any given content.



Unlike the integration with Heritirx (see document D6.7) the modules from Work Package 2 reside natively inside the Hanzo archicture. The modules are configured to both download and do link extraction on the target content.

The components are connected together with a messaging system. Messages, which contain URLs to be collected, are taken from the frontier and routed to the different components by the adversarial module which uses rules to select the approriate extractor.

3.1. Rich Media Capturing Module

The principle of the Rich Media Capturing module is to delegate the multimedia content retrieval to an external application (MPlayer or FLVStreamer) which is able to handle a larger spectrum of transfer protocols, such as real time protocols, widely used for video streaming.

The second version of the technology added a new transfer protocol (RTMP) to the list of real time protocols supported by the Rich Media Capturing module: RTSP, MMS and RTMP.

The Executing link extractor and the traditional HTML extractor both look for all suitable URLs and the adversarial module detects these and sends them to the Rich Media module.

In some cases the extraction of these URLs is more complex, for example they need to be determined from configuration files, and the Hanzo architecture also allows for rich media capture to be run as a Post Process with the URL's for capture determined by post processing of the warc files collected during a capture.

3.2. Link Extractor Module

In the Hanzo Architecture the link extraction module is also allowed to collect the material downloaded during the page extraction into warc files. The link extractor is a native module in the crawler application and it receives messages as routed by the adversarial module and delivers discovered links back to the frontier by the normal messaging system.

3.3. Adversarial Module

The Adversarial Module routes URLs for download through the crawler. Rules determine which capture method is most suitable for the target URL.

3.4. Crawl-time Temporal Coherence Module

The integration of the Temporal Coherence module is in two parts.

The first part is unique to the Hanzo crawler and generated the database content required for the second part which is identical with the integration used with Hertirix and is described in document D6.7.

The processes within the frontier generate the information required for the Temporal Coherence database.

Post crawl the database generated is the same as the one generated for Heritrix and can be used to generate the same analysis.

3.5. Post Processing

Post Processing in the Hanzo Architecture is the same as it is within the Heritrix based architecture and for more details refer to D6.7