LiWA
Living Web Archives

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

Report on Integration Strategy, Testing Plan and Test-bed Architecture

Deliverable No: D6.3

Version 1.0

Editor:	EA
Work Package:	WP6
Status:	Draft
Date:	M12
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, Learning Lab Lower Saxony (L3S), 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:	Report on Integration Strategy, Testing Plan and Test-bed Architecture
Author/Editor:	Radu Pop (EA)

Document History

Version	Date	Author/Editor	Description/Comments
0.1	Dec 16, 2008	Radu Pop	Outline
0.2	Jan 16, 2009	Radu Pop	Testbed description
0.3	Jan 30, 2009	Radu Pop	First version
0.4	Feb 12, 2009	Wolf Siberski	Revision
1.0	Feb 12, 2009	Pop	Final Version

Legal Notices

Table of Contents

Introduction

This document describes the implementation of the test bed and integration of the components from technical work packages WP 2 – WP 5 into this test bed. In D6.1 the LiWA strategy for the integration of components, the quality assurance, initial test cases and the overall architecture of the LiWA test bed was presented. The core of the strategy is an integrated development and testing environment. For source code management a SVN based repository made available. This repository is regularly monitored and updated components are automatically built and tested regarding their functionality. This continues building and testing process is coordinated by “CruiseControl”, an open source framework for maintaining build and test processes.

Such a continuous building and testing environment ensures seamless integration of components even if they are changing a lot, which is foreseen for the second project year. However, even if a large number of support tools are available the initial preparation of such an environment is complex and has to be done with care. As the aim of LiWA is to develop in the technical areas new approaches beyond the state of the art, it is not always possible to use well tested tools and libraries as a foundation. Also some components like the archive coherence WP4 are deeply integrated into the Heritrix crawler. Therefore special attention has to be given to the building and testing scripts.

After the first project year we have the whole building and testing infrastructure for LiWA in place, located on a powerful machine at European Archive. All technical components - as far as they are existing - are building and compiling. Due to the tight coupling of the archive coherence with the Heritrix crawler, just this component is currently integrated. Due to the expected advancements in all technical areas in the second project year the full integration will be done till month 23 (s. Deliverable D6.6). Detailed descriptions about the approaches and initial evaluation results will already be published in month 18 (s. Deliverable D6.5).

The achievements regarding the development infrastructure and test bed are documented in this report. It first describes the test-bed framework in section I. Afterwards the currently available test data sets for components test are introduced. Finally a brief description for the integration of the individual components into the test bed is given.

I. Testbed Framework

1. Automated Test Approach

The integrated testing framework set up at European Archive realizes the approach described in D6.1 (Report on Integration Strategy, Testing Plan, and Test-bed Architecture). The infrastructure allows automatic building of the current prototype software and running tests to check its correct function. This process is controlled by the software 'CruiseControl' (see Section 2.2), an open source framework for maintaining build and test processes.

As described in D6.1, LiWA consists of components active during a crawl and postprocessing components. As test environment for the former, the Heritrix crawler is deployed and configured (Section I.2.1**Error! Reference source not found.**). The postprocessing tasks are triggered by separate CruiseControl project configurations.

2. Software Components

2.1. Heritrix Crawler Deployment

The deployment and installation of Heritrix is fairly easy, as clearly explained also on the project's Web site (http://crawler.archive.org/articles/user_manual/install.html). However, we wrote additional scripts that allow automatically launching the crawler with a given set of seeds.

Launching the crawler has to be done from the command line, with the set of parameters prepared in the test suite. The role of the scripts is to be included as part of the automatic tests managed by the Cruise Control System. Moreover, we planned to run in parallel the two versions of Heritrix (the original one and the LiWA updated version). Having these two versions available at the same time allows to easily compare the original system with the new LiWA technology. The additional scripts will therefore help in dealing with the two crawling sessions, using the same list of seeds.

After the two crawls are finished, the results can be compared either by comparing the generated log files or by using for example the Temporal Coherence module in visualising the quality of the crawled content.

Another important adjustment in Heritrix launcher concerns the parameter setting the size of the memory heap to be used, increased to 1GB (**JAVA_OPTS="-Xmx1024m"**).

2.2. Cruise Control System for building and testing LiWA source code

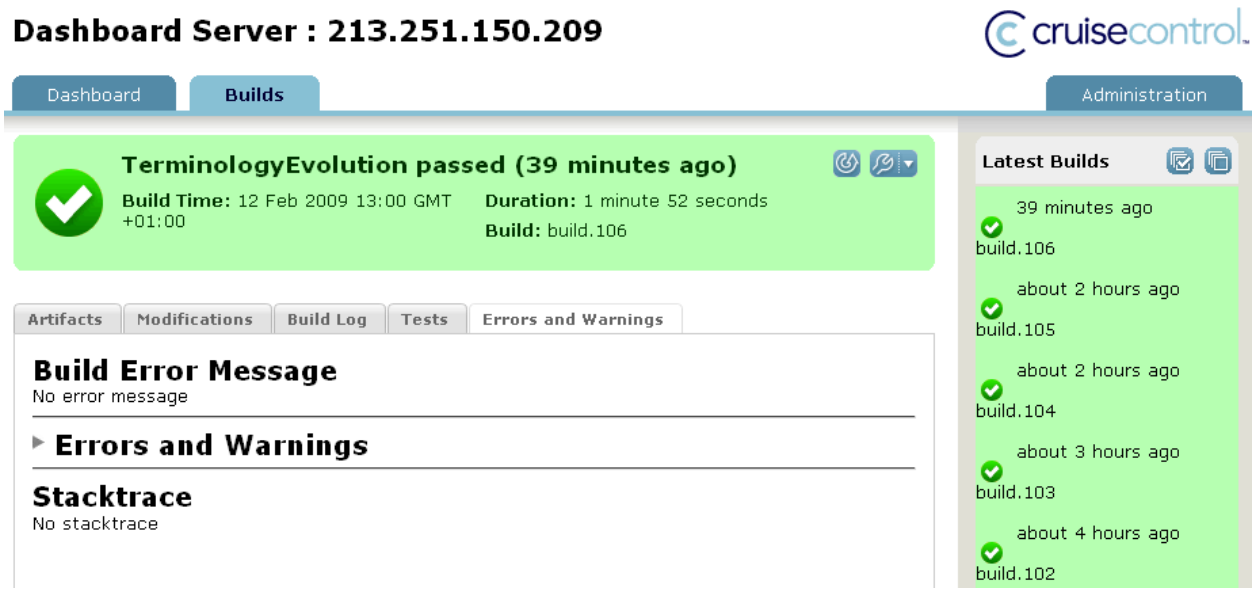

Figure 1: LiWA CruiseControl - Building and testing status of the Terminology Evolution Component

Following the plan stated in the Deliverable D6.1, we set up on the testing machine a Cruise Control system for builds and automatic unit tests of LiWA modules.

The Cruise Control system is accessible at:

http://debug.europarchive.org:8081/dashboard/

Cruise Control provides a plug-in that directly supports SVN repositories and it connects by https to the LiWA SVN.

We defined and configured the build scripts (Ant files for Java packages, Make files for C code, etc.) for each module.

In the first round of tests, the builds of the modules are triggered manually, in order to ensure first a correct configuration of the build scripts. In the second phase, the builds of the source code are triggered automatically by the updates performed on the SVN.
The JUnit tests are implemented incrementally for each LiWA module and they are committed together with the source code builds.

2.3. Updated Repository Structure

LiWA uses Subversion as its file repository systems. The access address is

https://svn.l3s.uni-hannover.de/liwa

While setting up the test and build environment it came clear that the proposed structure in the repository wasn't suitable for our purposes. Therefore we enhanced the basic structure as shown in Figure 2.

```
branches
tags
trunk
        source
                apps
                        coherence
                        streaming
                        socialweb
                        evolbrowser
                        spam
                crawler
                        coherence
                        heritrix
                        semantics
                        spam
                        warc
        documents
                deliverables
                papers
                WPx
```

Figure 2: Project Directory Hierarchy

2.4. Hardware

We dedicate a testing machine for LiWA project with the following main characteristics:

- **3.0 GB RAM** (extensible at 8.0 GB)

The main concern related to the RAM memory mainly involves the Heritrix crawler. Since Heritrix software has a Java based implementation and most of its modules (Frontier object, for instance) use memory loaded objects, the memory availability becomes an important issue, especially for the large crawls.

Moreover, we plan to run our tests in two different flavours of Heritrix inside this framework, as we previously mentioned. Both scenarios will use the same datasets: the same lists of seeds for crawling and eventually the same sets of WARC files in the post-processing phase.

We therefore estimate that for the first round of tests, the 3GB of memory would hold for running in parallel two instances of Heritrix (using two distinct Java Virtual Machines), allocating a maximum of 1GB heap memory for each instance.

2 x 400 GB storage disks (extensible at 2 x 1.0 TB)

A preliminary estimation for the storage space required for the tests includes a minimum of around 400 GB, the size of the data corpus used by the spam filter engine.

According to the data size resulting from the first round of tests, some additional storage may be externally mounted on request.

- **Debian Stable 4.0** operating system

This is a widely used Linux distribution and from the current requirements of the LiWA external modules this distribution seems to provide all the necessary tools and packages (for instance, several external libraries needed by the spam filter engine).

II. Test Data Sets

3. UK Web Snapshots 2006/2007

The data set that we are going to use in the first round of tests is represented by 13 snapshots of the UK domain used by the Spam Filter Engine to compile and extract the features.

It consists of a collection of around 500 GB of WARC files with the last versions of the snapshots, downloaded by Wp3 team and copied then on the LiWA testbed.

Each module uses a locally different data set for testing, according to each module requirements. After the first round of tests on the integration testbed, a unified version of the data set will be defined.

A second set of data that is ready to be used for the tests is is a GOV.UK collection hosted by European Archive. This collection contains a list of around 1000 UK government Web sites regularly crawled since 2006 with different crawl frequencies.

A subset of this collection has to be defined for each module, according to the specific requirements (type of content to be watched, size, history) and testing goals.

III. Components

4. Rich Media Crawling

The first module of Wp2 is being integrated with the crawler is a first version of the Rich Media Capture module.

The module is constructed as an external plug-in for the crawler (Heritrix). Using this approach, the identification and retrieval of streams is completely decoupled, allowing to use more efficient tools to analyze video and audio content and, at the same time, placing a lighter burden on the crawling process.

In the current architecture, the external module, which is coupled to the crawler, intercepts all requests going to streaming resources (normally the crawler would reject them as it does not know how to process them) and will pass them to a cluster of computers running the stream capturing tool.

Based on some performance consideration, we decided to use a standardized communication protocol between the subcomponents of the module, the *Advanced Message Queuing Protocol* (AMQP). Using this messaging architecture allows an architecture that scales with the load, as it increases and decreases dynamically the size of the cluster used to capture streaming resources.

This also allows to integrate new layers into our module. As an example, at a later point in the project the demand for a layer transforming all video and audio resources captured to some common predefined codecs might arise, in order to provide access through an unified interface.

5. Spam Detection

The first version of the Spam Filtering Engine still contains a rather long list of dependencies to different external libraries. Wp3 team is working on releasing a “lighter” version of the engine, but for the first integration tests we ensured a proper installation of the external libraries, when needed.

The list of the needed libraries is given in the table below:

Library	Category	Description
apache2-utils	net	utility programs for webservers
cron	admin	management of regular background processing
db4.2-util	utils	Berkeley v4.2 Database Utilities
fontconfig	utils	generic font configuration library
gawk	interpreters	GNU awk, a pattern scanning and processing language
libkrb53	libs	MIT Kerberos runtime libraries

libldap2	libs	OpenLDAP libraries
libltdl3	libs	A system independent dlopen wrapper for GNU libtool
libpng12-0	libs	PNG library - runtime
msttcorefonts	contrib/x11	Installer for Microsoft TrueType core fonts
openoffice.org	editors	high-quality office productivity suite
openoffice.org-bin	editors	OpenOffice.org office suite binary files
wget	web	retrieves files from the web
x-ttcidfont-conf	x11	Configure TrueType and CID fonts for X
xfonts-base	x11	standard fonts for X
xvfb	x11	virtual framebuffer X server

Another tool needed by the Spam Filtering module is the WEKA learning machine. WEKA (Waikato Environment for Knowledge Analysis) is a popular suite of machine learning software written in Java, developed at the University of Waikato.

The WEKA workbench contains a collection of algorithms for data analysis and predictive modelling, together with graphical user interfaces for easy access to this functionality.

6. Temporal Coherence

The integration of the Time Coherence module implies the interaction with the testbed framework at two different levels:

- an internal integration of LiWA specific code to the official version of Heritrix, and
- an external integration of the LiWA Processor classes that analyse the crawl database

For the internal integration, the following Heritrix classes have been modified:

- *org.archive.modules.fetcher.FetchHTTP*

  Description: given a URL to fetch, it reads the ETAG from the database and adds the If-None-Match HTTP header

  org.archive.net.UURIFactory.

  Description: class used for the URL normalization

  We identified problems with URL normalization, URLs containing the character ~ were not detected to be the same with URLs containing %7E, so this normalization had to be fixed.

- *org.archive.modules.writer.DefaultMetadataProvider*

  Description: the class was modified to store attributes which are used in other classes

These modified Heritrix classes for the temporal coherence analysis are committed in the LiWA SVN Repository under a distinct branch of the Heritrix code, corresponding to the LiWA updated version of the crawler.

For the external integration of the LiWA processor classes:

1. they are stored in a distinct package: ***de.mpg.mpii.liwa.****
2. LiWA Processor stores the URLs in the database
3. *DbSeetModuleImpl* is a class that implements the seed management

Some other existing classes, not part of Heritrix yet, are also used by the Time Coherence analyser for the:

- time coherence analysis
- graphML generation

In order to ensure the integration of the module we provided access to a running Oracle database. The schema of the database is created by a set of SQL scripts, which were tested on the Oracle installation of the testbed.

The Oracle database is accessible by the Web interface at:

http://debug.europarchive.org:8080/apex

We created a LiWA Tablespace with an initial 500 MB space, some extensions of the allocated tablespace could be reconfigured later on.

7. Semantic Evolution

The current requirements for the Terminology Extractor module concern the framework provided by Apache UIMA. The Unstructured Information Management application is a software system that analyzes large volumes of unstructured information in order to discover knowledge that is relevant to an end user.

UIMA enables applications to be decomposed into components, for example "language identification" => "language specific segmentation" => "sentence boundary detection" => "entity detection (person/place names etc.)". Each component implements interfaces defined by the framework and provides self-describing metadata via XML descriptor files. The framework manages these components and the data flow between them.

The UIMA infrastructure includes a simple server that receives REST requests and returns annotation results, for use by other work packages.

For the integration tests of the Terminology Extractor module, the UIMA SDK was installed on the testbed server. As defined in D6.1 a junit test was designed to verify the result of the extraction on a defined collection of documents.

8. Bibliography

[1] D6.1 Report on Integration Strategy, Testing Plan and Test-bed Architecture

[2] CruiseControl. http://cruisecontrol.sourceforge.net/