



# Quality Assurance Workflow, Release 1


## Release Report

### Authors

Denis Pitzalis, Matthieu Cord, Stéphane Gancarski (Université Pierre et Marie Curie), Peter May, Andrew Jackson (The British Library), Ivan Vujic (Microsoft Research), Johan van der Knijff (National Library of the Netherlands), Roman Graf, Reinhold Huber-Mörk (AIT Austrian Institute of Technology), Leila Medjkoune (Internet Memory Foundation), Bolette A. Jurik (State & University Library Denmark)

March 2012

*This work was partially supported by the SCAPE Project. The SCAPE project is co-funded by the European Union under FP7 ICT-2009.4.1 (Grant Agreement number 270137).*

*This work is licensed under a CC-BY-SA International License* 

## Document Approval

Person	Role	Partner
Matthieu Cord	WP lead	UPMC
Miguel Ferreira	SP lead	KEEPS

## Distribution

Person	Role	Partner
Matthieu Cord	Reviewing and feedback	UPMC
Stéphane Gancarski	Reviewing and feedback	UPMC
Miguel Ferreira	Reviewing and feedback	KEEPS

## Revision History

Vers.	Status	Author	Date	Changes
0.1	Initial version	Denis Pitzalis	31 Jan 2012	Outline and initial stub created.
0.2	Update	Roman Graf, AIT	16 Feb 2012	Various updates
0.2	Update	Bolette Jurik, SB	16 Feb 2012	Various updates
0.2	Update	Peter May, BL	16 Feb 2012	Various updates
0.2	Update	Ivan Vujic, MSR	16 Feb 2012	Various updates
0.2	Update	Johan van der Knijff, KB	16 Feb 2012	Various updates
0.3	Update	Reinhold Huber-Mörk, AIT	24 Feb 2012	Various updates
0.3	Update	Leila Medjkoune, IM	24 Feb 2012	Various updates
0.4	Update	Denis Pitzalis, UPMC	28 Feb 2012	Cleaned up and various updates
0.4	Update	Bolette Jurik, SB	28 Feb 2012	Various updates
0.4	Update	Peter May, BL	04 Mar 2012	Various updates
0.4	Update	Ivan Vujic, MSR	04 Mar 2012	Various updates
0.5	Update	Denis Pitzalis, UPMC	05 Mar 2012	Various updates
0.5	Update	Bolette Jurik, SB	06 Mar 2012	Various updates
0.6	Update	Denis Pitzalis, UPMC	09 Mar 2012	Cleaned up and various updates
0.6	Update	Bolette Jurik, SB	12 Mar 2012	Various updates
1.0	Update	Miguel Ferreira, Keeps	22 Mar 2012	Proof reading
1.0	Update	Pitzalis Denis, UPMC	22 Mar 2012	Final Reading

## Table of Contents

Table of Contents.....	4
1 Introduction.....	6
2 Background on Quality Assurance.....	7
2.1 A Migration Example.....	7
2.2 Methodology of work.....	7
3 Cross-media Taverna integration.....	9

4	Audio QA workflows.....	12
4.1	Initial problem and prototype solution.....	13
4.1.1	Signal Processing.....	13
4.2	Taverna workflows.....	14
4.2.1	Workflow for FFmpeg mp3 to wav Migration.....	16
4.2.2	Workflow for FFprobe Property Extraction.....	16
4.2.3	Workflow for WAV File Format Validation using JHOVE2.....	17
4.2.4	Workflow for Migration, Validation, Feature Extraction, Comparison.....	18
4.2.5	Workflow for migration QA Sound Comparison Tool.....	19
4.3	Benchmarks and Validation Tests.....	20
4.3.1	Mp3 to Wav Migrate Validate Compare Workflow Run Times.....	20
4.3.2	migrationQA SCAPE WS Wav File Comparison Workflow.....	21
4.4	Audio Property Extractor, Comparator and Format Evaluation.....	21
5	Web QA workflows.....	22
5.1	Scenarios description.....	22
5.2	Taverna workflow.....	23
5.3	Tools and Interfaces.....	25
5.3.1	Conversion.....	26
5.3.2	Feature Extraction.....	26
5.3.3	Similarity.....	26
5.3.4	Training and evaluation.....	27
6	Document QA workflows.....	29
6.1	Design of Feature Probes.....	29
6.2	Experiment Design for Evaluating Feature Probes.....	29
6.3	Document Representations and Analysis.....	30
6.3.1	Analysis of Document Rendering using XPS Representation .....	31
6.3.2	Analysis of PDF Rendering using Image Representations.....	32
6.3.3	Remarks on the rendering and QA algorithms.....	34
6.4	Structure Representation and Comparisons.....	34
6.5	Cloud Solution for Storage and Computation.....	34
6.5.1	Tools and Program Interfaces in MS Cloud.....	34
6.5.2	UI for the Cloud Based Document Conversion and Analysis.....	36
7	Image and QA tools.....	40
7.1	Imaging Tools.....	40
7.1.1	DPQAlib Library.....	41
7.1.2	Example Taverna Workflow.....	44
7.2	"QA Tool" Analysis Tool.....	46

7.2.1 Bitwiser.....	46
7.2.2 Example Taverna Workflow.....	47
7.2.3 Status.....	47
8 Knowledge base.....	48
9 Roadmap for Next Iteration and conclusions.....	49
10 Bibliography.....	50
Appendix A: External Tools.....	52
Appendix B: Description of tools in MS working environment.....	55
Appendix C: Description of tools in the development environment.....	57

## 1 Introduction

The objective of the PC.WP3 is to provide automated, scalable methods for quality assurance, applicable to a range of preservation workflows that are aimed at ensuring long term access to digital content, as exemplified by the SCAPE Test bed scenarios.

As per the description of work, this work package has been split into six top-level tasks. Task 1 aims at producing quality assurance workflows that will integrate the tools developed in tasks 2, 3 and 4.

Task 2 will output a range of tools capable of converting objects of various media types into a temporary format capable of being analysed and compared with another instance of that object, i.e. before and after any preservation action.

Task 3 is intended to select and integrate appropriate analysis tools to measure the quality into the workflows defined in task 1.

Task 4 provides new algorithms to quantifiably compare the converted objects.

Task 5 will examine the possibility of automatically repairing errors that can appear during migration.

Task 6 consists of validating and assessing conversion tools and processes.

In the past, large-scale quality assurance has relied on a combination of human intervention and statistical methods. In order to meet the challenges posed by large and heterogeneous digital collections, we will research and develop automated quality assurance approaches.

This involves designing tools and methods to measure different properties of digital objects. Furthermore, comparison of digital objects often requires transforming digital objects into a common intermediary format that can be used for a specific analysis. By aggregating measurements from different methods we expect to produce rich characterization of digital objects and effective metrics from their comparison.

Overall, PC.WP3 is concerned with specific aspects:

- A framework for connecting preservation goals with the preservation characterization actions (coordinated with the WP1 – Preservation planning). The comparison and coordination of the goals and preservation outcomes will be facilitated by quantitative evaluations, based on specified metrics.
- Interfaces among three different types of tools that comprise the digital object migration workflows: conversion tools, analysis tools, and comparison tools.
- Application of existing analysis tools (e.g. image OCR, layout analysis, audio conversion) to extract significant properties. Development of additional tools, such as conversion tools for various media digital object, video, web content, scientific analysis tools and emulators.
- Methods for comparing significant properties using existing tools. This includes development of tools and methods that provide quantitative confidence values for QA measurements in order to quantify their reliability and improve their accuracy.
- Development of methods for automatic repair when migration errors are detected.
- Creation of a knowledge base of characterization and migration errors.

Furthermore PC.WP3 is working side by side with the PW sub-project to agree on interfaces to deliver QA measures to the planning and the watch components for assessment and successive reaction and feedbacks.

## 2 Background on Quality Assurance

Quality assurance is an essential aspect of preservation methodology. Its objective is to assess

whether specific preservation goals are met. SCAPE focuses its interest in preservation goals that are directly related to the characteristics of the digital objects. For example, goals may stipulate that:

- A set of documents needs to be migrated from format A to format B so that the original documents, created by the software application SA, can be viewed by the application SB. For example, documents created by MS Word 1997 need to be migrated to HTML or Open Office format.
- All the hyperlinks and cross-references within the documents need to be accessible by a simple single-click access, as in the original document format. Hyperlinks are used to facilitate navigation within the document and access to the resources on the Web.
- The reliability of hyperlink detection needs to be no less than 99% for within-document links and 95% for the external links.
- Acquired files must conform to an agreed technical profile, be valid and complete

## 2.1 A Migration Example

The outcome of migration is, to a large degree, a function of the specific conversion tool used, the properties of the target document format, and the computing environment in which the transformation is facilitated and the target format used.

Quality assurance quantification means both:

- Assess of the conversion tools in general
- Evaluate the application of the conversion tools within a particular, real, context.

Assessment of the conversion tools is typically performed on known data in order to obtain *reliability estimations* for specific aspects of the tool. Some of these aspects are operational, such as speed of conversion. Others apply to the characteristics of the conversion output ranging from general aspects, such as file size of the target format documents, to specific characteristics such as font size of the section titles in the document.

Features that are related to the digital object themselves require comparison of the original and target documents. In essence, we need to determine how successful the mapping between them is, as by the format transformation tool.

## 2.2 Methodology of work

During the kick off meeting held in Vienna, leaders of work package 11, along with sub-project Quality Assurance leader and project leader, decided to organize the work-package in 7 working groups oriented to formats. All the formats sharing the same development strategies but with different approaches. This organization has enabled an horizontal communication between all the work-package tasks.

The structure of the deliverable follows the one of the working-groups:

- Cross-media taverna integration, described in chapter 3;
- Audio QA workflows, described in chapter 4;
- Web QA workflows, described in chapter 5;
- Document QA workflows, described in chapter 6;
- Image QA workflows, described in chapter 7.1;
- Tools QA workflows, described in chapter 7.2;
- Knowledge base, described in chapter 8.

### 3 Cross-media Taverna integration

The code for creating a command line tool wrapper for SCAPE QA tools is located in the repository GitHub [xatoolwrapper]. The Tool Wrapper is a common basis for developing tools in QA work package. It is a Java based application for creating a web service wrapper project for command line tools. Based on an XML tool description (different for each service), the application creates a maven project for the tool which includes web services and web service operations allowing to execute a command line tool via SOAP, RESTless, and Java API interfaces. For installation and deployment information a README file containing the instruction is provided while the a more detailed description of Taverna workflow system can be find at [taverna].

To describe a command line tool as a web service XML description files are used. Such a file comprises service name and package name, supported operations. It contains deployment information like host IP and ports.

Here is a sample operation description for image comparison tool with two inputs and one output parameter. The input parameters for the compare operation are of type URI and are required. The output is of XML type:

```
<operation oid="2" name="compare">
  <description>Compares image file features</description>
  <command>compare.bat ${input1} ${input2} ${output}</command>
  <inputs>
    <input name="input1">
      <Datatype>xsd:anyURI</Datatype>
      <Required>>true</Required>
      <Default>http://localhost:8080/testbild1.xml</Default>
      <CliMapping>input1</CliMapping>
      <Documentation>URL reference to input file</Documentation>
    </input>
    <input name="input2">
      <Datatype>xsd:anyURI</Datatype>
      <Required>>true</Required>
      <Default>http://localhost:8080/testbild1c.xml</Default>
      <CliMapping>input2</CliMapping>
      <Documentation>URL reference to input file</Documentation>
    </input>
  </inputs>
  <outputs>
    <output name="output">
      <Datatype>xsd:anyURI</Datatype>
      <CliMapping>output</CliMapping>
    </output>
  </outputs>
</operation>
```

```

    <Required>>false</Required>
    <Documentation>URL reference to output file</Documentation>
    <PrefixFromInput>input1</PrefixFromInput>
    <Extension>xml</Extension>
  </output>
</outputs>
</operation>

```

If necessary, customized batch file can be created. For instance if Taverna doesn't support required functionality like redirect output we created an example file called compare.bat file that can be used for redirection:

```

@ECHO OFF

REM *** Redirect compare service output to the passed parameter 3

ECHO Argument 1: %1
ECHO Argument 2: %2
ECHO Argument 3: %3

IF EXIST %1 ECHO 1 is OK
IF EXIST %2 ECHO 2 is OK
REM *** Kick the tool
compare.exe %1 %2 > %3
ECHO AAAA: %ERRORLEVEL%
IF ERRORLEVEL 1 GOTO EXCEPTION
EXIT %ERRORLEVEL%

REM -----
:EXCEPTION
ECHO !!!! EXCEPTION !!!!
EXIT %ERRORLEVEL%

```

The XML tool description files and associated BAT files are stored under <https://github.com/openplanets/scape/tree/master/pc-qa-toolwrapper>. For each tool should be created a directory which name includes the version of the tool. Big binary files should be downloaded separately using download button in GitHub.

The generated web service can then be added to Taverna workflow using common service adding procedure and generated service URI. Tomcat should be running at that moment. Such a service can be used for Taverna workflow creation.



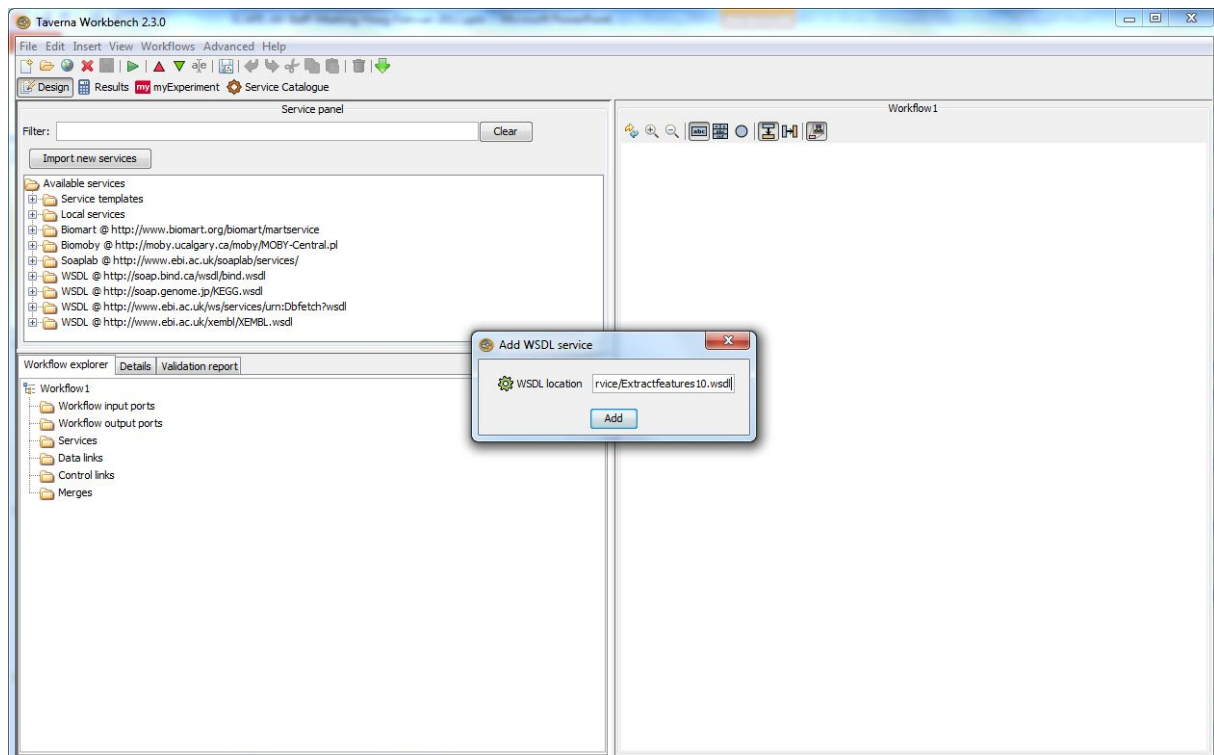


Figure 1: Add command line tool WSDL web service to Taverna

The resulting Taverna workflow are usually stored on the myExperiment platform under the SCAPE group.

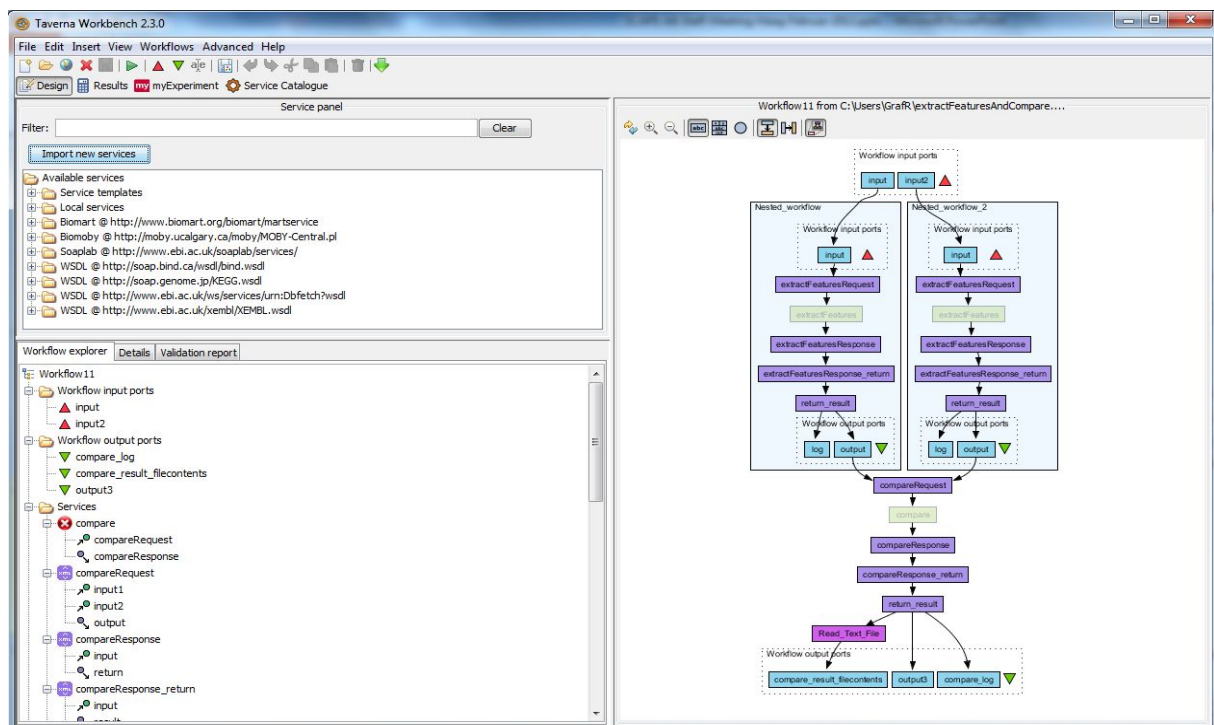


Figure 2: Use command line tool WSDL web service in Taverna Workflow

## 4 Audio QA workflows

Automated quality assurance (QA) is required for preservation actions when dealing with large scale operations, because the alternative, manual, QA processes are very demanding in terms of human resources. People are also liable to make mistakes, especially under stressed conditions, and so automated QA is looked upon to efficiently improve both the correctness of results and the performance. An example of a preservation process that requires QA is the conversion process between different audio formats. Although many quite established tools exist in the open source domain, none of these can ever be proved to be 100% free of bugs. Thus tools can fail unfortunately, and potentially they can fail in a way that users do not notice. QA is therefore especially important when dealing with large scale collections of millions of files where tools will have to run in a distributed, unattended environment.

In this chapter we will describe how SCAPE is intending to deal with quality control tools for audio file formats. Previous projects (including the PLANETS project) have been partly working in this area with the development of technologies like XC\*L [XC\*L] that is capable of extracting features of different file formats and comparing those – e.g. simple comparison of the length of two audio files to check if both the original and the migrated file is of the exact same length.

Certain error scenarios will not be detected by approaches like this however. Consider a simple example of migrating an mp3 file to a wav file for long term preservation. This is the *LSDRT6 Migrate mp3 to wav* scenario [LSDRT6]. If the mp3-decoder fails half way through then the resulting wav file may potentially end up being the exact same length as the original mp3 file but the second half is empty (has no-sound).

Such scenarios imply that other algorithms, perhaps looking more directly at the content rather than "just" extracting simple features, must be developed.

We have defined 3 levels of abstraction for features in audio QA, much like the image driven scenarios described in chapter 7.1:

- Level 1: Audio metadata/attributes that can easily be retrieved from an audio file (e.g. number of channels, length). This level is, in many cases, available directly in the metadata of many audio file formats. Differences in Level 1 features will reveal serious errors in migrations. Level 1 features are meaningful for a file by themselves, and are also used in characterisation.
- Level 2: Standard audio processing features (e.g. peak values, stereo perspective, mean level, dynamics, balance). This level normally requires software to extract the features out of the file formats. Differences in Level 2 features in a migration scenario can reveal errors of conversion not apparent in a level 1 feature comparison. Level 2 features are also meaningful for a single file where they, in many cases, say something about the quality of a file. Level 2 features can thus also be used in single file QA scenarios, where a file delivered for preservation must meet certain requirements.
- Level 3: Sophisticated features to detect similarities in audio files. These kind of features are computed out of audio files – an example could be a "print" (in an image file format) of the wave form or usage of Fast Fourier Transform (FFT). This kind of feature does not give us new, meaningful information about a single file, and is normally only used for comparing two audio files.

### 4.1 Initial problem and prototype solution

The first research performed and prototype developed starts at Level-3 (where no known work within the long term preservation community has been done) and uses FFT and cross correlation methods to compare waveforms. The tool suite being developed is called *xcorrSound*, and the tool used in the following workflow is called *migrationQA*.

We have experimented with our solution on a classic scenario describing a radio broadcast collection; the dataset has been provided by the sample collection available from The State and University Library in Denmark and is available to the SCAPE consortium. The scenario is briefly described: The State and University Library in Denmark possesses a large amount of digital audio files which are radio broadcasts from DR (Danmarks (Danish) Radio). All radio broadcasts from DR in the period 1989 to 2005 have been digitised. The broadcasts were recorded on 2 hour tapes. In order not to lose any data, one tape was put in one recorder and a few minutes before it reached the end, another recorder was started with another tape. This yields two tapes with some overlap. All these tapes have now been digitized and the digitized dataset is 20Tbytes of audio files in 2 hour chunks with overlaps of unknown duration; the library wishes to remove this overlap.

Amongst the many difficulties that can be encountered for such a complex scenario, two are specifically challenging:

- Since we are using different tape recorders and subsequently digitizing the recordings, we cannot rely on the two audio files to be identical on a bitwise level, which means we have to find more sophisticated methods of identifying where the overlap is.
- The length of audio overlap is not the same for each pair of files.

A similar scenario arises when we do quality assessment to evaluate the impact a migration. Particularly we want to be able to determine if we have lost data or if the result is too different from the original.

In both cases we essentially test the similarity of two audio files. In this way we can measure the distance or difference between the original records and the digitized copy, and we can also solve the problem of "unknown overlap length".

A limitation to keep in mind when applying our algorithms is that we have to ensure not to use the same codec (coder / decoder) library used during the migration to measure the QA (Level 3). e.g. if we used some FFmpeg based tool to migrate our original mp3 files to wav files and then we use some other tools based on the same library (FFmpeg) an error of the library in the encode phase will not be discovered. Therefore our repository must be checked with at least two completely independent implementations of codec libraries.

#### 4.1.1 Signal Processing

First of all we measure what is called *cross correlation* in signal processing: given two signals (i.e. in our case the two audio files) their cross correlation is a function of time-lag describing how well they correlate.

Imagine having one audio file function plotted where the x-axis is time and y-axis is the sound it produces at that time step like in Figure 3. Then the cross correlation tries to overlay the other audio file and continuously slide it whilst evaluating how well the two waveforms correlate with the current lag in order to find the best match.

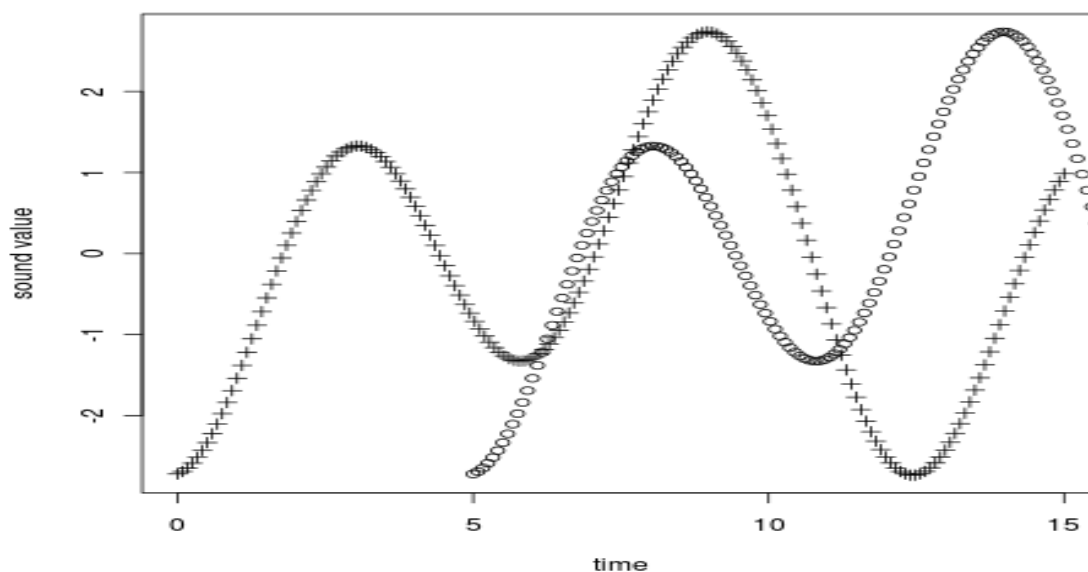


Figure 3: Example of Cross Correlation for two audio files

If we shift the function plotted with + in Figure 3 to the right and we evaluate the distance between the two lines at every step, we eventually find that the two functions are the same, except that they are not aligned.

This method works for the 'overlap scenario' on regular "wave" files. At the time of this first deliverable we are testing how well this algorithm can perform in migration quality assurance scenarios by splitting the converted file and the original file into sufficiently short pieces and retrieving a value (between 0 and 1) indicating how well they correlate. In this way we should be able to report if there are errors during the migration.

The reason for splitting up the files into sufficiently short chunks is twofold. It may help improve performance, and it minimises the risk of missing shorter intervals of mismatch. If we calculate the cross correlation of two two-hour files, which are mostly identical, but one of the files has one minute of silence, where the other still has radio broadcast, this shorter mismatch may be statistically outweighed by the larger matching intervals, and thus go undetected.

This experimental method is currently only available on wave files although we are now porting it to other file formats, keeping with a modular approach to enable our software to be easily adaptable to file formats that do not yet exist.

## 4.2 Taverna workflows

SB currently owns a collection of mp3 files. They are part of the Danish cultural heritage that SB preserves. The majority of the rest of the library audio collection is in WAV (BWF). This format has been chosen as the preservation format as this is a raw format, which needs less interpretation or fewer layers of interpretation to be understood by humans, and is also a robust format.

The mp3 files are to be migrated to WAV according to policy. This is the *LSDRT6 Migrate mp3 to wav* scenario [LSDRT6]. The actual migration will be done using FFmpeg which is one of the SCAPE Action Services recommended tools. The QA will be split into a number of steps. The first step is validation that the migrated file is a correct file in the target format. We currently use JHOVE2 for this validation. The second step is extraction of simple properties of the original and the migrated files, and comparing these properties to see if they are 'close enough'. We currently use FFprobe to extract properties. Another step could be to extract more properties by 'playing' the two files.

The third step in this solution uses the migrationQA analysis tool comparing the sound waves. To use the tool we have to 'play' or interpret the files, just as a human needs to 'play' or interpret the files to

hear the sound. A human cannot look at file A and tell if it is correct or corrupted. We choose a player P and define 'file A played on player P' to be correct. A small randomly chosen subset of files will be played on player P and checked by human ears to be correct making this definition probable. The result of playing file A on player P (when no one is listening) is a WAV file. Playing both the original and the migrated file on Player P, we get two WAV file, which we can compare using the analysis tool xcorrSound. We currently use MPG321 to 'play' the audio files. MPG321 is an independent implementation of mp3-decoder, thus independent from FFmpeg, actually used to migrate the files. Note that no strict order on the steps is imposed. A Taverna workflow might look like the mock-up in Figure 4 (Wor\_MockUp). In short:

- mp3 files are migrated to wav using FFmpeg.
- QA:
  - Validation that the migrated file is a valid WAV file using JHOVE2.
  - Extraction of level 1 (header) properties of original and migrated file using FFprobe. Comparison of properties.
  - Possible extraction and comparison of level 2 properties of original and migrated file.
  - Play original and migrated file using MPG321 and compare the waveforms and output similarity-factor using migrationQA sound comparison tool.

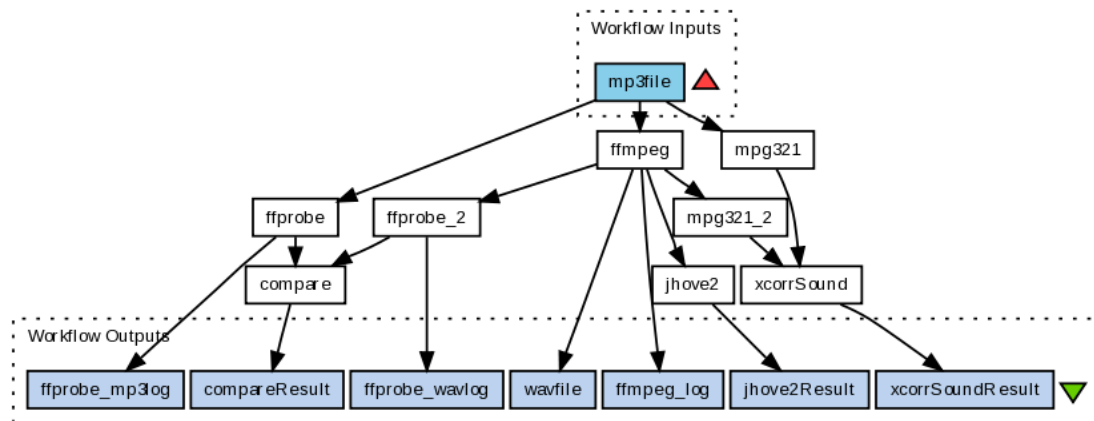


Figure 4: Taverna Mockup Workflow MP3 to WAV

Note the migrated file is a WAV file in this scenario, which means that we only need to play the original file using MPG321. This gives us a second WAV file, and we can compare the two files. This is the solution used in the current workflows.

The reason for wanting to 'play' both original and migrated file using MPG321 is twofold. Firstly we want to make sure we are evaluating the migration rather than the playback. Experience from the current workflows is that MPG321 actually adds a short bit of nothing (silence) in the beginning of the playback, which means that the migration QA tool must shift the migrated sound file to get a match.

Secondly playing both the original and the migrated file, decouples the migration QA demand of WAV input files from the scenario demand of migration to WAV.

All Audio Taverna workflows presented are published on [www.myexperiment.org](http://www.myexperiment.org).

#### 4.2.1 Workflow for FFmpeg mp3 to wav Migration

The *FFmpeg mp3 to Wav Migration SCAPE Web Service Workflow* in Figure 5 (Wor\_FFmpeg) is a simple workflow, which takes an mp3 input file and uses the FFmpeg SCAPE Web Service to migrate the file to WAV.

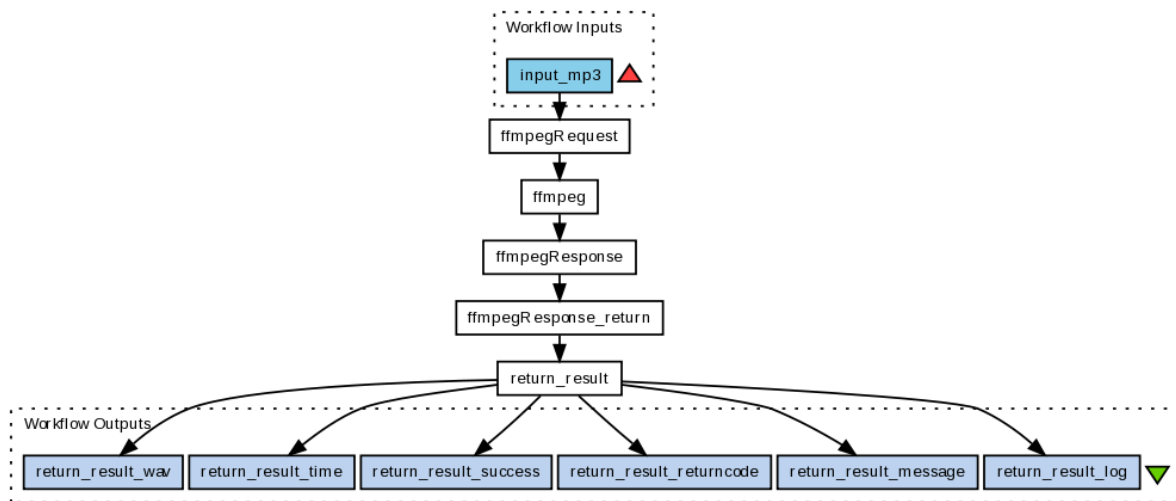


Figure 5: Taverna Workflow FFmpeg migration MP3 to WAV

FFmpeg tool is one of the audio migration tools recommended by the SCAPE Action Services work package. The FFmpeg SCAPE Web Service relies on a local installation of FFmpeg on the host machine. The webservice is based on a shell script, which pipes the FFmpeg log output from std.out to a log file, also on the host machine. The script is named ffmpeg.sh; it is available along with the FFmpeg tool descriptor on <https://github.com/openplanets/scape>. All the SCAPE web services described in this section are also available from <http://fue.onb.ac.at/scape/>.

#### 4.2.2 Workflow for FFprobe Property Extraction

The *FFprobe Audio Files Property Extraction and Comparison Workflow* in Figure 6 [Wor\_FFprobe] uses a simple nested *FFprobe Audio File Property Extraction SCAPE Web Service Workflow* twice. The nested workflow takes an audio input file and uses the FFprobe SCAPE Web Service Workflow to extract properties.

The *Extraction and Comparison* workflow takes both an mp3 input file and a migrated wav input file. Properties are extracted from both files using the nested FFprobe SCAPE web service workflow, which returns a property text file. Some of the properties are expected to be consistent after the migration, and these properties are compared.

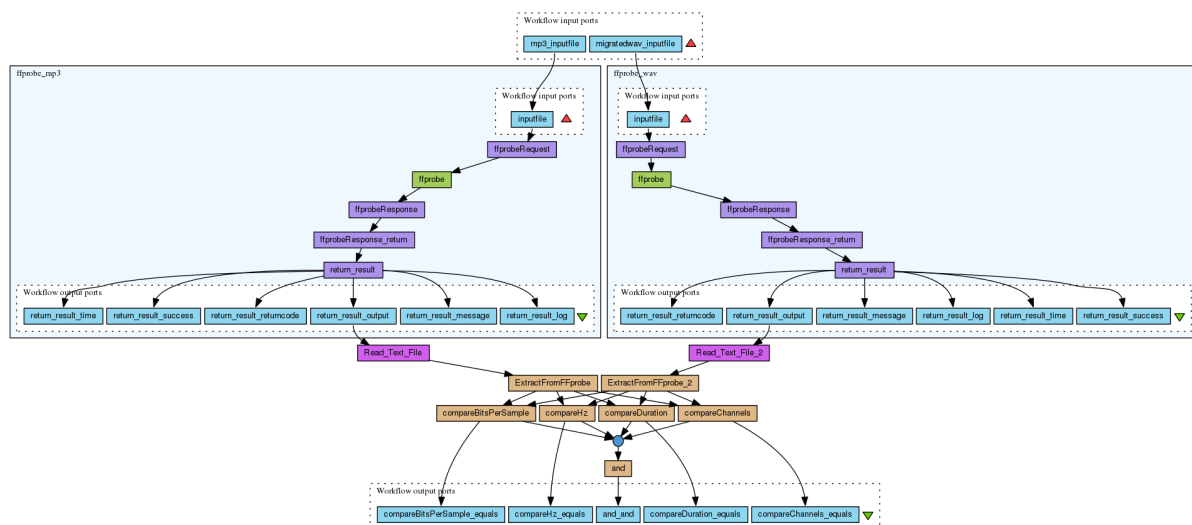


Figure 6: Taverna Workflow for FFprobe Property Extraction and Comparison in QA of MP3 to WAV migration

The FFprobe tool is an FFmpeg based project and part of the FFmpeg tool, and has been used previously at Statsbiblioteket. It will be part of an evaluation of different possibilities for property

15

extraction. The corresponding FFprobe SCAPE web service also relies on a local installation of FFmpeg on the host machine, and on a script, which pipes the log output from std.out to a log file, also on the host machine. The script is named ffprobe.sh; it is also on github, and this is what the ffprobe.xml tool descriptor uses.

The migration command attempts to keep as many properties of the original as possible, but some properties do change when migrating to a different format.

The FFprobe properties that can be compared are sampling rate, channels and bits per sample, while the bit rate follows the format. Also the duration we would expect to be the same, but we have found that in some cases this is not the case. The duration can be off by up to 60 ms without human ears being able to tell the difference, but with the migrationQA tool (see section 4.2.5) telling us that the best match off-set is not zero, which probably means that the migration introduced a short 'quiet' bit in the beginning of the file. This is one case, where the properties comparison would normally require us to reject the migration, but where the waveform comparison in migrationQA informs us that no information have actually been lost.

The Taverna beanshell scripts ensures that sampling rate, channels and bits per sample are the same, and that duration is off by no more than 100 ms.

The logs or the migration and property extraction events should also be saved as part of the file audit trail. We note that FFprobe and FFmpeg are inter-linked. Thus a different tool for property extraction would be preferable.

### 4.2.3 Workflow for WAV File Format Validation using JHOVE2

The Taverna workflow *JHOVE2 Web Service Validate WAV files* in Figure 7 (Wor\_JHOVE2) uses a *JHOVE2 SCAPE Web Service* set up to validate the input WAV files.

JHOVE2 is a characterisation tool, which can also be used for validation. JHOVE2 was chosen as it includes a WAVE Module which validates WAVE files by checking for conformance to all normative requirements in WAVE. The output of the web service is a JHOVE2 property xml file, containing the various properties JHOVE2 extracted from the file. To check that the file is actually valid, the JHOVE2 feature "isValid" should be true. This is checked by the Taverna workflow. Names of invalid files are copied to a separate list, for manual checks.

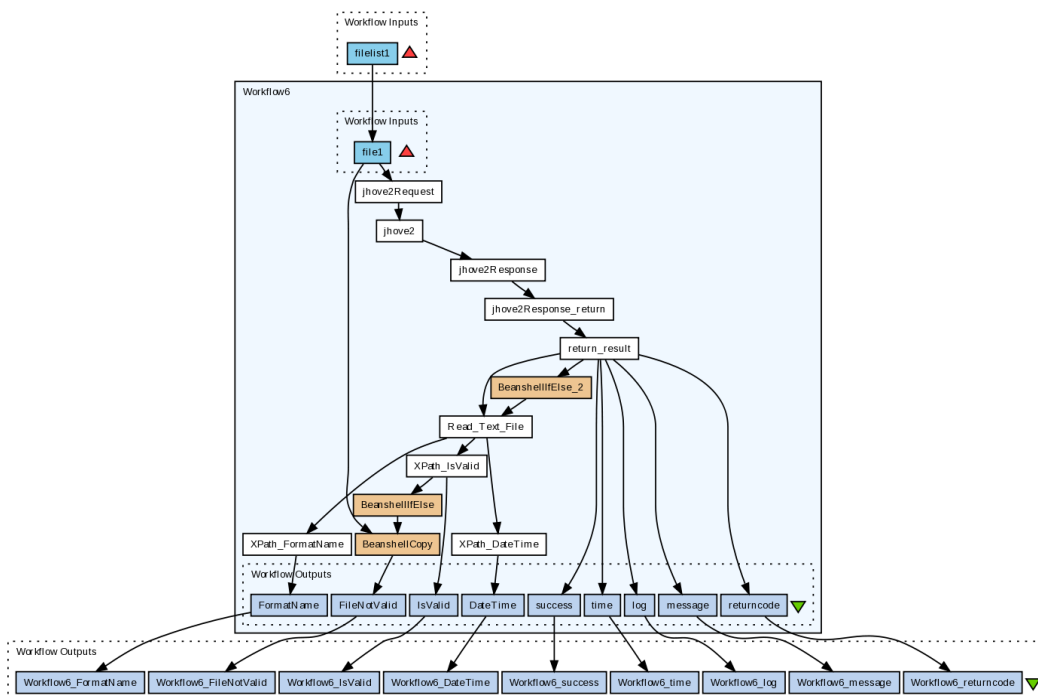


Figure 7: Taverna workflow for WAV File Format Validation using JHOVE2

#### 4.2.4 Workflow for Migration, Validation, Feature Extraction, Comparison

The *Mp3 to Wav Migrate Validate Compare Workflow* combines all of the above workflows to migrate mp3 to wav and perform a basic Quality Assurance, see Figure 8 [Wor\_Basic].

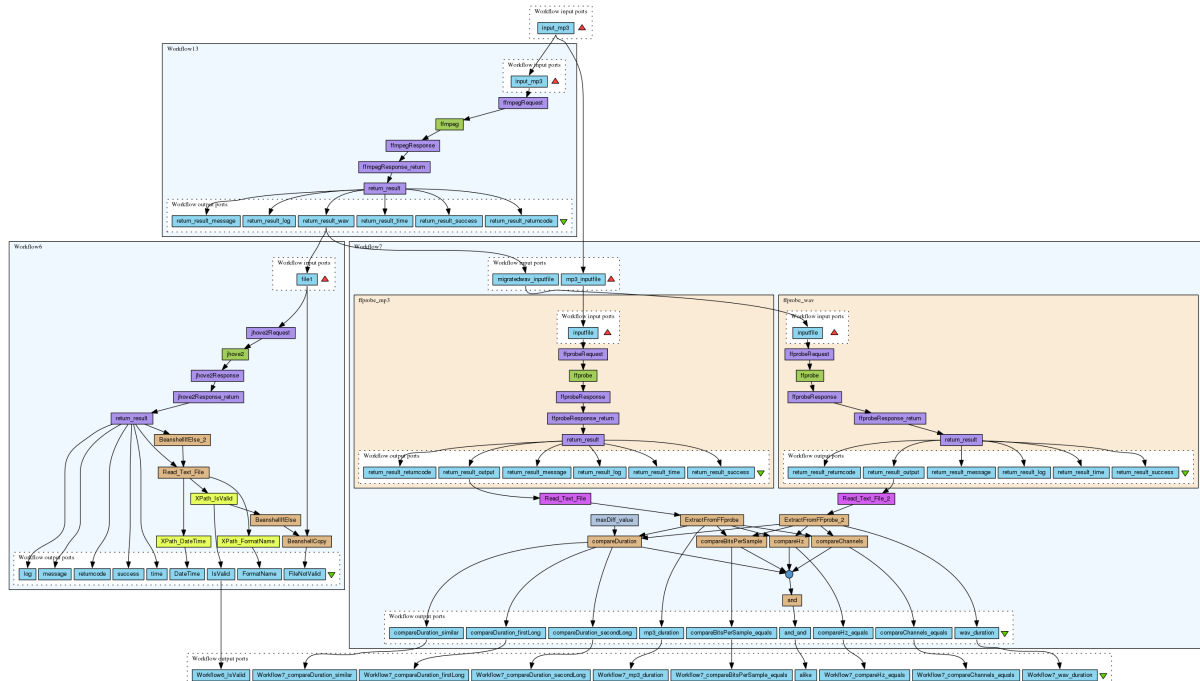


Figure 8: Combined Workflow for Migration, Validation, Property Extraction and Comp

#### 4.2.5 Workflow for migration QA Sound Comparison Tool

Figure 9 shows the migrationQA SCAPE Web Service Wav File Comparison Workflow (Wor\_migrationQA) which takes two wav input files (original and migrated) and uses the migrationQA SCAPE Web Service to compare the files.

The migrationQA workflow is not part of the combined migration and QA workflow (Figure 8) as it still does not scale to the two hour files in the dataset we are currently working with. The migrationQA tool is a new tool based on the xcorrSound tool described in the Audio introduction. The migrationQA SCAPE Web Service needs a local installation of the tool on the host machine. The migrationQA tool is available from <https://github.com/openplanets/scape-xcorrSound>. The tool descriptor migrationQA.xml uses a script migrationQA.sh to pipe the output into a file. The script is also available from github and must be put on the host machine.



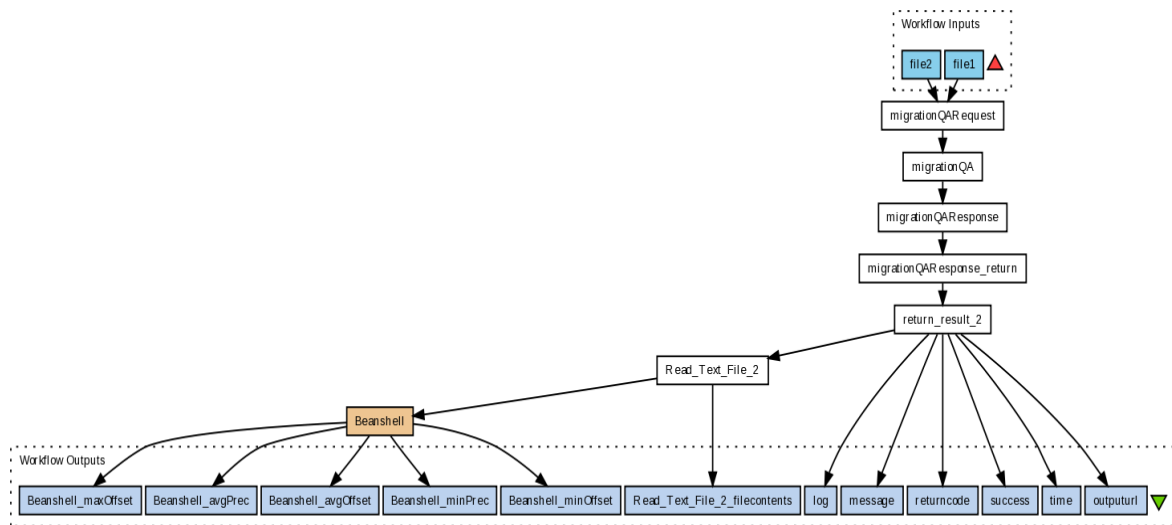


Figure 9: Taverna workflow for migration QA Sound Comparison Tool

Note that the input to this workflow is two WAV files. The *MPG321 Play mp3 to Wav SCAPE Web Service Workflow (Wor\_MPG321)* is a simple workflow akin to the FFmpeg SCAPE Web Service Workflow and the FFprobe SCAPE Web Service Workflow. Given an input mp3 file the workflow uses the MPG321 SCAPE Web Service to play the mp3 file into a Wav output file. MPG321 is a command-line mp3 player. We use it here as an mp3 to wav file decoder. The MPG321 web service relies on a local installation of MPG321 on the host machine. The tool descriptor mpg321.xml uses the mpg321 command line tool directly. When using the migrationQA workflow to compare the migrated WAV file to the original mp3, the original file is 'decoded' into WAV using the MPG321 workflow. The *migrationQA* tool developed in this WP basically compares two waveforms using fast fourier transformations to be able to tell whether the actual content of the two files is the same or close to the same. This will reveal migration errors where the resulting file might have all the right properties (e.g. length) but it might be fully or partially empty as an example where the migration tool failed because of certain errors in either the original files or the migration tool itself. And when needing to migrate millions of files this automated QA is quite critical. This migrationQA program produces output that can be used to compare two wav files that are expected to be alike. The program outputs an offset value and a match value for each 'chunk', where the two audio files by default are split into 5 seconds chunks. The offset value is calculated in seconds. The match value is a number between 0 and 1, with 1 indicating completely identical waveforms and 0 indicating totally different waveforms. One way to use these numbers is to verify that all chunks have the same off-set and that all chunks have a good match value (close to 1). The current version of the migrationQA tool splits the audio input files into 5 second chunks. This means that the tool now (sort of) scales to long audio files. It also minimises risk of 'drowning' short intervals of 'noise'. At this moment the tool does not work correctly on audio files shorter than 5 seconds or on very long audio files. The mp3 files in this migration scenario are two hour radio broadcasts. This means that the current tool does not scale. And the web service times out before the tool completes on longer audio files. To make a tool that scales, two approaches are under consideration. We can try using the current tool on down-sampled files, but with longer time chunks. To test the accuracy, we can create some smaller test sets with small file sizes and known results. Also, the current tool cuts the files to be compared into shorter pieces, but compares them sequentially. It should be possible to do some of the comparisons in parallel. This could either be done by letting MPG321 output a set of files, or by making the migrationQA tool multi-threaded. Some level of approximation is also feasible. If we cut the 2-hour sound recordings into 1440 5-second-chunks, we can do some statistics on 'how many chunks do I need to compare to establish

successful migration with a 95% likelihood'. It may be that if 144 of these chunks all have the same off-set and all have a good match value, we are able to say that we are 95% certain the full sound recordings match. That would give us a direct likelihood vs. resources relationship. This may be an interesting piece of research to do in full.

### 4.3 Benchmarks and Validation Tests

The Migration and Partial QA solution described above has been developed as a Taverna workflow using web services. This puts focus on availability rather than scalability. The sparse tests so far have also been run through Taverna.

#### 4.3.1 Mp3 to Wav Migrate Validate Compare Workflow Run Times

We tested the *Mp3 to Wav Migrate Validate Compare Workflow* presented in Section 4.2.4 on file P1\_1000\_1200\_890106\_001.mp3 from the *mp3 (128kbit) with Danish Radio broadcasts* testbed dataset. The file is 112 Mbyte and the duration is 2 hours, 2 minutes and 5.23 seconds. The workflow was run from Taverna on a local work station using the web services deployed on the Stats Biblioteket test machine. The total time for the workflow is 2.3 minutes, and the most expensive nested workflow is the JHove2Validate workflow with 1.3 minutes, closely followed by the FFmpegMigrate workflow with 59.2 s. The FFprobeExtractCompare workflow with 12.9 seconds seems to be the cheapest QA component in this set-up. The result of the workflow is a migrated WAV file, and a report that it is valid and that the extracted properties have been preserved.

Running the workflow on 4 additional files from the dataset gave similar results of between 2.2 and 2.3 minutes, valid migrated WAV files and properties preserved. The 6<sup>th</sup> test run also gave a result wav file, but we could not hear the file, and neither JHOVE2 or FFprobe was able to read the file.

Interestingly the nested FFprobe workflow failed, while the nested JHove2 workflow finished, but without output. At the intermediate values, we see the JHOVE2 SCAPE web service is unable to read the migrated file. This raises a question of error handling in the Taverna workflows. The nice feature is that in this layered approach, it is still easy to pinpoint where the error originated, but we may want to look at more consistent error handling. Maybe the workflow should not fail in this case, but rather give meaningful output about a failed web service or beanshell.

We note that we are working on 2-hour sound files. The average file size of the original mp3 files is only 118Mb, but the migrated wav files are approximately 1.4Gb. This means we can probably not hope to improve much on the performance of the actual FFmpeg migration of the individual files. The mp3 (128kbit) with Danish Radio broadcasts collection is 20 TB and around 150000 files. This means that running the basic workflow migrations sequentially on the test machine would take around 300 days. We can however hope to improve by using the Scape execution platform instead of doing the migrations sequentially.

#### 4.3.2 migrationQA SCAPE WS Wav File Comparison Workflow

The migrationQA workflow does not scale nicely to 2 hour sound files, but we have run a test on a file cut to 12Mb (about a tenth of the original size) using dd. The *Mp3 to Wav Migrate Validate Compare Workflow* used only 34 seconds on the cut file. The migration and the file format validation were successful, but the property comparison reported that the files were not 'close enough'. The reason for this is that cutting the file does not change the header information, so the duration of the original cut file is supposedly 2 hours, 2 minutes and 5.23 seconds, while the duration of the migrated file is 13 minutes and 6.38 seconds.

Playing the original cut mp3 using the *MPG321 Play mp3 to Wav SCAPE Web Service Workflow* used 11.7 seconds. The *migrationQA SCAPE Web Service Wav File Comparison Workflow* presented in Section 4.2.5 took 1.4 minutes. The result was also negative, but an inspection of the output showed that only the last chunk differed, which probably means that FFmpeg and MPG321 handled the cut off differently.

We will do some additional tests to get an estimate of the performance, but we estimate that including the migrationQA and doing a migration plus full QA of the full dataset will take a decade with the current performance.

#### **4.4 Audio Property Extractor, Comparator and Format Evaluation**

Along with the audio QA workflow task, we have also been evaluating available tools for audio property extraction and comparison, and existing audio characteristics formats. This task includes XCL, AQUAadio / getID3, FITS, Tika, FFprobe and AES. The XCL tools are available from [http://planetarium.hki.uni-koeln.de/planets\\_cms/](http://planetarium.hki.uni-koeln.de/planets_cms/). AQUAadio / getID3 is available from <https://github.com/openplanets/AQuA/tree/master/aquadio>. AES is the Audio Engineering Society. AES Standards include AES60-2011 Core audio metadata and AES57-2011 Metadata - Audio object structures for preservation and restoration.

The AES formats are available from <http://www.aes.org/standards/>. They could be compared directly to expected output from above tools. This task has an evaluation/documentation focus, and has been put on hold while we worked on a solution including web services and Taverna work flows for the “Audio mp3 to wav Migration and QA Workflow” described in the SCAPE wiki.

## 5 Web QA workflows

Maintaining high quality of Web Archives has always been a challenge and is becoming more and more complex in regard to the complexity of objects archived, the technologies used by webmasters and to the growing size of the Web archives and of the Web itself.

Web Archives are composed of very heterogeneous data most of the time stored in the (W)ARC format [WARC], which can be defined as a container for all resources captured during a crawl without any specific logical order.

To capture this data, web archives use crawlers such as Heritrix [heritrix], a parsing robot, which captures resources from the Web following predefined parameters and a scope. Crawls can be selective (domain, sub domain or even page or resource level) or large (.uk or .eu domain, for example).

Although improvements are constantly made in the domain of crawling strategies and tools (execution based crawlers or tools, access tools, etc.), web content remains extremely complex to capture and access, and crawls can never be defined as exhaustive.

In this context, and because Web Archives are growing in size and preservation of Web content is becoming more and more essential, developing scalable quality assurance methods is crucial.

For now, most of the quality assurance work (QA) is done manually by visually comparing pages of a captured site to the live version. This process is very costly and time consuming and cannot be applied to large crawls nor to a regular quality check of Web Archives. It is therefore not a scalable method.

### 5.1 Scenarios description

To solve this problem, quality assurance scenarios were described as part of the Web Testbed Work Package (TB.WP.1) of the project.

All scenarios outlined in this section aim at resolving issues encountered by Web Archives looking into providing high quality level archives but also maintaining this quality over time and preserving not only the content itself but also maintaining access to this content.

- WCT1: Improving completeness of archives by automatically detecting missing "elements" on a web page or part of a web page.

Defining crawl frequencies to get as much original content as possible by avoiding duplicates.

- WCT4: Using image comparison to detect rendering issues due to format obsolescence within the archive and monitor technological landscape.
- WCT7: Applying image comparison to Web migration scenarios.

These scenarios will be achieved through the use of a workflow using image and/or structural comparison.

Comparison will be made at pages level (or part of pages) using screen shots and/or the structure of the web page (DOM) .

We envisage scenarios using image references that could be stored in a database.

A first important step would therefore be to create such databases of reference snapshots of Web pages at a specific time. Each snapshot would be associated to a number of metadata that could be used depending of the cases described above. Useful metadata are the date and time of the snapshot and any information related to its environment such as the browser used and its version.

Using references snapshots, we would then define large and automated test strategies to control quality of specific crawls or the Web Archive quality itself. Such tests would serve all scenarios described above.

A first use case would be to compare web pages or part of web pages to reference screenshots after a crawl is completed to detect missing part or rendering issues. This detection could trigger actions such as a recrawl of part of the domain crawled or a recrawl of the whole domain.

This comparison could also be made regularly on a defined subset of the archive that would

be used to detect obsolescence of format by identifying rendering issues when using such or such browser or browser version.

Results of these comparisons would be fed to a statistical Data Base that could be used within the automated watch component, in particular to the web watch adaptor [PW.WP.1\_D12.1].

Through these scenarios web archives would not only gain in quality but would also contribute actively to format registries completion.

A second possible use case would be related to the migration to a new infrastructure or system, Web archives would use their reference snapshot Data Base to test access and integrity of content to ensure the quality of the archive.

Internet Memory Foundation (IMF) is migrating its web content, currently stored into (W)ARC files to a new infrastructure based on Hbase. The archive contains around 200 TB of data and is growing rapidly. Most of the content crawled will need to be migrated sometimes this year.

Once the new infrastructure is ready, services provided to cultural institutions by IM will have to rely on this new infrastructure. The Foundation is currently providing a high-level quality archive and related services such as redirection from live missing content to the archive or a full text search.

Looking at the investment in term of manual quality assurance, crawl preparation and developments, Internet Memory Foundation does not want to get a lower quality after content is migrated.

We are therefore planning to build a "quality test" migration using tools and methodologies developed by UPMC to detect and repair migration defects as described in WP11 work description.

## 5.2 Taverna workflow

The Web QA workflow is a workflow that uses screenshots of web pages as input, analyse them using visual and structural comparison and decides if the web page should be recrawled or not depending on the quality.

In Figure 10 it is shown how the MarcAlizer tools decides whether a webpage must be archived or not, according to the visual and structural changes. The class needs as inputs: VIPS files, described in [CYWM2003], and web screenshots. Finally, a score that represents the classifier decision and the computational time required are printed. A negative classification score means that the two webpages are dissimilar and thus both need to be archived, a positive score means the two webpages are similar.

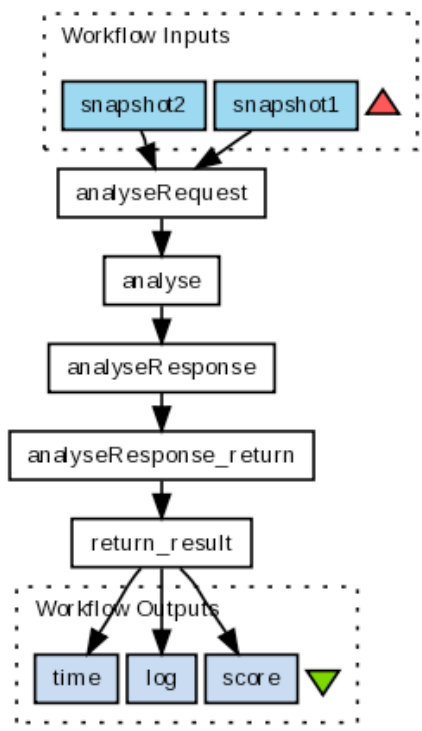


Figure 10: Taverna workflow for Web QA

### 5.3 Tools and Interfaces

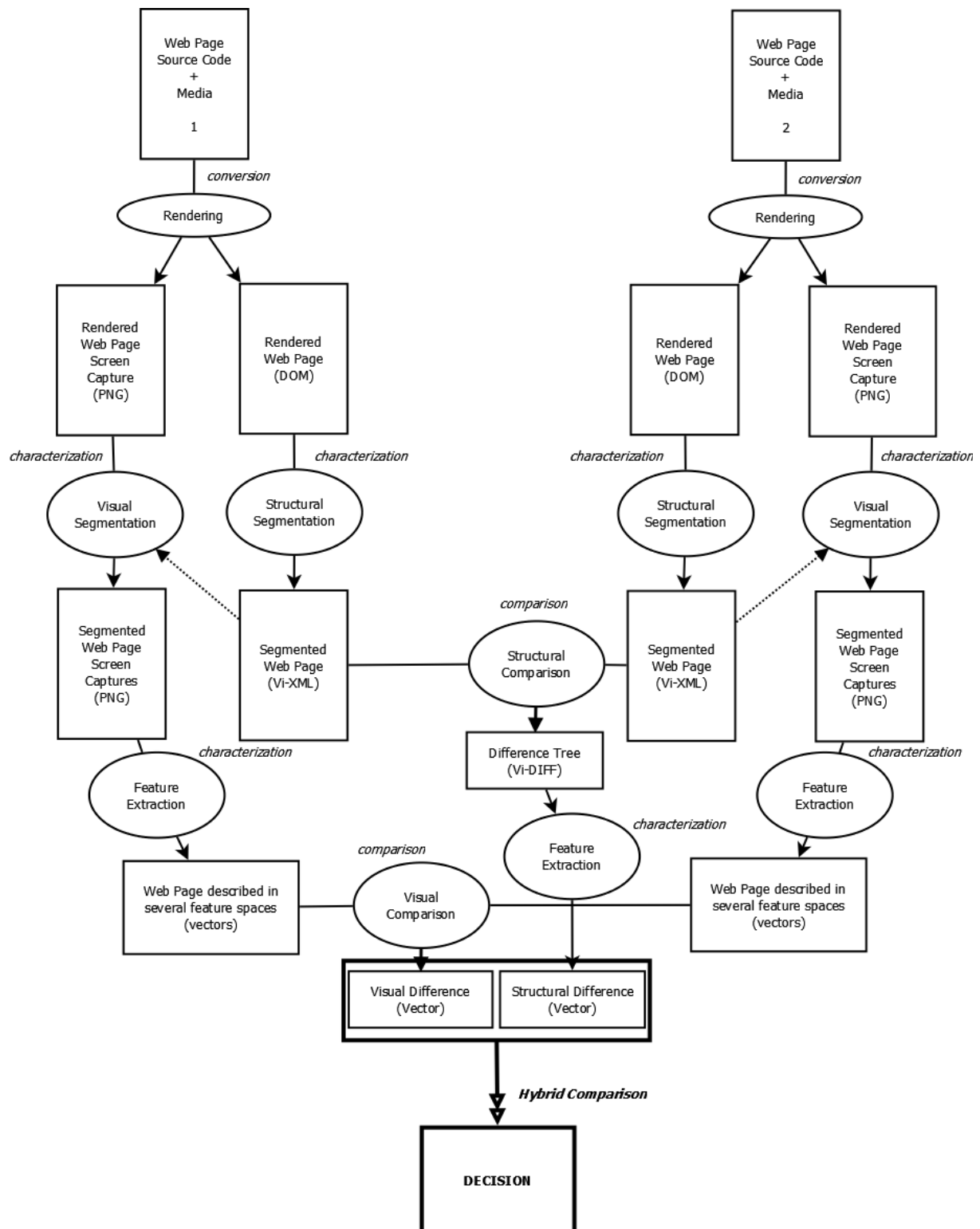


Figure 11: A Webpage is described by its source code (html, Javascript...) and the media (background, pictures...) it embeds.

The diagram in Figure 11 show the interaction between the different steps necessary for quality assessment of web pages: conversion, feature extraction and similarity calculation.

### 5.3.1 Conversion

The conversion step is determined by the Web browser used to load the Webpage, its version and the installed plugins. In our case, the Web browser we use is Internet Explorer since it is integrated in VIPS (Vision-based Page Segmentation), a Webpage segmentation program. VIPS [CYWM2003] returns an XML tree that represents the segmented Web page. The segmentation is based on heuristics that use the HTML tags of the Web page and the visual homogeneity (called Degree of Coherence) between the segmentation blocks. We use an enriched version, developed by [BSGP2009], of the XML tree returned by VIPS, named Vi-XML.

A screen capture of the Webpage is also taken in the PNG format with an Internet Explorer Webpage rendering capture utility named IECapt (<http://iecapt.sourceforge.net/>) that uses the same version of Internet Explorer as VIPS. Since only the DLL of VIPS is available on the internet, we had to use another program (IECapt) to take the screen capture (because it is not integrated in VIPS).

The coordinates of the segmented blocks of the XML tree obtained with VIPS are used to get the different sub-images corresponding to the visual blocks of the Webpage.

### 5.3.2 Feature Extraction

The Visual Feature Extraction step proceeds as follows [SS2001].

Basic visual components are selected in the image: pixels, patches, regions, points of interest, edges, etc. To describe these visual components, shape-based, colour, texture features are commonly used.

In the case of colour, which is the most often used feature for image indexing, the building of colour features is based on the choice of a colour space, which can include properties such as invariance with illumination. Many systems simply use the RGB colour space however, transformed colour spaces such as HSV or CIElab, differentiating hue from intensity, are also widely used.

Once the features are extracted, building a visual codebook is an effective means of extracting the relevant visual content of an image database; such a process is used by most retrieval systems.

The codebook can be determined *statically*, the elements (words) of the codebook are then a priori defined. A second approach is to perform a *dynamic* or *adaptive* clustering, using a standard clustering algorithm such as k-means [HW1979]. In this case, the visual codebook is adapted to the image database. Once colour, texture, shape-based codebooks are carried out, image signatures are computed. For each pixel, the closest codeword is detected and a pixel distribution is generated for each image over the visual codebook.

In the visual codebooks computed from local feature extraction and characterization as Scale-Invariant Feature Transform [L1999] descriptors, the resulting codebook dimension is usually much larger, around 1000. There is no consensus about the best way to handle this problem of dimension, and there are a lot of attempts to deal with every high dimension giving promising results.

### 5.3.3 Similarity

Once signatures are computed, a metric or similarity function has to be defined to compare images. Typically the Euclidian distance is used to compute the similarity (or dissimilarity) between histograms, or more generally a Minkowski distance. Similarity functions can also be learnt from a training set, for instance with kernel methods.



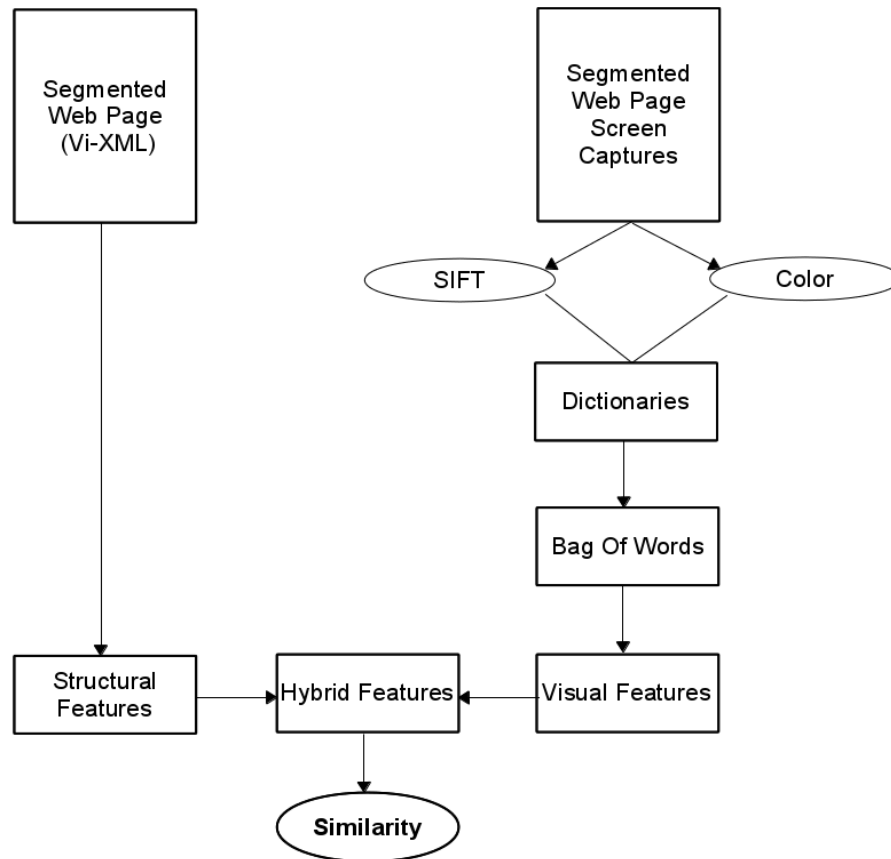


Figure 12: Functional similarity diagram

In our approach, we merge the structural and visual differences to obtain a similarity vector combining the DOM structure of the code of the Webpage and its visual aspect.

### 5.3.4 Training and evaluation

The visual and structural comparison methodology described in this section were trained on a set of data provided by Internet Memory.

The IM quality assurance team selected and compared visually a set of urls (pages of archived sites) chosen because they were identical (same url) but crawled at a different time. Around one thousand annotated urls were provided to the UPMC team, from which around two-hundred showed significant enough differences. These two-hundred were used to improve the tool training.

The annotation consisted in assessing whether two urls (pages) were "similar" or "dissimilar".

For the first round of tests, we added the "ambiguous" option as it was sometimes difficult to choose between similar and dissimilar.

For the second set of annotation, we removed the ambiguous option and asked our QA team to annotate all pairs as similar or dissimilar.

The assessment guidelines can be summarized as follows:

- Method: QA team will compare both the structure and the content of pages to create pairs. Page will be checked entirely.
- Evaluation criteria: Would we ask for a re-crawl in this case?
- Pages: Several level of pages (homepage, page at deeper level that might be updated less frequency)

- Frequency: Close in time (1 month is the largest interval)
- Tag:
  - "Similar": Pairs are very similar but not totally identical. Pairs are not different enough to trigger a re-crawl.
  - "Dissimilar": Pairs contain strong differences, enough to trigger a re-crawl.

To evaluate our system we used a single 3.47GHz PC to compute the average running times of comparison on the Internet Memory dataset. The captured images of Web pages are about 929 ( $\pm 251$ ) width and 1272 ( $\pm 723$ ) height pixels.

The brute force based algorithm time of execution of the whole process is close to 20 seconds being the SIFT computation the bottleneck.

Some of the parameters can be optimized to speed up the process without losing the accuracy of the results. With these optimizations the time of computation to process each couple of images from the Internet Memeory dataset in less than two seconds with an accuracy of 88%.

## 6 Document QA workflows

In this chapter we will provide a survey of approaches and tools for quality assurance in the preservation of documents created using office productivity tools such as Adobe Publishing Tools, Microsoft Office Suite, Libre Office and similar software. For the sake of simplicity and common understanding, we will refer to these documents simply as "office documents".

We first reflect on the requirements for quality assurance, emphasizing the need for (1) comparing document rendering and layout, and (2) preserving or enhancing the document structure to enable rich usage scenarios (including search, browsing and document content analysis). We then describe the technique of using intermediary document representations to enable tuning and calibration of the document analysis tools and to measure specific document properties. Since the scale of a document migration and its QA analysis represent an issue, we include the description of the cloud architecture that would support such operations and will be built in concert with the development of the QA methods and tools. We then discuss the data sets that will be made available for conducting the work in our workgroup.

### 6.1 Design of Feature Probes

A standard approach is to design *feature probes* into document formats to detect, extract, and compare a specific characteristic. That, in turn, requires that the two digital objects are transformed into the *same representation* that allows the same processing and comparison of a specific feature. In contrast to document format migration, these transformations of documents are referred to as *intermediary representations* created solely for the assessment of the tools or migration outputs. For example, in order to determine whether two versions of a document, one in format A and another in format B, have the same number of pages, one may use, as an 'independent' reference point, the printer files created when documents are sent to the same printer. That would involve applying the same analysis of the printer files for the two documents, i.e., identifying and counting codes for the page breaks.

Generally, selection of the intermediary representations is driven by a specific document characteristic that is of interest and the availability of techniques to extract and measure that characteristic. For example, a number of features can be detected by converting into lossy formats, like images, and applying vision techniques. Others may require preservation of structural elements of the documents, such as paragraphs, line feeds, etc. In that case the documents may be converted into XPS, for example.

### 6.2 Experiment Design for Evaluating Feature Probes

Like any other tools, feature probes need to be evaluated for their accuracy of analysis. In order to automate the evaluation, one typically resorts to the 'round-trip' evaluation which normally involves three phases, listed below. For example, if we want to assess the accuracy of determining paragraphs of text from JPG images of documents using the analysis of the OCR output, we may apply the following round-trip experiment:

- Select set of documents in the format A whose characteristics are fully known. For example, we may have an XML document for which we know all the layout properties.
- Transform the documents into JPG images and apply OCR to create the basis for layout analysis. From the OCR output extract information about the paragraphs in the document.
- Automatically compare the output of the step 2 with the known characteristics of the documents in step 1.

This approach provides a basic evaluation of the feature probe effectiveness. It can be used to decide whether the particular intermediary format and analysis is generally reliable for characterizing specific aspects of the document.

In many instances, a finer grained analysis of errors is necessary. For example, assuming that the tool correctly detects 85% of the paragraph breaks, it would still be important to understand whether this is acceptable in different usage scenarios. For example, if paragraph detection is necessary for document rendering in reading applications, the remaining 15% of incorrectly detected paragraphs may significantly affect the user experience. If, however, it is used for text indexing at the paragraph level in order to facilitate precise search, this level of accuracy may not be detrimental to the search quality.

Tracking of systematic errors and characterizing the significance of such errors is an important aspect of quality assurance. This is typically facilitated by careful data sampling and collecting feedback from the expert assessors. However, with the availability of crowd sourcing platforms, we have an opportunity to scale up the operations and design Human Intelligence Tasks (HITs) to capture human perception of the errors and perceived significance within a specific usage scenario. Alternatively one may adopt an "open source QA" model, in which the volunteers can participate in the QA process assisting with the time-consuming manual aspects.

Therefore, the architecture for automated evaluation of document characterization tools should be enhanced with support for large scale human feedback. However, we need to make provisions for the quality assurance of such human engagements. In the case of crowd sourcing, many factors affect the output of the crowd workers, including the pay, expertise, required time and effort, etc. On the other hand, the open source QA model exhibits drawbacks similar to the open source software, lacking the control of the task progress. Thus, appropriate measures need to be taken to control for these factors.

### 6.3 Document Representations and Analysis

In this section we describe document representation and characterization techniques for two specific preservation goals:

- Rendering of the document and characterization of the layout features
- Preservation of the document structure to enable interaction and access through search and browsing.

In both instances we aim towards the comparison of documents and therefore consider the common intermediary representations that support the required analysis. Table 1 provides a summary of the methods that we consider in this section.

Document Formats	Intermediary Format	Analysis Techniques
WordPerfect, MS Office	XPS	OCR analysis for layout features
PDF	BMP	Vision techniques
MS Office	Annotated XPS	XML

Table 1: Comparison of office documents

#### 6.3.1 Analysis of Document Rendering using XPS Representation

As part of the PLANETS project, Microsoft Research integrated open source tools for conversion of office documents in proprietary formats into XML and HTML based formats. In order to enable visual inspection of documents and assessment of the conversion process, the resulting documents were subject to several transformations. First, we use XPS to create the rendering of the documents in the common intermediary representation. We then applied the OCR analysis to detect the difference in the rendering of the documents. Finally, we

render the annotated documents in the image viewer, indicating the document differences in the XPS representations. Figure 13 shows the document processing workflow.

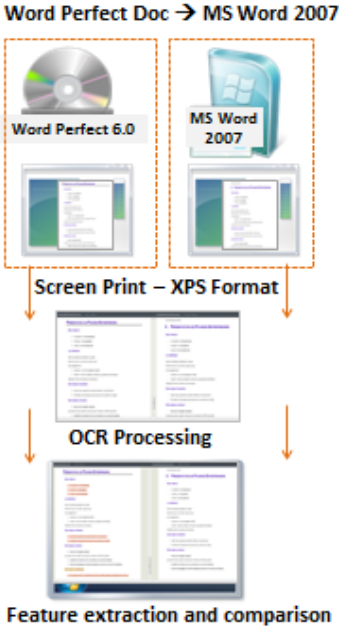
Source Format	Target Format
– WordPerfect 5	– OpenXML
– WordPerfect 6	– ODF
– DOS Word	– UOF
– Word 2, 6, 95	– HTML
– Word 97-2003	– XCDL (format defined in PLANETS/PC)
– RTF	
– ODF	
– OpenXML	

*Table 2: Document conversions investigated within PLANETS and supported by SCAPE*

Most of conversions from the Table 2 are currently supported in SCAPE Microsoft Azure platform and by aggregating information from the OCR output analysis, we can provide a report indicating the difference in the XPS representation. However, this analysis is subject to the uncertainty about the accuracy of two other tools:

- The accuracy of the Document XPS conversion;
- The accuracy of the OCR analysis of XPS input

Accuracy metrics are going to be investigated in more detail and we will try to correlate results with human perception.



*Figure 13: Document comparison using XPS as intermediary format*

### 6.3.2 Analysis of PDF Rendering using Image Representations

In this section we describe QA of preservation actions for PDF documents based on:

- The conversion of PDFs into images and
- The extraction of significant properties with comparison based on image analysis.

Figure 14 illustrates the PDF document Comparison Workflow. It makes use of *ImageMagick software* [Appendix A: External Tools] to convert PDF content to BMP images using its *convert* command line tool. This common raster image format is then used as an intermediary representation to compare the renderings of two PDF files through feature extraction.

Feature extraction is provided by *extractFeatures* tool and comparison is performed by the *compare* tool (see section 7.1.1). Both tools are written in C++ and provided as executables using associated DLLs on Windows or shared objects on Linux. The results of feature extraction are represented in XML format. Such XML representation is used as input for *compare* tool. Given two input XML files, the *compare* tool generates a comparison report in XML format.

As shown in Figure 14, the PDF comparison workflow comprises four phases:

- *PDF Rendering*: The execution of the workflow begins by converting each PDF using the *ImageMagick* tool, resulting in a set of image files in common raster image format, e.g. in BMP format. The number of created raster image files depends on the PDF content.
- *Extract features*: This step applies the *extractFeatures* tool to each BMP image file in order to extract colour histogram features. The *extractfeatures* tool (see Section Command Line Tool: *extractFeatures*) has two input parameters: the URL to the input image file and the number of histogram bins as integer. The output is an XML file with histogram data.

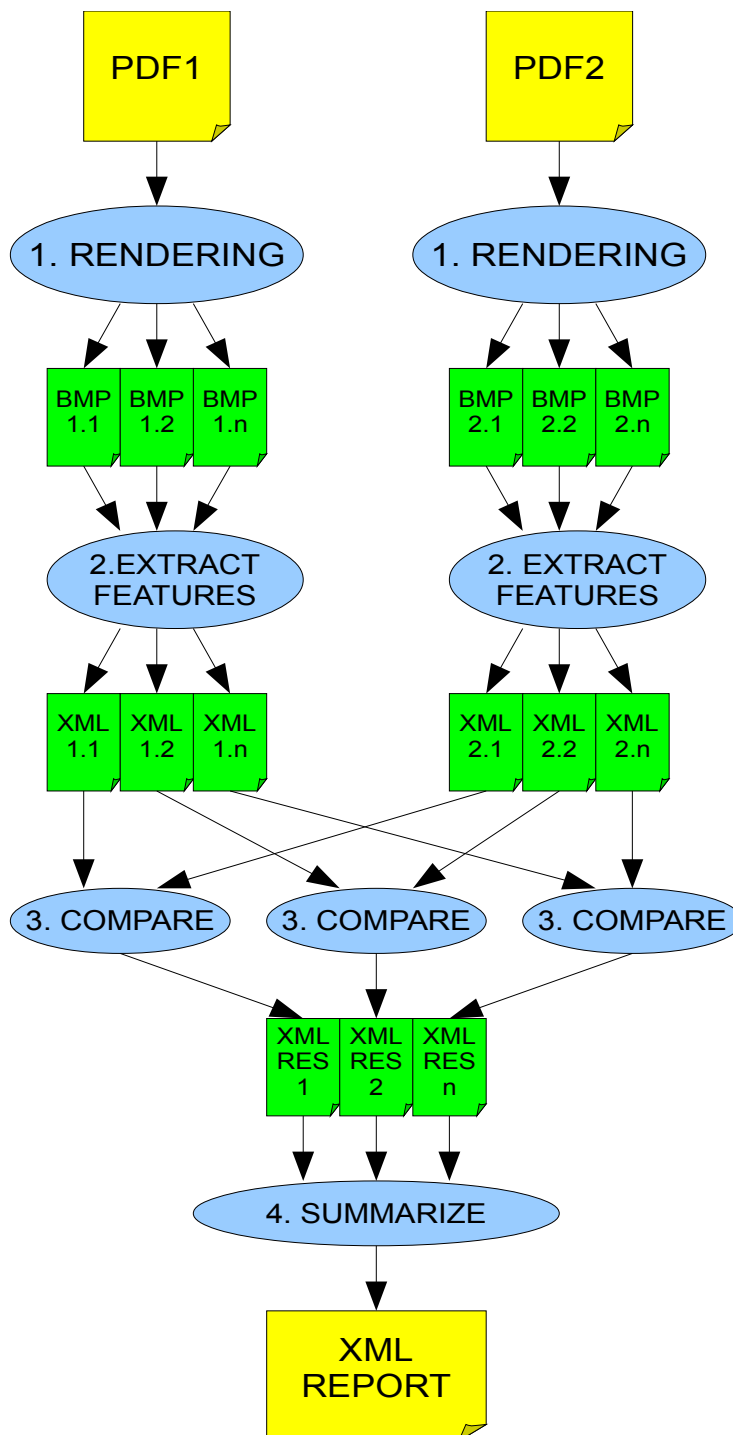


Figure 14: PDF comparison Workflow

- *Compare*: In this step XML files containing histogram features from one PDF file are compared with corresponding XML features files of the second PDF file using the *compare* QA tool (see Section Command Line Tool: compare).

The output is an XML file containing the comparison result.

- *Summarize*: In this last step, comparison results for each XML chunk of PDF file and chunk counts are analysed and summarized.

### 6.3.3 Remarks on the rendering and QA algorithms

*ImageMagick* conversion is limited with respect to the font and colour quality. Also, the

handling of layers and spot colours in PDFs is not implemented in version 6.4.9-1. Some shortcomings of *ImageMagick* are solved by *Ghostscript*, but handling of layers is still unsupported.

The methods for QA are currently under development and more sophisticated methods are being investigated for inclusion in future releases.

## 6.4 Structure Representation and Comparisons

In some instances we expect to be able to expand the conversion tools to expose the analysis that is performed during the conversion process. For example, the conversion of office documents into XPS includes analysis of the document structure. This information can be preserved and used as part of the intermediary representation of the documents, used for comparison. This is under the assumption that XPS converters are standardized and provide similar information across office document formats. When that is the case, we can expand our analysis from simple OCR or Image analysis with the comparisons of the XML representations.

Similarly, we consider methods for representing the Document Object Models (DOM) of MS Office Documents that will allow the comparison and validation that particular elements of the DOM are correctly migrated. This is particularly important in instances when interaction with the document is of essence.

For example, preserving references within the document or the table of contents as a means for navigating the document may be an essential requirement for preserving the utility of that digital object.

In other scenarios we will be looking at the migration strategies that enhance the interactivity with the document. For example, digitized books are represented in the TIFF or JPG formats. Through the analysis of the OCR software we can, in many instances, reconstruct the content structure of the book. The resulting table of contents can then be used to enhance access to the document's content through browsing and indexing for search that takes into account the book structure.

## 6.5 Cloud Solution for Storage and Computation

In order to facilitate the scale and the computational demand for document migration and QA, the SCAPE project will develop cloud solutions that support the preservation action operations. Based on early discussions amongst the SCAPE partners, there will be two implementations of the cloud solutions on different computing platforms, each offering different opportunities and challenges. The multiplicity of the cloud solutions will enable investigation of the interoperability issues and comparison of different approaches.

In the following section we briefly describe the characteristics of the cloud solution based on MS Azure technologies.

### 6.5.1 Tools and Program Interfaces in MS Cloud

In this section we provide overview of the solution that is being developed to explore how the architecture proposed for SCAPE can be implemented on Windows Azure. This solution makes use of a number of key Microsoft technologies including Silverlight, RIA Services, WCF, SQL Azure and Windows Azure. There is also a description of tools that are going to be used for PDF2BMP QA method that will run on the SCAPE platform.

Tools needed for the working environment for MS Office documents are: Visual Studio 2010, Silverlight 4 Tools, WCF RIA Services, Expression Blend (optional), Silverlight Toolkit, Windows Azure Tools & SDK, SQL Server 2008 R2 and Windows Azure Free Trial.

Tools needed for the development environment for PDF2BMP are : ImageMagick, Cmake, Xerces, OpenCV and TCLAP.

#### **Interactive document conversion in MS Cloud**

As the first step towards a semi-automatic document preservation workflow, we have designed a simple interactive workflow for ingesting, converting, comparing, reporting and



analysing office documents.

MSR SCAPE solution comprises of an Authentication layer, Data management layer, Tool and resources layer. Actions that user performs are reflected in architecture presented in Figure 15.

Logging is important part of the solution, therefore we are automatically logging every operation within system processes, authentication, data and operation execution.

The workflow supports three main types of activities:

- management of the collections: importing, previewing, and selecting documents for conversion,
- conversion process: selecting the operators for the conversion,
- analysis of the converted data: selecting the comparison tools that provide conversion analysis and viewing the comparative reports.

More details about the steps that typically comprise an interactive conversion workflow are described in Figure 16.

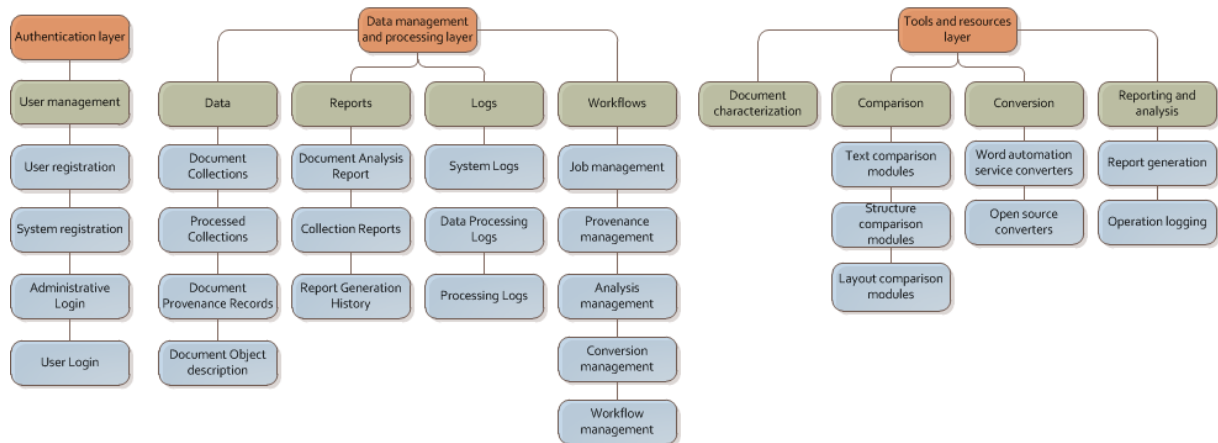


Figure 15: High level components of architecture - MSR SCAPE Cloud solution

Our further work will build on this workflow to include human input as part of the document comparison phase, first from trusted and qualified assessors and then through crowd-sourcing.

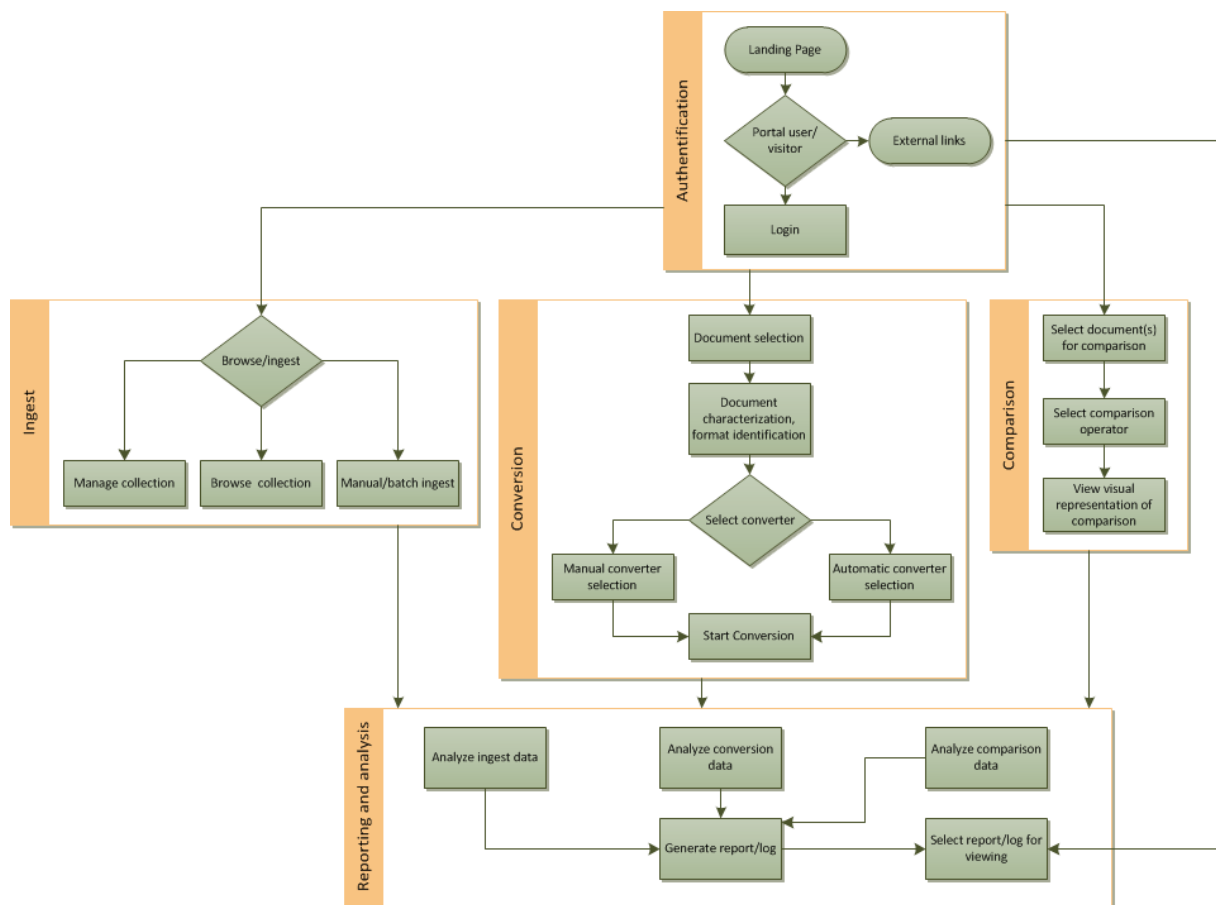


Figure 16: User interaction workflow - MSR SCAPE Cloud solution

Using MS tools (see Appendix B: Description of tools in MS working environment) we have designed a user interface that supports interactive document conversion and analysis workflow. In the following section we describe the basic concepts and features that support individual steps in the workflow.

### 6.5.2 UI for the Cloud Based Document Conversion and Analysis

We created a portal UI that supports the main three activities: uploading and organizing collections, conversion of documents, comparison and analysis of converters output.

After selecting collection of documents user wants to convert, documents are being uploaded (Figure 17 – 18). When ingest is done, system automatically preforms characterization in the background (Figure 19).

User automatically gets recommended conversion path, he can also select specific conversion(s), even conversion chain if there is more than one converter required for destination format (Figure 20). Conversion progress and comparison of two documents is presented in Figure 21 and Figure 22.

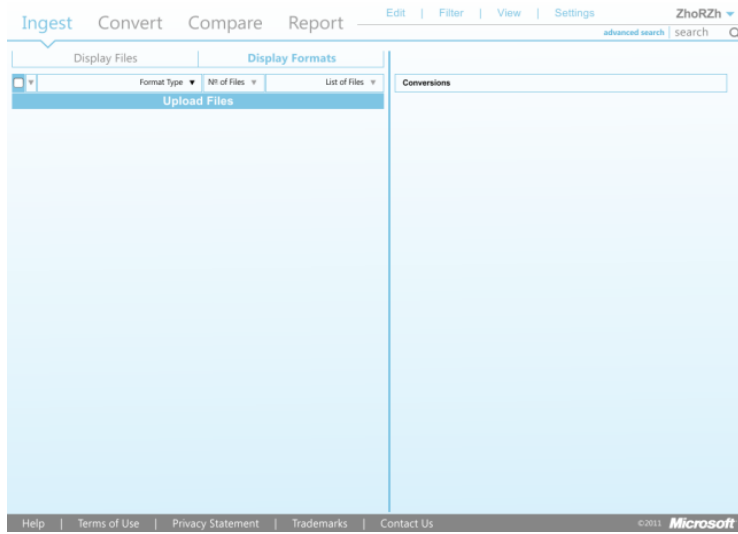


Figure 17: Pre ingest phase – empty collection

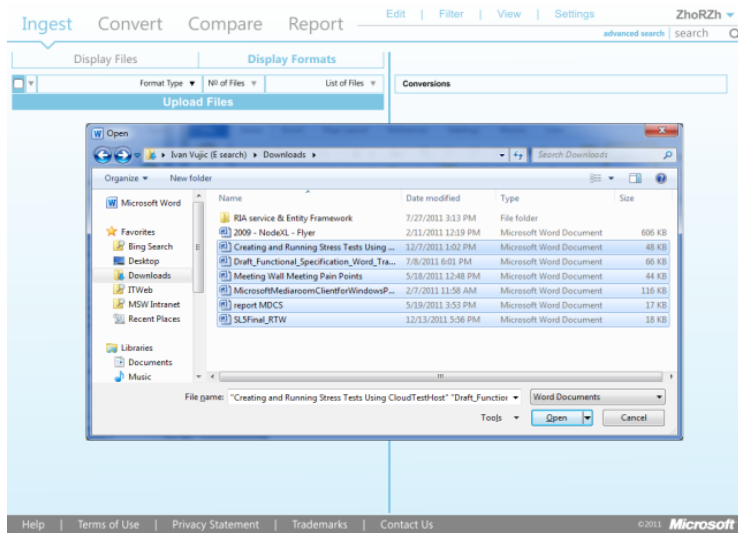


Figure 18: Ingest phase

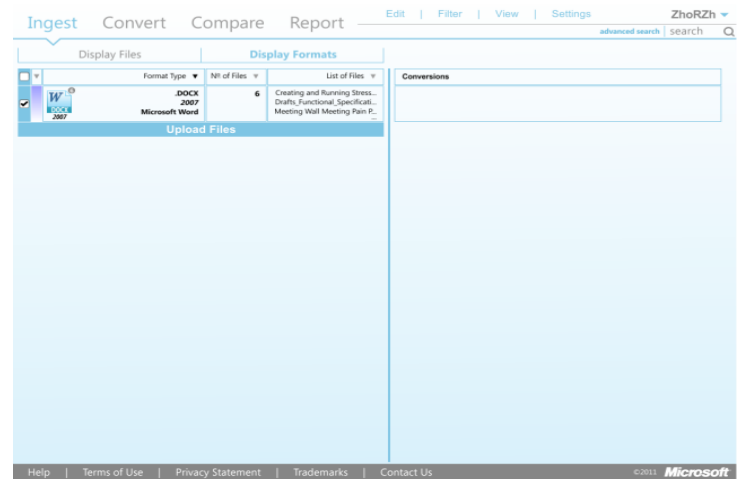


Figure 19: Background document characterizations

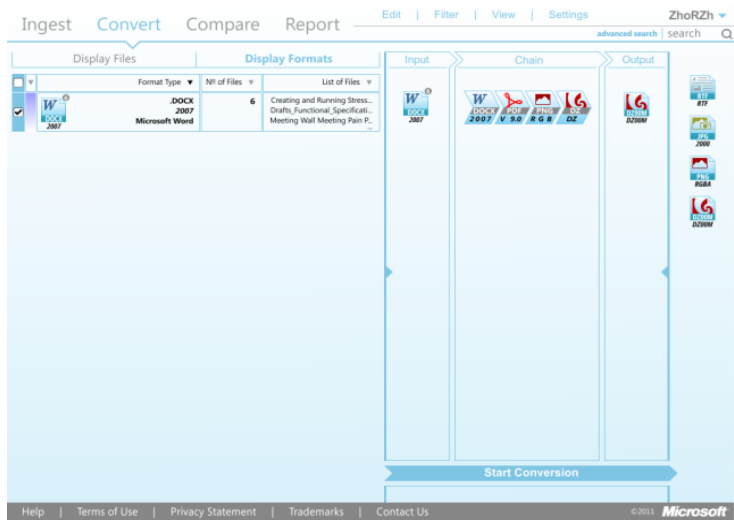


Figure 20: Converter, conversion path selection

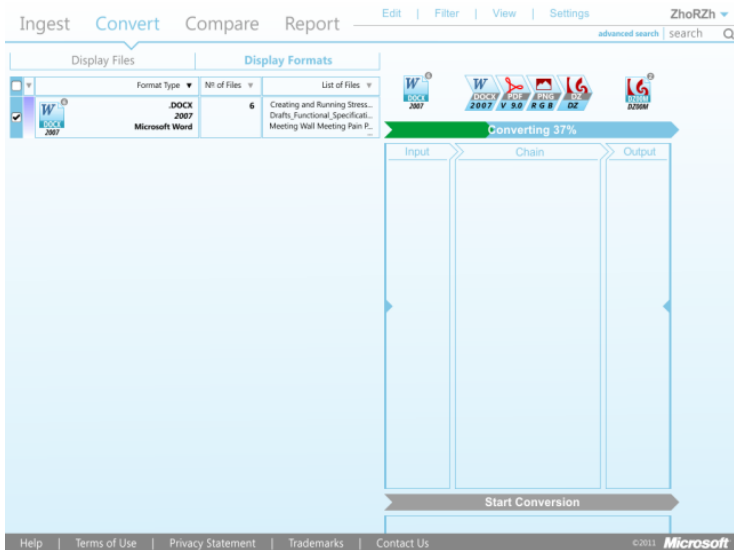


Figure 21: Conversion phase

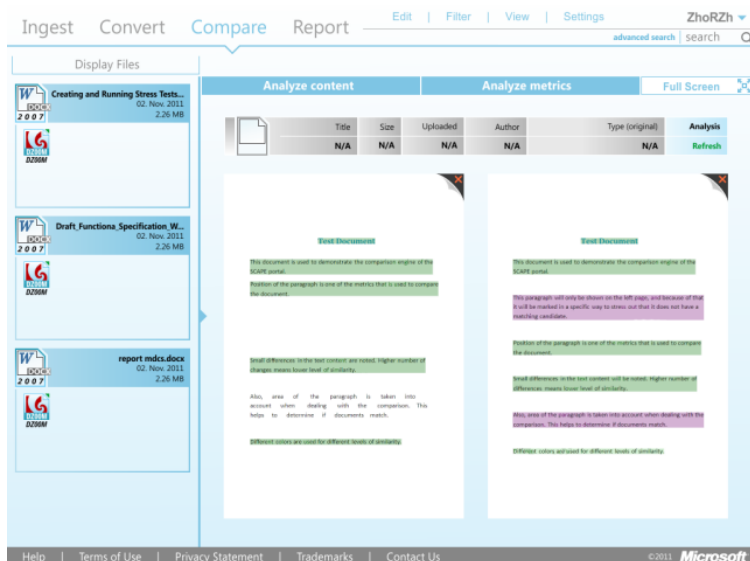


Figure 22: Comparison phase

Based on the initial sketches we created a portal UI that supports the main three activities: browsing and organizing collections, selecting tools and collecting documents, and analysing the output of the converters. The above figures shows a more refined design that will be implemented as the first UI for the system.

## 7 Image and QA tools

This section describes QA tools that are being developed in this work package.

### 7.1 Imaging Tools

Quality assurance (QA) is required for preservation actions involving conversion between image formats. The currently implemented QA tools "extractFeatures" and "compare" (see chapter DPQAlib Library) enable feature extraction from images and comparison of image files in common file formats, especially TIFF and JPEG2000.

From conversations with content providers and received data the following cases are identified for the addressed media:

- Conversion of images in TIFF format, e.g. containing newspaper or book scans, to lossless JPEG2000 format. Quality assurance compares image content before and after conversion, especially completeness and colour are verified.
- Conversion of images in TIFF or lossless JPEG2000 format to lossy JPEG2000 format. Quality assurance compares image content before and after conversion, in particular, differences in content are separated from compression artifacts.
- Conversion of images in TIFF format to lossless JPEG2000 format involving geometric modifications, e.g. cropping or undistortion. Quality assurance compares image content before and after conversion, in particular, differences in content are separated from geometric modifications.
- Conversion of images in TIFF format to lossless JPEG2000 format involving radiometric modifications, e.g. binarization or shading correction. Quality assurance compares image content before and after conversion, in particular, differences in content are separated from radiometric modifications.
- Digital Preservation Quality Assurance (DPQA) Tools

To address the specific needs of imaging QA, custom software was developed. The Digital Preservation Quality Assurance (DPQA) tool-set consists of a shared library *DPQAlib* and two command line tools *extractFeatures* and *compare*. These tools provide diverse functions to analyse and compare images concerning different preservation criteria. This analysis is divided into three levels concerning the degree of abstraction:

*Level 1:* Image metadata that can easily be retrieved from file or image format attributes is extracted and compared to detect obvious differences (e.g. dimension, colour channels)

*Level 2:* Standard image processing features (e.g. histograms, profiles) are calculated and compared. This level detects differences in toning as well as geometrically misalignments.

*Level 3:* Sophisticated features to detect similarities in images (e.g. is new image a cropped version of the old one without significant information loss?).

The functionality of the DPQA tools is currently very limited. Nevertheless, the software design is generic and further methods for quality assurance can and will be added in future.

These software packages were developed in a MS Visual Studio 2008 / MS Windows 7 environment. The implementation is based and restricted to ANSI C++, enabling the software to be compiled and executed on both Windows and Linux platforms. CMake meta-build files are used for cross-compilation.

Due to the cross compilation requirements, the Microsoft XML processing library could not be used and is substituted by the Apache Xerces-C++ XML parser library. For image processing demands the elaborated OpenCV library (see external tools Appendix A: External Tools) is used as well as the TCLAP library is used for command line arguments parsing.

## 7.1.1 DPQAlib Library

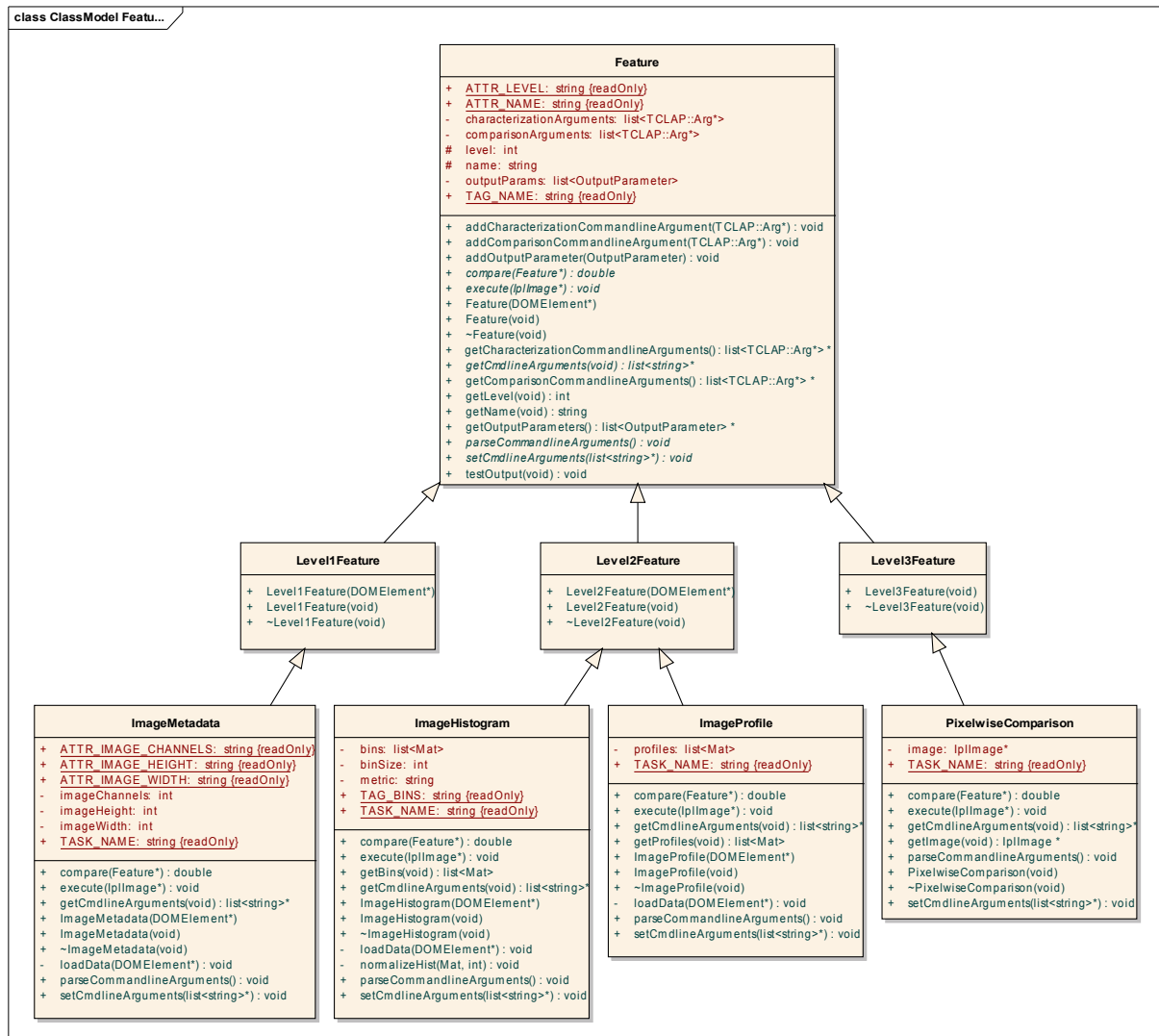


Figure 23: Class Diagram DPQAlib

DPQAlib is developed using object oriented paradigms and design patterns. The analysis level concept is implemented by means of a class hierarchy, see Figure 23. All implemented feature extraction algorithms derive from the *Feature* class and inherit common properties and methods. This concept unifies the feature extraction procedures and provides a flexible interface for further features.

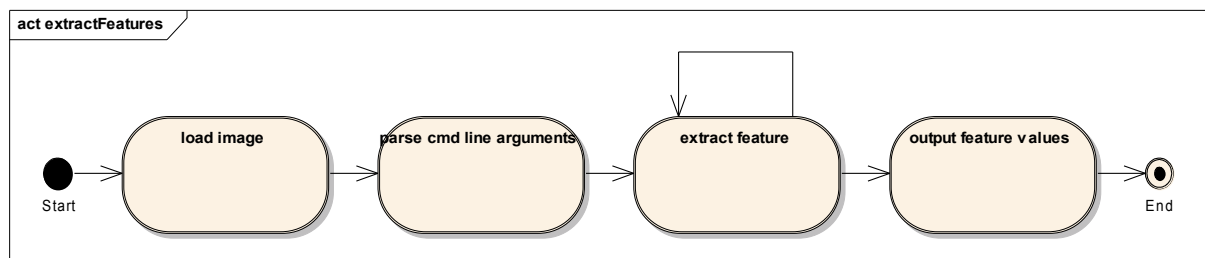


Figure 24: Class diagram for extractFeatures

Currently only the four features ImageMetadata, ImageHistogram, ImageProfile and PixelwiseComparison are implemented. Further features based on document and layout analysis are

currently investigated for their applicability.

All relevant functionality has been implemented in the library DPQAlib. The command line tools only invoke functions of this library and provide I/O operations as well as user interaction.

#### 7.1.1.1 *Command Line Tool: extractFeatures*

The command line tool *extractFeatures* takes a digital image as input. The program is based on the OpenCV library (see external tools Appendix A: External Tools) that can handle nearly every popular image format. This tool processes level one and two of the previously described threefold processing hierarchy.

Currently three features are implemented:

- ImageMetadata: only standard metadata is extracted (image dimension, color channels). Further attributes, especially jpeg related metadata, are currently being investigated.
- ImageHistogram: standard colour histograms. For each colour channel a histogram is calculated. By default the number of bins is equivalent to the number of colours in the RGB colour space (256). This value can be modified by the user through a command line argument.
- ImageProfile: standard image profile containing a horizontal and vertical projection of each colour channel.

Usage:

*extractFeatures* can be invoked with the following command:

```
extractfeatures.exe [--numbins <int>] [-n <string>] ... [--l2] [--l1] [--  
  [--version] [-h] <file>
```

By default all features are extracted with their standard parameters. To reduce the feature set the following parameters can be use:

- --l1: all level 1 features will be extracted.
- --l2: all level 2 features will be extracted.

Supplying one of these parameters reduces the features set to the demanded level and excludes all others. Providing both arguments results in the default settings of the tool. The feature set can be reduced further, by supplying the --no switch to exclude distinct features.

Output:

The results of the feature extraction are printed as XML formatted semi structured data onto standard output. It is intended that the output is piped into an XML file. Here is a sample output of "extractFeatures" tool with numbers parameter equals 5:

```
<?xml version="1.0" encoding="UTF-8"?>  
<characterization>  
<task level="1" name="imageMetadata">  
  <ImageWidth>1024</ImageWidth>  
  <ImageHeight>768</ImageHeight>  
  <ImageChannels>3</ImageChannels>  
</task>  
<task level="2" name="imageHistogram">  
  <bins dimension="1" numberofbins="5" range="0,255">0.567799, 0.0367724, 0.00389481,  
0.0070254, 0.384445</bins>  
  <bins dimension="2" numberofbins="5" range="0,255">0.412443, 0.16591, 0.0554899, 0.210939,  
0.155219</bins>  
  <bins dimension="3" numberofbins="5" range="0,255">0.193193, 0.179962, 0.254397, 0.30114,  
0.071214</bins>  
</task>  
</characterization>
```



### 7.1.1.2 Command Line Tool: compare

The tool *compare* is either responsible for comparing the results of the tool *extractFeatures* or for directly comparing images based on level three features. Consequently the tool takes either two digital images or two xml files as input.

Currently only a trivial one by one pixel comparison is implemented. Further features that analyse similarity between two images are currently being investigated.

Usage:

Compare can be invoked with the following command:

```
compare.exe [--metric <str>] [-l <int>] [--] [--version] [-h] <file1> <file2>
```

The standard image processing metrics in this command specify the calculation algorithm. <file1> and <file2> are XML files and are outputs of "extractFeatures" tool execution. Only output of "extractFeatures" can be used as an input for compare.exe tool. By default no parameters are required (beyond the two input files). It is assumed that xml files for level one or two will be compared. If a level three comparison is required the parameter -l 3 or --level 3 has to be supplied. The implemented features may also provide their own parameters. Currently only *ImageHistogram* defines a parameter for feature comparison. Here standard image processing metrics are used for histogram comparison, with only the following metrics being currently implemented:

CV\_COMP\_CHISQRCV\_COMP\_CORREL, CV\_COMP\_INTERSECT and CV\_COMP\_BHATTACHARYYA.

Output:

The result of the comparison is printed as XML formatted semi structured data onto standard output:

```
<?xml version="1.0"?>
```

```
<comparison>
```

```
<task level="1" name="ImageMetadata">
```

```
<result>0</result>
```

```
</task>
```

```
<task level="2" name="ImageHistogram">
```

```
<result>0.633491</result>
```

```
</task>
```

```
<task level="2" name="ImageProfile">
```

```
<result>20.3755</result>
```

```
</task>
```

```
</comparison>
```

### 7.1.2 Example Taverna Workflow

Figure 25 shows a Taverna workflow which utilises the two command line tools, *extractfeatures* and *compare*, to perform a comparison on two input images, for example for images before and after conversion. The output from this workflow indicates how similar the two input images are.

Some prior work has to be done before the DPQA tools can be used however. Both are written in C++ and provided as executables (DLLs on Windows, and Shared Objects on Linux), and so they have to be exposed as web services using the xa-toolwrapper

(<https://github.com/openplanets/scape/tree/master/xa-toolwrapper>) before they can be used within Taverna.

The workflow in a multistep process, first requiring both input images to undergo feature extraction (using the *extractfeatures* tool) before the features are compared (using the *compare* tool). Feature extraction expects a URL to the input image and returns an XML file with histogram data. After generating such XML files for both images, these can be used as input to the *compare* tool, which returns a comparison report, also in XML format. This report compares different characteristics to provide a normalised comparison value between 0 and 1, 0 being identical and 1 being completely different.

Workflow in Figure 25 is available for download, as many other, from myExperiment

<http://www.myexperiment.org/workflows/2579.html>.

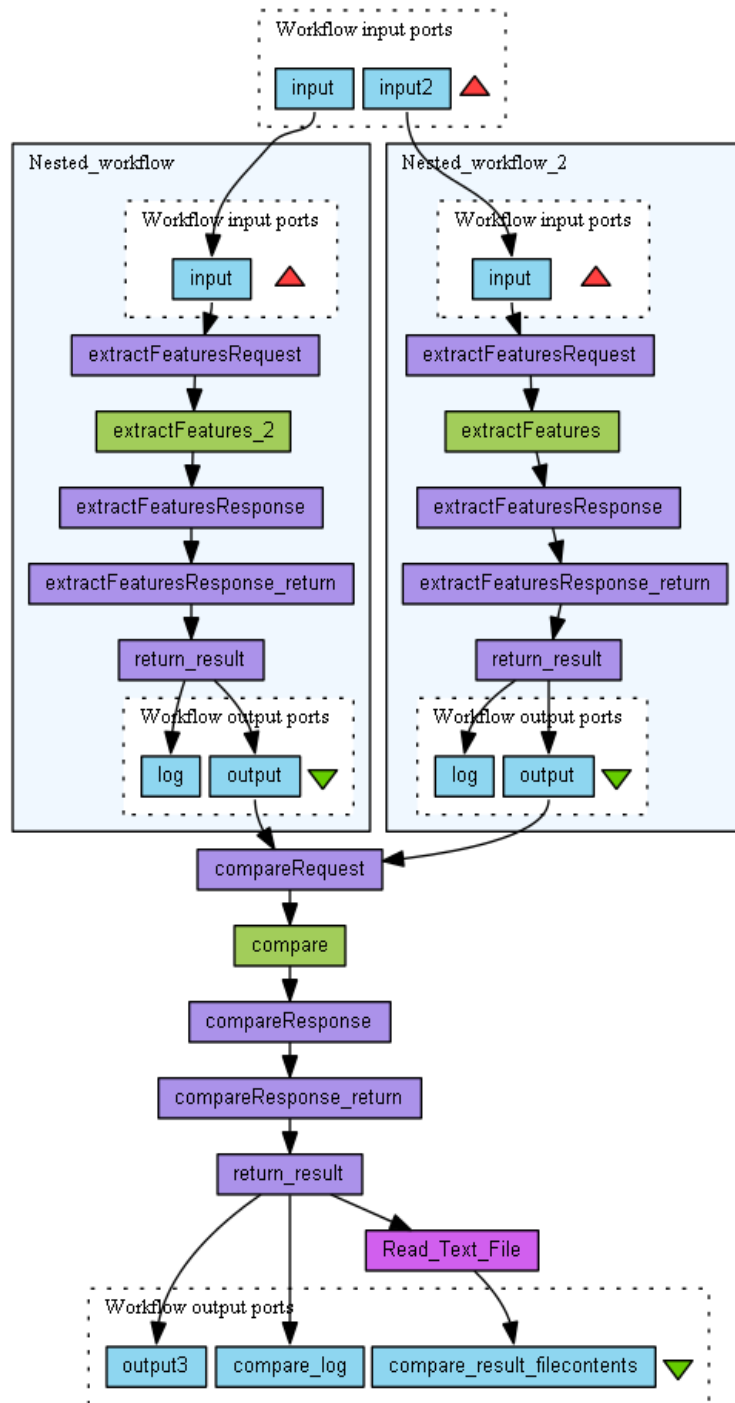


Figure 25: Taverna Workflow for extractfeatures and compare tools

## 7.2 "QA Tool" Analysis Tool

Quality assurance (QA) is required for preservation actions, in order to ensure that the action has been executed successfully. In general, this is done by analysing the input data and the output data, i.e. the media, and comparing the two objects against each other and against any relevant standards. The QA task described here is different. Instead of analysing the inputs and outputs, we instead analyse the underlying software process itself. By exploring the execution of the preservation action in detail, we make it possible to understand how the underlying tool processes the data in a much more general sense than when comparing specific input/output pairs. Rather, we are performing QA upon the tool itself, and by increasing our confidence in the quality of the tool we increase our confidence in any actions processed using that tool.

While the approach is quite different to the other tasks in this work package, it still requires test data and scenarios in order to ensure that it is testing the most relevant features of the tools. Therefore, after having looked at the Testbed scenarios and the available data, the Tool QA Workflows will be developed to analyse the following processes:

- Conversion of images in TIFF format, (e.g. containing newspaper or book scans), to lossless JPEG2000 format.
- Conversion of images in TIFF or lossless JPEG2000 format to lossy JPEG2000 format.

However, the approach is quite general, and can be used to explore all sorts of preservation processes. Therefore, once these initial cases have been investigated, the approach will be expanded to explore other cases.

### 7.2.1 Bitwiser

The Bitwiser tool works by systematically flipping every single bit in the given input file, and reiterating the given tool for each flipped bit: the tool execution is then monitored and compared with the execution of the same tool using the original input file. The responses of the tool can be classified as:

1. No output created:
  - a) Tool exits gracefully.
  - b) Tool crashes.
2. Output created:
  - a) Output file differs (i.e. that bit was noticed).
  - b) Output file matches original (i.e. that bit was ignored).

The set of bits that lead to states 1(a), 1(b) and 2(a) are all 'covered' by the tool, i.e. they are all 'noticed' and alter the execution. In particular, all bits classified under 2(a) are bits that count directly towards the output file. The set of bits that lead to state 2(b) appear to be ignored by the tool. This list of ignored bits can then be investigated further, to see if this implies that significant information has been discarded.

#### 7.2.1.1 Interface

The tool has a relatively simple command-line interface, taking a command-line template (tool.ptspec) from one file, and another file as input data.

```
% bitwiser -ocm coveragemap.png tool.ptspec input.data
```

The tool will output XML that contains the properties in a standard format, and create and output an image for the coverage map called 'coveragemap.png'.

### 7.2.2 Example Taverna Workflow

Figure 26 below shows what the overall dataflow looks like, and how the tool would appear in Taverna.

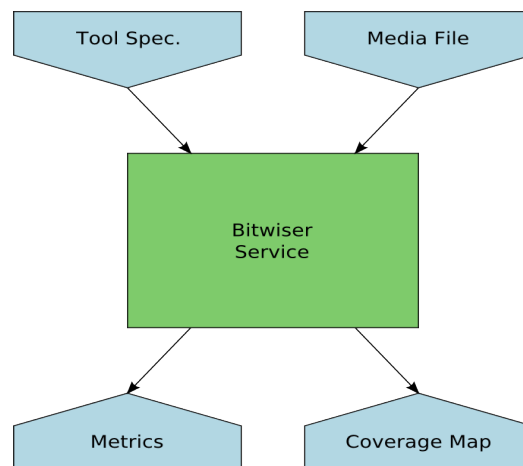


Figure 26: Schematic Taverna Workflow for the Bitwiser tool

In terms of the dataflow, the process looks very simple. Given a tool specification (essentially a command-line template showing how the tool should be invoked) and an example input media file, the Bitwiser service does a deep analysis of the process of running the given tool with the given input. This leads to the creation of some metrics that describe the process, including an estimate of what fraction of the source files bits are taken into account when the tool runs. This output is accompanied by a coverage map, which is a visual representation of how the process interprets the bits.

### 7.2.3 Status

The tool is currently at the proof-of-concept stage, and contains code that works but requires refinement. It also needs tests and the addition of the command line interface.

## 8 Knowledge base

Digital file objects may have features that can be a threat to long-term access. Examples are: print or copy restrictions in a PDF file may prevent a future migration; incorrectly formatted JPEG 2000 images may be perfectly readable, but if references to embedded ICC profiles are implemented in some non-standard way, this may result in the loss of these ICC profiles during some future migration. In many cases, content holders may be completely unaware of the risks involved, and such issues may not be addressed by existing control policies.

The Knowledge Base task addresses this situation by creating a knowledge base of known 'problematic' (from a preservation point of view) features for a limited number of file formats. This also includes ways to detect such features with existing characterisation tools. The knowledge base is implemented as a Wiki that is part of the OPF registry that also hosts the SCAPE scenarios, issues, solutions and tools. Although these are all interlinked, the scope of the work that is done in QA is limited to format-specific risks. Thus far, an initial version of the knowledge base has been set up, containing entries for the JP2 and JPX (JPEG 2000) formats.

The current version of the knowledge base is accessible through the following link:

<http://wiki.opf-labs.org/display/TR/Formats>

## 9 Roadmap for Next Iteration and conclusions

The plan for the next year is to move focus from availability to scalability. This means that we would like to have the workflows running locally on a SCAPE platform rather than using web services, and hopefully we can get performance that scales to the actual dataset sizes in the issues.

Work this year has highlighted some improvements that can be made to error handling. How to do this though depends on how we expect to use Taverna. This is a decision yet to be made.

We plan to widen the scope from audio to audio and video. We expect the *LSDR T4 Out-of-sync Sound and Video in wmv to Video Format-X Migration Results* [LSDR T4] to be able to use some of the same tools used in *LSDR T6 Migrate mp3 to wav* (this report). *LSDR T7 Characterise and Validate very large video files* [LSDR T7] (*IS22 Characterise and Validate very large mpeg-1 and mpeg-2 files*) has a validation aspect like migration issues, and we may be able to use some of the same tools and workflows again. As mentioned in chapter Audio QA workflows *LSDR T5 Detect audio files with very bad sound quality* [LSDR T5] is also an audio QA scenario, and we hope to be able to use the work done in the migrationQA tool to write a solution fitting this scenario. We have not prioritised the different scenarios, performance work and work in error handling yet.

The Bitwiser tool, as mentioned in section Status, is currently at the proof-of-concept stage, requiring further development and refinement to its code, along with any appropriate Taverna workflows.

There is also potential to improve performance through parallelisation by running many instances of the Bitwiser tool across subsections of the input file and combining the results.

Entries for the JP2 and JPX (JPEG 2000) formats will be elaborated further in 2012 (in tandem with the work on the *jpylyzer* tool, which is able to detect most problematic features). Entries for PDF will be added as well. Here, the list of features that are not allowed in the PDF/A-1 profile (and how to detect these) will serve as a starting point.

## 10 Bibliography

- [heritrix] Heritrix website. [Online] Link: <http://crawler.archive.org/index.html>
- [VKJ2011] Van der Knijff, Johan. "Jpylyzer: validator and properties extractor for JPEG 2000 Part 1" (JP2), User Manual. December 2011.
- [taverna] Taverna github. [Online]  
Link: <http://www.myexperiment.org/groups/490.html>
- [xatoolwrapper] Taverna github. [Online]  
Link: <https://github.com/openplanets/scape/tree/master/xa-toolwrapper>
- [XC\*L] XC\*L website. [Online]  
Link: [http://planetarium.hki.uni-koeln.de/planets\\_cms/about-xcl](http://planetarium.hki.uni-koeln.de/planets_cms/about-xcl)
- [WARC] ARC website. [Online]  
Link: <http://www.archive.org/web/researcher/ArcFileFormat.php>
- [Wor\_FFmpeg] Workflow Entry: FFmpeg mp3 to Wav Migration SCAPE Web Service Workflow. [Online] Link: <http://www.myexperiment.org/workflows/2713.html>.
- [Wor\_FFprobe] Workflow Entry: FFprobe Audio Files Property Extraction and Comparison Workflow. [Online]  
Link: <http://www.myexperiment.org/workflows/2715.html>.
- [Wor\_JHOVE2] Workflow Entry: JHOVE2 Wav Validation SCAPE Web Service Workflow. [Online] Link: <http://www.myexperiment.org/workflows/2637.html>.
- [Wor\_migrationQA] Workflow Entry: migrationQA SCAPE Web Service Wav File Comparison Workflow. [Online]  
Link: <http://www.myexperiment.org/workflows/2717.html>.
- [Wor\_MockUp] Workflow Entry: Mock-Up mp3 To Wav Migrate And QA. [Online]  
Link: <http://www.myexperiment.org/workflows/2639.html>.
- [Wor\_Basic] Workflow Entry: Mp3 to Wav Migrate Validate Compare . [Online]  
Link: <http://www.myexperiment.org/workflows/2687.html>.
- [Wor\_MPG321] Workflow Entry: MPG321 Play mp3 to Wav SCAPE Web Service Workflow. [Online]  
Link: <http://www.myexperiment.org/workflows/2721.html>.
- [LSDRT4] LSDRT4 Out-of-sync Sound and Video in wmv to Video Format-X Migration Results [Online].  
Link: <http://wiki.opf-labs.org/display/SP/LSDRT4+Out-of-sync+Sound+and+Video+in+wmv+to+Video+Format-X+Migration+Results>.
- [LSDRT5] LSDRT5 Detect audio files with very bad sound quality [Online].  
Link: <http://wiki.opf-labs.org/display/SP/LSDRT5+Detect+audio+files+with+very+bad+sound+quality>.
- [LSDRT6] LSDRT6 Migrate mp3 to wav [Online].  
Link: <http://wiki.opf-labs.org/display/SP/LSDRT6+Migrate+mp3+to+wav>.
- [LSDRT7] LSDRT7 Characterise and Validate very large video files [Online].

Link: <http://wiki.opf-labs.org/display/SP/LSDRT7+Characterise+and+Validate+very+large+video+files>.

- [PW.WP.1\_D12.1] SCAPE Deliverable: Identification of triggers and preservation Watch component architecture, subcomponents and data model
- [BSGP2009] M. Ben Saad, S. Gancarski, and Z. Pehlivan. A novel web, archiving approach based on visual pages analysis. In IWAW 2009.
- [SS2001] L. G. Shapiro and G. C. Stockman. *Computer Vision*. Prentice Hall 2001. ISBN 978-0130307965.
- [CYWM2003] D. Cai. S. Yu. J.R. Wen. W.Y. Ma. VIPs: a Vision-based Page Segmentation Algorithm. Nov. 1, 2003. Technical Report. MSR-TR-2003-79
- [HW1979] Hartigan, J. A.; Wong, M. A. (1979). "Algorithm AS 136: A K-Means Clustering Algorithm". *Journal of the Royal Statistical Society, Series C (Applied Statistics)* 28 (1): 100–108
- [L1999] Lowe, David G. (1999). "Object recognition from local scale-invariant features". *Proceedings of the International Conference on Computer Vision*. 2. pp. 1150–1157.



## Appendix A: External Tools

### ImageMagick

The ImageMagick is free software which license is compatible with GNU LPG. This tool includes a set of utilities on command line, allowing converting PDF file to a rendered image file like BMP.

Version: 6.4.9-1  
Homepage: <http://www.imagemagick.org/script/index.php>  
License: Freeware

### CMake

CMake is a cross-platform, open-source build system. Compiler assignments are expressed through annotations and meta-expressions in simple text files. According to these meta-build descriptions CMake generates specific build files for each platform dependent build system (e.g. Makefile for Linux, Visual Studio project files for Windows, etc.). CMake is able to detect and solve library dependencies. It can be invoked on the command line, but a graphical user interface is also provided.

Version: 2.8.4  
Homepage: <http://www.cmake.org/>  
License: Creative Commons Attribution-NoDerivs 3.0 Unported License

### Xerces

Xerces is a validating XML parser available for C++, Java and Perl. The library provides easy means to read and write XML data.

Version: 3.1.1  
Homepage: <http://xerces.apache.org/xerces-c/>  
License: Apache Software License, Version 2.0.

### OpenCV

The Open Source Computer Vision (OpenCV) library provides functions for real time computer vision developed under the BSD license. For this project the new C++ interface is used.

Version: 2.2  
Homepage: <http://opencv.willowgarage.com/wiki/>  
License: BSD License

### TCLAP

The Templated C++ Command Line Parser Library (TCLAP) is a small library for command line parsing.

Version: 1.2.1  
Homepage: <http://tclap.sourceforge.net/>  
License: MIT License

### Silverlight 4 Tools

Silverlight is a development platform for creating interactive user experiences for Web, desktop, and mobile applications when online or offline. Silverlight is a plug-in, powered by the .NET framework and compatible with multiple browsers, devices and operating systems. Silverlight 4 adds features, like webcam, microphone, and printing.

Version: 4  
Homepage: <http://www.silverlight.net/>  
License: n/a

### Expression Blend (optional)

Microsoft Expression Blend is a user interface design tool for creating graphical interfaces for web and desktop applications that blend the features of these two types of applications. It is an interactive, WYSIWYG front-end for designing XAML-based interfaces for Windows Presentation Foundation and Silverlight applications. It is one of the applications in the Microsoft Expression Studio suite. Expression Blend supports the WPF text engine with advance OpenType typography and ClearType, vector-based 2D widgets, and 3D widgets with hardware acceleration via DirectX.

Version: 4  
Homepage: [http://www.microsoft.com/expression/products/Blend\\_Overview.aspx](http://www.microsoft.com/expression/products/Blend_Overview.aspx)  
License: n/a

**Silverlight toolkit**

Silverlight Toolkit is open source project that shares new components and functionality for designers, developers, and the community to provide an efficient way to help shape product development. Toolkit releases includes full open source code, samples, documentation, and design-time support for controls focusing on both Silverlight 4 as well as the Windows Phone.

Version: February 2011  
Homepage: <http://silverlight.codeplex.com/>  
License: Microsoft Public License (Ms-PL)

**Windows Azure Tools & SDK**

The Windows Azure Platform is a Microsoft cloud platform used to build, host and scale web applications through Microsoft datacenters. Windows Azure Platform is thus classified as platform as a service and forms part of Microsoft's cloud computing strategy, along with their software as a service offering, Microsoft Online Services. The platform consists of various on-demand services hosted in Microsoft data centers: Windows Azure (an operating system providing scalable compute and storage facilities), SQL Azure (a cloud-based, scale-out version of SQL Server) and Windows Azure AppFabric (a collection of services supporting applications both in the cloud and on premise).

Version: April 2011  
Homepage: <http://www.microsoft.com/windowsazure/sdk/>  
License: n/a

## Appendix B: Description of tools in MS working environment

### Silverlight

The *Silverlight* application can communicate directly with the BLOB (Binary Large Object) storage system (primarily for file upload) via a REST API giving the advantage of direct communication with the storage space without having to go via a service on the server.

The typical way to access BLOB storage is to construct a web request and then sign that request with the storage account's shared key.

This model is fine when the shared key is embedded within trusted code however, placing the shared key within a *Silverlight* application makes it possible for anyone to extract the key and take full control of the storage account. One option is to proxy all requests that come from the client via a web service; however the downside is that all requests must be transferred via this service which could potentially become a bottleneck.

The solution to this problem is to use *Shared Access Signatures* (SAS) which allow for access rights to be provided to containers and blobs at a more granular level than by simply setting a container's permissions for public access. Using a SAS allows for the separation of code that signs the request from the code that executes it and users can be granted access to a specific blob or any blob within a specific container for a specified period of time.

### WCF RIA Services

WCF RIA Services are used for data management (CRUD – Create Read Update Delete) and user authentication, the data store being an instance of SQL Azure.

User authentication is provided by the ASP.NET membership system and the RIA authentication client is configured to use this via forms authentication.

Worker roles are used to self-host WCF services which perform conversion / OCR / pagination (much like a console application or windows service). It should be noted that the TCP protocol cannot be used if Web Roles (IIS7 and SVC endpoints) are used to host the service so the speed gains in exchange for the management of the service is acceptable in this situation. The endpoint types are Internal and as such do not hook into the load balancer so a simple randomiser is used to select the endpoint to use – Input endpoints would require security and be externally facing.

### Local storage

Is file system directory on the server on which a particular Worker role instance is running – the contents of the local storage are not persistent across instance failures and restarts as the role may be redeployed to another virtual machine due to a failure or restart. Local storage cannot be shared between role instances, but it is faster than other storage mechanisms as it resides on the same VM as the role instance is running. The size is limited to the size of the VM as specified for the service model (smallest is 250GB) which can also be configured via the property pages for the role.

A possible alternative to using the local storage system is to make use of Windows Azure Drive which allow for Azure applications to use existing NTFS APIs to access a durable drive in the cloud. They are implemented as Azure Page Blob containing an NTFS-formatted Virtual Hard Drive (VHD) and since the drive is an NTFS formatted page blob the standard blob interfaces can be used to upload / download the VHDs to the cloud, speed however would be a consideration for this scenario.

### Windows Azure

Supports native execution allowing role hosts to PInvoke into system Dll's or to a proprietary packaged native DLL. There is however a caveat to this in that Windows Azure runs on a 64bit OS and role hosts (both worker and web) are also 64 bit executable and as such will run as 64 bit binaries. This means that to PInvoke to a native DLL it must be compiled for a 64 bit platform; this however is not always an option (for example if the original source code is not available). The solution described here is to host the 32bit DLL in a 32bit program that runs on WoW (windows on windows), WCF is then used to marshal a call between the role process and the DLL. The two processes can both access local storage as they are executing on the same VM and communicate via named pipes for the same reason. Note that at the time of writing in order for WCF to set up a local communication pipe the hosting application must be built against .NET 3.5.

## Appendix C: Description of tools in the development environment

### **PDF2BMP**

Tested OS: Windows 7, UBUNTU 11

IDE: MS Visual Studio 2008, Cmake

External Libraries: Xerces 3.1.1, OpenCV 2.2, TCLAP

The software package is developed in a MS Visual Studio 2008 / MS Windows 7 environment. CMake meta-build files are used for cross-compilation. The implementation is based and restricted to ANSI C++. Thus, the software can be compiled and executed on both Windows and Linux platforms.

Due to the cross compilation requirements, the Microsoft XML processing library could not be used and is substituted by the Apache Xerces-C++ XML parser library. For image processing demands the elaborated OpenCV library is used as well as the TCLAP library is used for command line arguments parsing.

### **Microsoft Office documents**

Tested OS: Windows 7

IDE: MS Visual Studio 2010

External libraries: Silverlight Toolkit

Visual Studio, Silverlight 4 Tools, Silverlight Toolkit and WCF RIA Services will need to be installed before solution is deployed. The solution can be run locally within the Azure development environment if a live implementation is not necessary or desirable, also configuring a local SQL server to provide membership information if this option is taken (although it is also possible to make use of SQL Express with VS2010). Please refer to this article for more details on setting up the membership schema in SQL server:

<http://www.asp.net/security/tutorials/creating-the-membership-schema-in-sql-server-vb>.

There is however one serious problem with using the development environment in combination with Silverlight and REST.