| | |
|---|---|
| Project Number | IST-2006-033789 |
| Project Title | Planets |
| Title of Deliverable | Validation Framework |
| Deliverable Number | **PP5/D4** |
| Contributing Sub-project and Work-package | SP/PP/5 |
| Deliverable Dissemination Level | External<br>Planets All |
| Deliverable Nature | Software and Release Report |
| Contractual Delivery Date | 31st March 2010 |
| Actual Delivery Date | |
| Author(s) | Volker Heydegger<br>Christoph Becker |

**Keyword list**

Digital Preservation, Preservation Planning, Tool support, Objectives, Requirements, Criteria, Significant Properties, Technical characteristics, File format characteristics, Migration, Extraction, Characterisation, XCL, XCEL, XCDL, Extractor, Comparator, Ontology, Evaluation, Validation, Framework

## Contributors

| Person | Role | Partner | Contribution |
|---|---|---|---|
| Volker Heydegger | Author (lead) | UzK | 1.3, 2.2 (main contribution) |
| Christoph Becker | Author | TUW | 2.1, 2.3, 2.4 (main contribution) |

## References


| Ref. | Document | Date | Details and Version |
|---|---|---|---|
| 1 | Report on methodology for specifying preservation plans | 31st July 2007 | Planets PP4/D1 report<br>available at http://www.planets-project.eu/publications |
| 2 | Christoph Becker, Hannes Kulovits, Mark Guttenbrunner, Stephan Strodl, Andreas Rauber, Hans Hofman. Systematic Preservation Planning: Evaluating potential strategies and building preservation plans | December 2009 | International Journal on Digital Libraries (IJDL). DOI: 10.1007/s00799-009-0057-1 |
| 3 | David Tarrant, Steve Hitchcock, Les Carr: Where the Semantic Web and Web 2.0 meet format risk management: P2 registry. | October 2009 | Contribution to: The Sixth International Conference on Preservation of Digital Objects (iPres 2009).<br>Available at http://eprints.ecs.soton.ac.uk/17556/ |
| 4 | PRONOM Unique Identifiers. The National Archives. | | website available at http://www.nationalarchives.gov.uk/aboutapps/pronom/puid.htm |
| 5 | Specification of Basic Metrics and Evaluation Framework | May 2007 | Planets PP5/D1 report<br>Available at: http://planetarium.hki.uni-koeln.de/planets_cms/deliverables |
| 6 | Christoph Becker, Hannes Kulovits, Andreas Rauber, Hans Hofman. Plato: a service-oriented decision support system for preservation planning. | June 2008 | In: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL'08). Pittsburgh, Pennsylvania, June 16-20, 2008. (accepted for publication) |
| 7 | White Paper: Representation Information Registries | January 2008 | Planets PC3/D7 report<br>available at http://www.planets-project.eu/docs/reports/Planets_PC3-D7_RepInformationRegistries.pdf |
| 8 | Christoph Becker, Hannes Kulovits, Michael Kraxner, Riccardo Gottardi, Andreas Rauber, and Randolph Welte. Adding quality-awareness to evaluate migration web-services and remote emulation for digital preservation | September 2009 | Proceedings of the 13th European Conference on Digital Libraries, ECDL 2009. LNCS 5714, Springer. |
| 9 | Manfred Thaller (Ed.): The eXtensible Characterisation Languages - XCL | May 2009 | In: Kölner Beiträge zu einer geisteswissenschaftlichen Fachinformatik, Vol.3 (1st edition), Hamburg: Dr.Kovac, 2009. (English language)<br>Also published as Planets PC2/D12, PC2/D13, PC4/D7 report:<br>eXtensible Characterisation Language Suite<br>Available at : http://planetarium.hki.uni-koeln.de/planets_cms/deliverables<br>(The 2nd edition of both sources will be released by the end of the Planets project, May 2010.) |
| 10 | Jan Schnasse, Sebastian Beyl, Elona Chudobkaite, Volker Heydegger, Manfred Thaller: XCL: The Extensible Characterisation Languages - One Step towards an Automatic Evaluation of Format Conversions: ECDL 2008: 444-446 | September 2008 | In: Research and Advanced Technology for Digital Libraries, Volume 5173/2008, pp. 444-446 (=Proceedings of the 12th European Conference on Digital Libraries, ECDL 2008)<br>DOI: 10.1007/978-3-540-87599-4 |
| 11 | PP4-D3 Report on service integration in Plato2. Planets external deliverable | May 2008 | Available at http://www.planets-project.eu/docs/reports/Planets_PP4-D3_ReportonServiceIntegrationInPlato-final.pdf |
| 12 | Manfred Thaller, Sebastian Beyl, Elona Chudobkaite, Volker Heydegger, Jan Schnasse: Significant Characteristics to Abstract Content: Long Term Preservation of Information. | September 2008 | In: Research and Advanced Technology for Digital Libraries, Volume 5173/2008, pp. 41-49 (=Proceedings of the 12th European Conference on Digital Libraries, ECDL 2008) |

| | | | |
|---|---|---|---|
| | | | DOI: 10.1007/978-3-540-87599-4_5 |
| 13 | Final version of Service Developers Guidelines | March 2010 | Planets IF6-D3 report<br>available at http://www.planets-project.eu/docs/reports |

# EXECUTIVE SUMMARY

This document is the release report of the validation framework developed in the Planets project, workpackage PP5. The validation framework connects the practical outputs of other Planets work-packages, mainly those of PP4, PC2 and PC4, within a common framework: Preservation requirements and criteria as they are defined in the objective trees in PP4 are mapped to the technical characteristics described by XCL in PC2/4, and to other criteria such as performance and format risks.

The validation framework consists of:

1. Comparison measures as basis for computing simple measurements or aggregated measurements over several low-level characteristics
2. A mapping mechanism for connecting criteria to measurements related to technical characteristics of both digital objects and tools,
3. Extension of the measurements to cover other measurable criteria of interest such as format risks and runtime performance.

Aspects (1) and (2) were initially described in report PP5-D1, *Specification of basic metric and evaluation framework* [5]. In the final version they are fully deployed within Planets software components. (1) is a fundamental part of the Comparator tool, related to the XCL and its components. (2) and (3) are part of the Planets Preservation Planning Tool (Plato). Since all parts are strongly bound to Plato and the Comparator, the specifications of the comparison measures and the evaluation framework are also integrated in the documentation part of these tools[1].

[1] Available at http://planetarium.hki.uni-koeln.de/planets_cms/ and http://www.ifs.tuwien.ac.at/dp/plato/

This document briefly summarizes central issues of the validation framework and describes the most important advancements of its parts: The comparison measures have been expanded in order to enable comparison of text related information and to give a summary result, i.e. indicating if a migration of content A to content B has been successful or not. The evaluation framework is now integrated into Plato so that it extends to other technical aspects of interest, such as run-time performance and format risks. To this end, we have developed a prototype of a quality-aware migration engine (3) that is shortly described.

# TABLE OF CONTENTS

# 1. Validation Framework

## 1.1 Introduction

A variety of tools performing preservation actions such as migration or emulation exist today; most often, there is no clearly optimal solution, however. The complex situations and requirements that need to be considered when deciding which solution is best suited for a given collection of objects make this decision a complex one. Preservation planning aids in the decision-making process by evaluating available solutions against clearly defined and measurable criteria. This evaluation needs verification and comparison of documents and objects before and after migration, or during emulation, to be able to judge migration quality in terms of defined requirements. It thus has to rely on an analysis of the logical structure of objects that is able to decompose documents and describe their content in an abstract form, independent of the file format. Especially considering migration actions working on large numbers of objects, it is essential to validate the authenticity of transformed objects automatically. When migrating a million documents from ODF to PDF/A, validation of these objects cannot be done manually.

When comparing the content of two files stored in two different formats, we have to distinguish between the abstract content and the way in which it is wrapped technically. On a very abstract level, this will be impossible for a long time: whether an image of a hand-written note contains the same 'information' as a transcription of that note in UTF-8 is philosophically interesting, but scarcely decidable on an engineering level. In a more restricted way, a solution is possible if we express the content stored in different file formats in terms of an abstract model of that type of content.

The eXtensible Characterisation Languages (XCL) described in [9] support the automatic validation of document conversions and the evaluation of conversion quality by hierarchically decomposing documents from different sources and representing them in an abstract XML language. The extraction language XCEL allows the Extractor component to extract the content of any document provided in a format for which an XCEL specification exists. The content is described in the description language XCDL and can thus be compared to other documents in a straightforward way. This differentiates the XCL approach from the approach used by JHove and similar projects. The XCL does not only attempt to extract a set of characteristics from a file, but it proposes to additionally express the complete informational content of a file in a format-independent model. Thus it supports the comparison of objects from different sources.

However, the question remains in which way the technical characteristics extracted from files should be compared, and how to interpret comparison results. This document outlines the main building blocks that are needed to answer these questions.

From the bottom-up, objects are characterised and based on these technical characteristics expressed in the XCDL comparison metrics are computed. From the top, objectives for preservation are defined, requirements that a preservation strategy must fulfil. A large part of these refers to significant properties of the objects in question. These properties are broken down according to common patterns, such as the five aspects 'content, appearance, structure, behaviour, and context'.

Still, a gap remains between these characteristics that come from the intellectual understanding of objects and the technical characteristics extracted from files. This gap is closed from one side by the comparison metrics mentioned above. The bridge from the other side consists of a mapping between the requirements and the comparison metrics that connects the two trees (objective tree from preservation planning; tree of extracted characteristics from characterisation) and thus supports the automatic evaluation and validation of preservation actions.

Figure 1 schematically illustrates the conceptual layers of the validation framework and their relationships, putting them in context with other work done in Planets. On the top, PP4 defines requirements using the objective tree structure. These requirements deal in large parts with significant properties of objects. On the leaf level of the tree, criteria such as the following may be found:

- *The textual content of all documents shall be unchanged,*
- *The resolution of all images in all web pages must not change,* or
- *The Table of Content (TOC) must be preserved*.

The lower the level in the tree, the more the objectives move from abstract intellectual concepts to concrete, technically extractable and measurable criteria. Some are computable in a straight-forward way, such as the width of images in pixels; others are more complicated. Still, it can be imagined that an algorithm could be developed that is able to, for example, characterise a document in such a way that the textual content can be compared across different file formats, or an algorithm that compares the structure and content of tables across documents.
This is where, from bottom-up, the eXtensible Characterisation Languages comes in, extracting technical characteristics from the actual objects. The eXtensible Characterisation Languages essentially decompose objects and describe their content in an abstract form, independent of the file format.

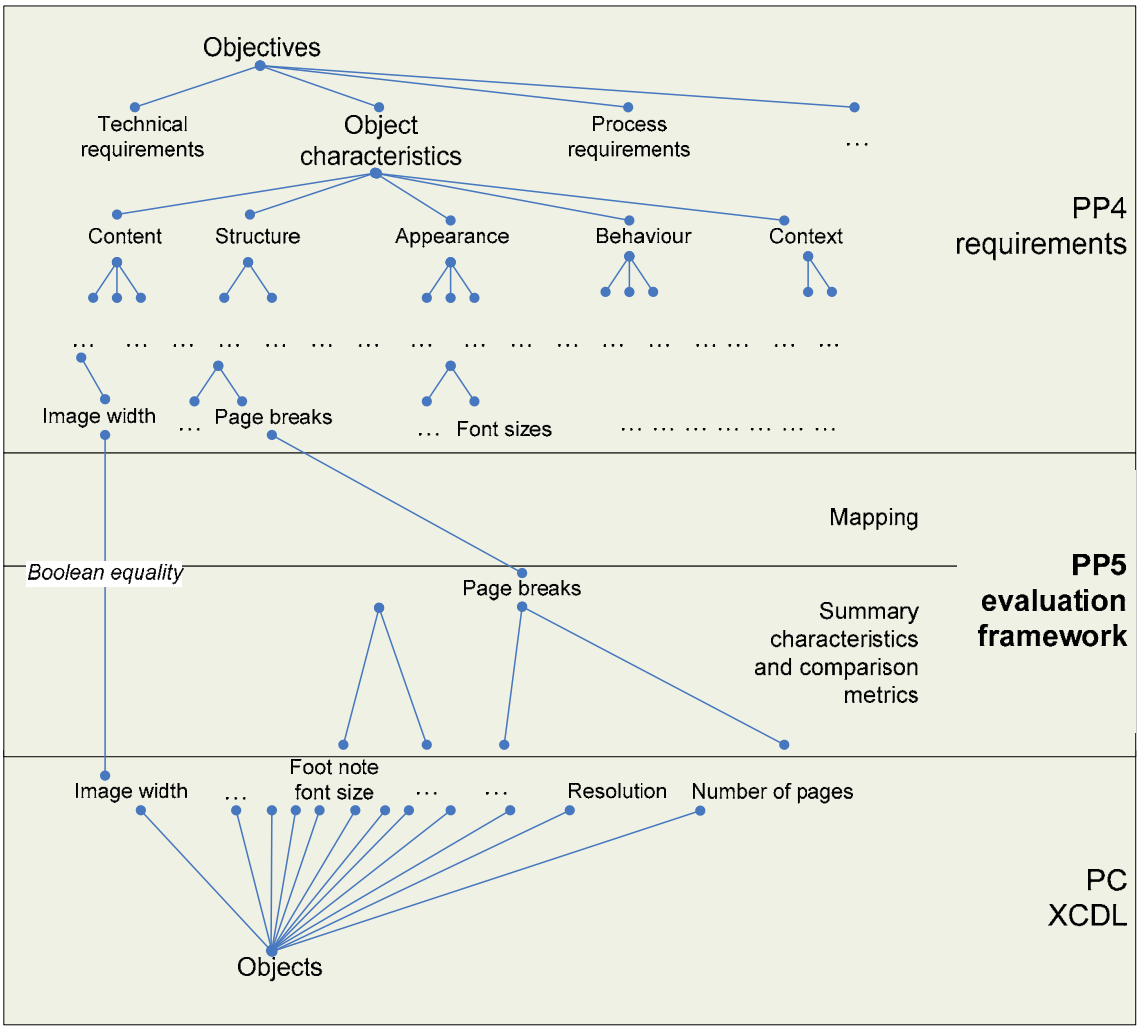


**Figure 1: From objects to objectives: Mapping requirements to technical characteristics of objects**

What is needed to connect these components is a framework that maps significant properties as they are defined in the objective trees to the technical characteristics described by XCL. This is done by the validation framework, i.e. its components, the evaluation framework and the comparison measures. Both have been deployed as software components. The evaluation framework is part of the Planets Preservation Planning Tool (Plato); the comparison measures are deployed within the XCL, namely within the Comparator Tool, which is also integrated into the Planets Interoperability Framework (IF) as one of the web services [9, 13].

Not shown in this figure, but also relevant, is the connection of other branches of the requirements tree to cover aspects, e.g., format risks and technical requirements, such as the performance of tools, which are relevant to the decision making and can be measured automatically.

## 1.2 Components of the Validation Framework

The Planets preservation planning approach as it is described in [1] and implemented in the planning tool Plato [6,11] defines requirements on preservation strategies in a hierarchical form, using a tree structure called an 'objective tree'. A significant part of these objectives is naturally concerned with preserving specific criteria of the objects in question (usually referred to as 'significant properties').

Up until now, evaluation of how a preservation action has preserved – or destroyed – these criteria and measurements of format risks and properties of the action components had to be performed manually and entered into the decision support software during the planning process.

The main goal of the evaluation framework described here is to automate this evaluation process and thus both reduce the level of effort required for evaluating potential preservation actions during the planning procedure and improve the reproducibility of evaluation decisions.

As Figure 1 shows, a connection is needed between technical characteristics and intellectual properties of objects. While the definition of comparison metrics provides one half of this link, a connection between the criteria defined in objective trees and these metrics must be created as well.

The validation framework thus provides a mapping mechanism from criteria to properties and their corresponding comparison metrics. This is done by a three-step process.

1. First characteristics that are measurable for a file format, and the metrics that can be used to compare them, are queried.
2. Second, these characteristics and the corresponding comparison metrics are mapped onto criteria defined in the objective tree.
3. Third, during evaluation of preservation actions, the comparison service is accessed, providing a list of characteristics to compare and the selection of comparison metrics. These are used to compute the similarity between the objects that shall be compared.

The sequence diagram shown in Figure 2 illustrates this process in its basic form, showing the main components of the validation framework. The preservation planner, who operates Plato through a graphical user interface, sends a query for characteristics and their corresponding metrics from the comparison service. The input parameter for this query is the input format identified by a Pronom Unique Identifier (PUID) designating a Pronom format held in the characterisation registry. As a response, Plato receives a list of characteristics that can be measured in this format, and corresponding metrics for comparing these characteristics.

These characteristics are then mapped onto criteria in the objective tree. Screenshots of the user interface that supports the mapping procedure in the planning tool are provided in Figures 6-9. The user can map criteria such as the colour depth of an image to a technical characteristic named 'bitDepth'. In a multi-step wizard, the planner is further able to select which comparison metric shall be used to compare the transformed object to the original. This can be a simple binary test for equality, but also a test for proportional deviation.[2]

The actual comparison of the original and the transformed objects takes place in the third step where the comparison service is called, providing the objects to be compared and the properties and metrics that shall be computed.

2 Cp. specification of measures (in [9], available at: http://planetarium.hki.uni-koeln.de/planets_cms/)

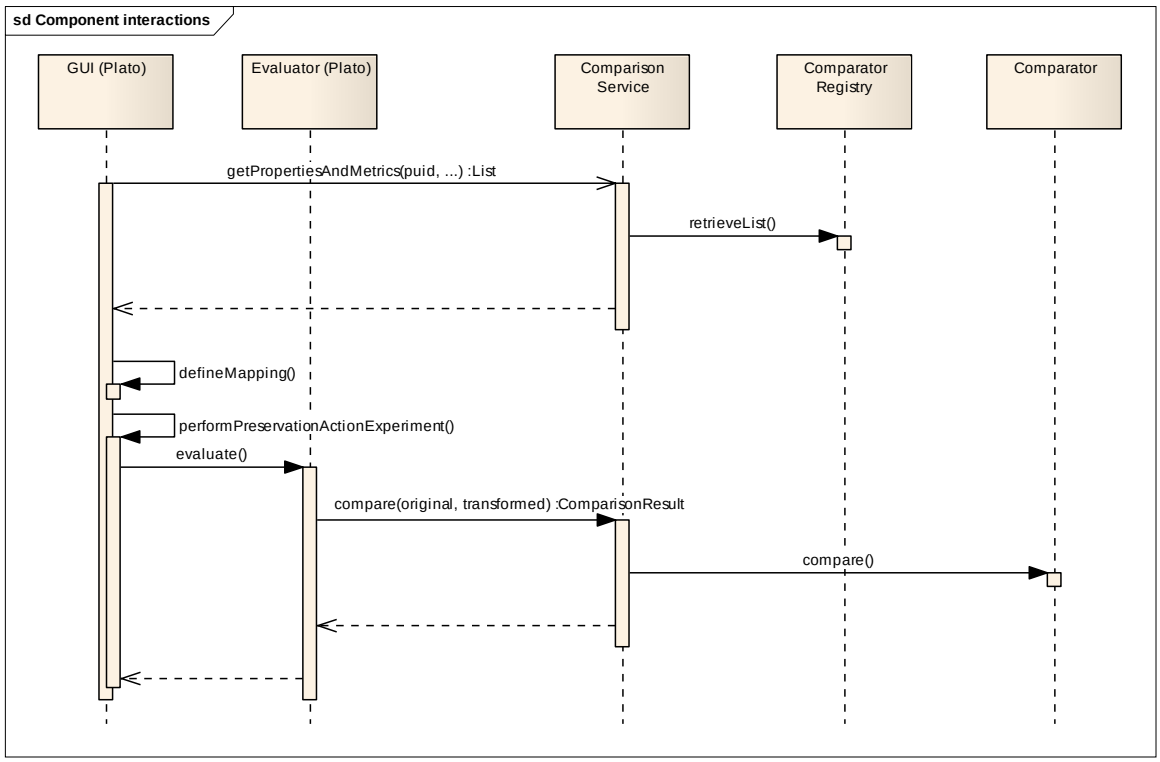


**Figure 2: Sequence of interaction between the components of the validation framework**

## 1.3 Comparison Measures

### 1.3.1 Motivation

As outlined in the previous chapter, the validation framework needs a connection between technical characteristics and intellectual properties. One half of this link is provided through the definition of adequate comparison measures.

Technical characteristics of digital objects, at least the most significant ones, are usually defined by file formats, i.e. described in the file format specification and codified in a specific file. The Extensible Characterisation Languages (XCL) [9] have been developed in order to enable the extraction and representation of file format characteristics (syn. 'properties'). The eXtensible Characterisation Definition Language (XCDL)[3], as one part of the XCL, allows for representation of such properties; the eXtensible Characterisation Extraction Language (XCEL)[4] is a generic language for describing how to extract characteristics from files. Both language parts have been designed with the goal in mind of making them usable in a 'real' practical way, i.e. to enable dedicated software tools to process them. The Extractor tool extracts the content (i.e. the properties) of files using an instantiation of the XCEL (a so-called XCEL description of a specific file format) and finally writes down these properties to a file using the XCDL syntax. The XCDL file which now contains a specific set of file format characteristics can then be compared by another software component, the Comparator tool. The main goal of this tool is the automated evaluation of file format conversions. One of the first steps towards such an automated comparison of file format based characteristics is to define a set of appropriate measures, general enough to meet the requirements for comparisons on file format level and particularly with regard to the specific representation of properties provided by the XCDL.

[3] Cp. [9], chapters 2, 9.1

[4] Cp. [9], chapters 3, 9.2

These requirements defined for the XCL also perfectly fit into the requirements for the validation framework. Therefore, the validation framework uses the XCL tools and its descriptions, especially the Comparator tool and the XCDL. The comparison measures are integrated into the Comparator and utilized within the validation framework. Since the Comparator works on XCDL descriptions of objects for comparison, the measures - at least in parts – are not totally detached from the XCL concepts and partly reflect the structure and logic of XCDL-built files.

As a consequence for classification, the comparison measures are grouped into two main classes. The first one, the group of *basic measures*, has already been defined in the initial specification of the validation framework [5]. This category is relatively independent from the XCL implications and comprises well-established measures which make use of fundamental mathematical concepts (e.g., a simple boolean comparison on strings, so called 'root mean squared error' from the image processing domain). These basic measures are most applicable for straightforward and elementary comparisons (e.g., one-to-one related properties), where an isolated comparison result is needed for a specific property and for a specific context. Within the evaluation framework, for example, basic measures can be applied on properties along with the mapping wizard where a criterion is connected to a measurable criterion derived from an XCDL description.[5]

While the set of basic measures is sufficient enough to meet the requirements of basic elementary comparisons within the evaluation framework, it is not sufficient for complex comparisons. Therefore, a second category of measures has been added, called *structural measures*. Metrics of this group are more closely related to the XCDL structure, i.e. to specific elements, attributes and logical concepts of the XCDL, in order to enable complex comparisons (see next section).

All measures are documented in detail within the XCL release [9]. For classification purposes, each measure is assigned to a logical naming. The measure is verbally described, accompanied by other essential information (allowed input and output XCL data types, XCL-specific ID(s), equation) and an example is provided if considered necessary for understanding. Figure 3 shows an excerpt from the measure specification part of the XCL documentation [9].

**Measure name: levenshteinDistance**

Id: 15, 41, 44
Explanation: The Levenshtein distance is a distance measure for strings. Two strings are compared with respect to the three basic operations insert, delete, and replace, in order to transform string A into string B. The value for this metric is the number of operations needed for transformation.
XCL data type of input value: string
XCL data type of output value: int

Example:

Consider these two strings:
byte
bit

The operations needed to transform 'byte' to 'bit' are e.g.
1. byte -> bite (Replace 'y' with 'i')
2. bite -> bit (Delete 'e')
In this example, the Levenshtein distance is: $d_{lev} = 2$.

**Figure 3: Excerpt from metrics specification for measure 'levenshteinDistance'.**

For details on how to obtain the XCL documentation and software, including the Comparator Tool and the measure specification, see chapter 2.2 below.

[5] Cp. section 2.1

### 1.3.2 Enhancements

1. Definition of Structural Measures

In the initial report [5] we defined a basic set of comparison metrics which enable operation on a certain number of categories of comparison. These are comparisons on the level of

1) single properties (one-to-one)
2) sets of properties (one-to-many, many-to-many)
3) references of property values to XCL normalised data
4) complex objects (nested objects, related objects)
5) XCDL data categories (labelled values, normalised data)
6) XCDL data types
7) 'nature' of property occurence (property with single value, set of values)
8) multiple relations between categories 1 – 7 (which make up the entire digital object)

The basic set of metrics is sufficient enough to satisfy the fundamental requirements of the validation framework as defined in [5]; it meets the categories 1, 5, 6 and partly 7 (single values supported).

However, to meet the requirements for comparison on the rest of the identified levels, the existing set of basic metrics had to be extended. New measures, operating on additional structural parts of the XCDL descriptions, have now been added to the existing set, arriving at the definition of measures which are called *structural measures*.

We classified the new measures as 'structural' since they operate on the XCDL structure, i.e. on certain elements and attributes of the XCDL syntax in combination with the file-extracted value(s) for a specific property. To make this more evident, consider the following example.

Let us assume a text contains three different fonts (e.g., Arial, Times, Courier style). The XCL Ontology[6] defines 'font' and its associated information as a set of specific *properties* of text. A PDF file as well as a DOC file contains this font information in its encoded bytestream. With the help of the XCL tools it is possible to extract this information and to represent it using the XCDL syntax.

```
<property id="p19" source="raw" cat="descr">
   <name id="id159">fontName</name>
   <valueSet id="i_i1_s185">
      <labValue>
         <val>Arial</val>
         <type>string</type>
      </labValue>
      <dataRef ind="normSpecific" propertySetId="o1_id_4"/>
   </valueSet>
   <valueSet id="i_i1_s263">
      <labValue>
         <val>Courier</val>
         <type>string</type>
      </labValue>
      <dataRef ind="normSpecific" propertySetId="o1_id_5"/>
   </valueSet>
   <valueSet id="i_i1_s324">
      <labValue>
         <val>Times</val>
         <type>string</type>
      </labValue>
      <dataRef ind="normSpecific" propertySetId="o1_id_0"/>
   </valueSet>
</property>
```

**Figure 4 : XCDL representation of property 'fontName'.**

[6] Simply spoken, the XCL Ontology is a controlled vocabulary where relations between object categories are modelled. According to the Ontology, 'font' (property) is related to text (object) as 'font is a property of text'. See [9], chapter 4 ('XCL Ontology') for more.

Figure 4 shows an exerpt from an XCDL description of such font information (in this case the naming of fonts) extracted from a file.

One task within an evaluation action could be to find out if this specific font information ('fontName') has changed or not, assuming a conversion scenario where, e.g., the text which is originally stored as a Microsoft Word file (in short: DOC file) is converted to a PDF file. For this purpose, each binary file, the (source) DOC file and the resulting (target) PDF file, is extracted to an XCDL representation.

Assuming a 'normal' conversion where the result of the conversion process turned out to be as expected, i.e. three different font names are associated with the three fonts used in the text. The resulting XCDL descriptions for both files then contain a property 'fontName' not only with one but with three values: 'Arial', 'Times' and 'Courier'. These three values are expressed as three repeating structures within the XCDL description (a tagset <valueSet> is inserted for each value).

In order to find out now whether the font names and their number have changed we first need to know if the number of value sets (and thus the number of <valueSet> elements) is the same in both XCDLs. Secondly, the actual values have to be compared.

This example is a typical case where the application of metrics from the set of basic metrics is not possible, since factors which are expressed in the XCDL syntax have to be taken into account. Therefore, it is necessary to define measures which are able to cope with circumstances such as multiple property values. Exactly this is done with introducing the structural measures. They combine comparisons of such structural factors with the comparison of 'real' values (the property values extracted from the binary files), i.e. they actually summarize the results of application of different structural and basic measures.

## 2. Extension of Measures for Handling Comparisons of Content from Different Information Domains

The extension of the basic measures towards the definition of structural measures is very closely related to a second main enhancement of the comparison measures. In the initial version of the validation framework, it was only possible to compare content of objects containing image-related information. This has now been extended to objects containing text-related information. In terms of file formats this means that the measures of this version are now able to handle content of image file formats, text-based formats and, by release of the final version of the XCL tools[7], audio file formats.

Structural measures are often related to content from the text information domain. This is a result of the nature of this kind of information: a text based format often contains text with more sophisticated formatting (as demonstrated in the example before); image information by contrast is in almost all cases 'simple' in the way that image properties have a single value (an image has exactly one width, height, bitDepth, etc.). Additionally, there are almost no cross-references of properties as is typical for text-based information. For example,, if different fonts are used, they refer to different parts of the text: There may be a paragraph starting with characters in font-style 'Times New Roman' followed by a text section where font-style 'Courier New' is used. These issues are also handled by the structural measures.[8]

The fundamental difference in the nature of the information from different domains also makes it necessary to introduce additional basic measures: the information contained in audio data is totally different from image data. Comparing audio data with a measure from the image data domain, e.g. the root mean squared error would not make much sense since audio data is related to frequencies and not to pixel data. In a similar way, it is also necessary to define appropriate measures for text data. This has also been introduced in the updated version of the comparison measures.

[7] At the end of May 2010.

[8] Cp. [9], chapter 9.6.1.2.2, where measures which relate to such references are defined.

# 2. Software: Tools and Components

## 2.1 Plato

The planning tool Plato is a decision support tool that implements the Planets Preservation Planning method [2] and integrates services for content characterisation, preservation action and automatic object comparison in a service-oriented architecture to provide maximum support for preservation planning endeavours [6].

Plato is freely available as a web application at:

www.ifs.tuwien.ac.at/dp/plato

Extensive documentation is available at the website.

The core part of the planning approach is the definition and evaluation of requirements in the objective tree. To this end, a tree editor is provided in the software, with two-way integration of mind-mapping software as an additional interface option. On the criteria level, measurements can be specified by creating a connection as outlined in Figure 5.

The next figures show the mapping wizard that connects a criterion to a measurable criterion in XCDL. In the tree editor, the column to the right indicates the status of mapping. Activating it starts the mapping wizard.

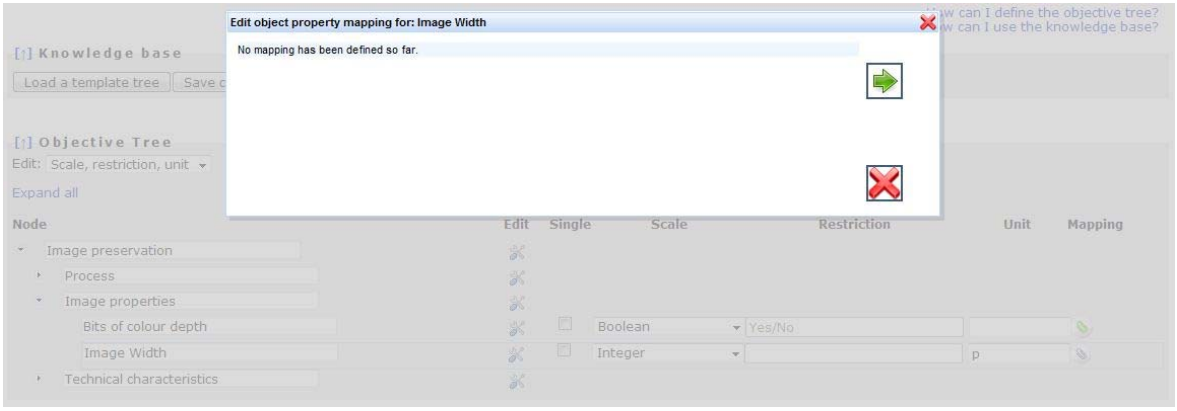


**Figure 5: Starting the mapping wizard in Plato**


**Figure 6: Selecting a property to be mapped**

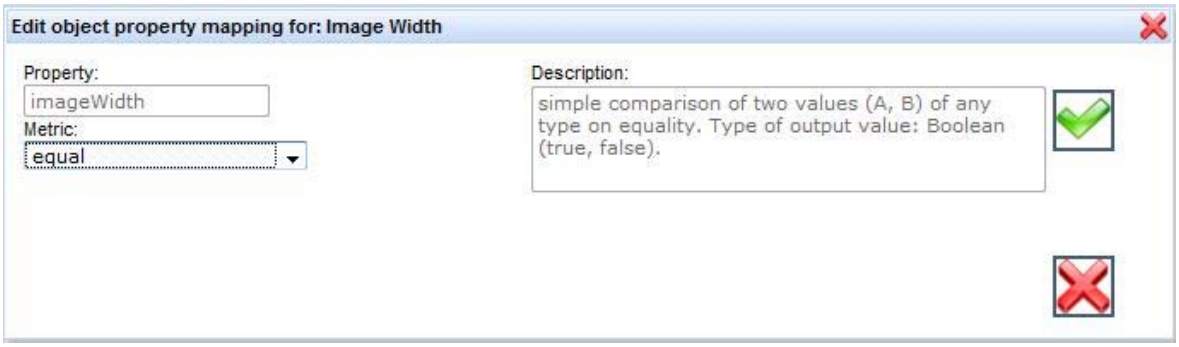


**Figure 7: The basic metric 'equal' that measures equality**

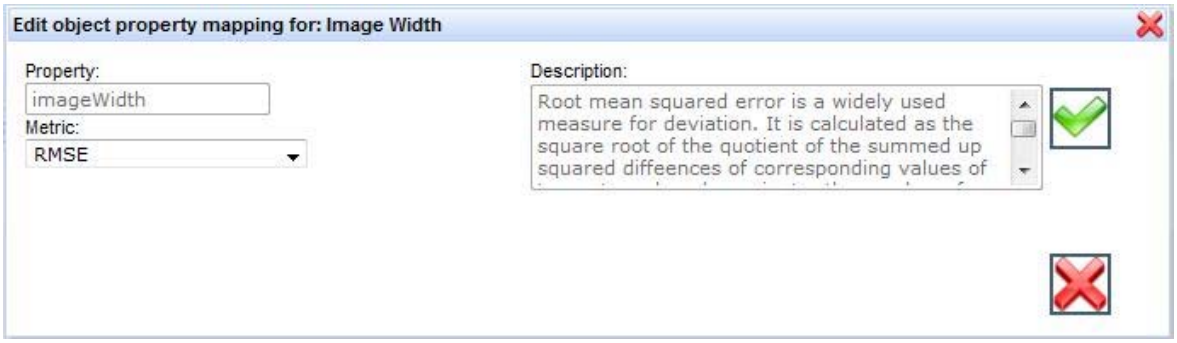


**Figure 8: Root mean squared error (RMSE)**

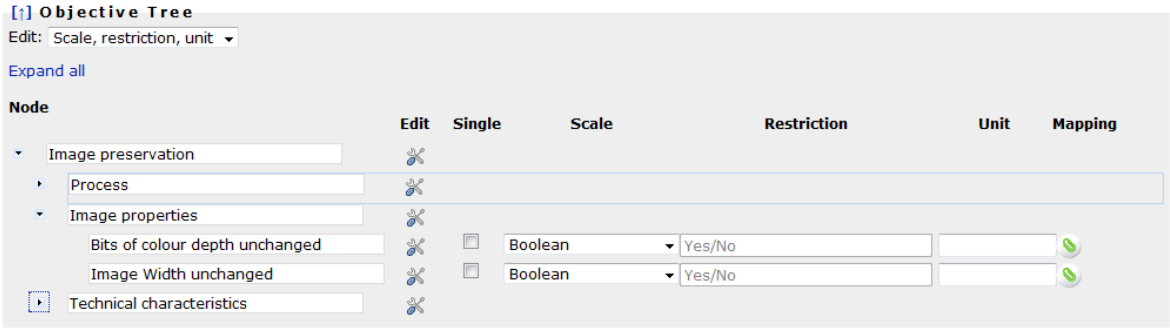


**Figure 9: Mapped properties are visually indicated in the column *Mapping***

## 2.2 Comparator

Comparator is one of the two software tools of the XCL Suite [9]. It is able to compare properties of digital objects as described through their representation as files.

From a technical point of view, it processes files that contain the properties of digital objects in a specific language expressed in XML, the eXtensible Characterisation Definition Language (XCDL). XCDL files are currently created by the Extractor, the XCL tool that is able to extract properties from files with the help of the eXtensible Characterisation Extraction Language (XCEL).[9]

[9] XCDL descriptions do not have to be created necessarily by the Extractor software. Another approach for creating XCDL descriptions by means of XSLT is currently under work (transforming JHove output to XCDL). This will be finished by end of April 2010, and published as internal Planets report PC4-D14 and additionally integrated into the second edition of [9], end of May 2010.

For its comparison algorithms the Comparator uses the comparison measures defined within the validation framework.

Comparator targets three application scenarios. These are

1. Automated evaluation of large-scale file format migrations (conversion), in association with XCL tools and functionalities.
2. Support of preservation planning tasks within the validation framework.
3. Support of experiments conducted via the Planets testbed.

For application scenario 1, a stand-alone command-line tool is available. For the other two target applications, Comparator has been wrapped as a web service by the Planets Interoperability Framework (IF). The interfaces defined in [5] ('Metrics toolbox and interfaces') have been fully integrated into the IF. For a detailed specification and description see [9], chapter 6 ('XCL in Planets') and [13].

All major (alpha) releases of the Comparator can be obtained from:
http://planetarium.hki.uni-koeln.de/planets_cms/comparator-command-line-tool

Additionally, minor versions in beta status can currently be downloaded from:
http://gforge.planets-project.eu/gf/

Comparator and the XCL tools and components are open source. [10]

Both sources include detailed documentation of the Comparator tool and the associated comparison measures.

## 2.3 Quality-aware Migration

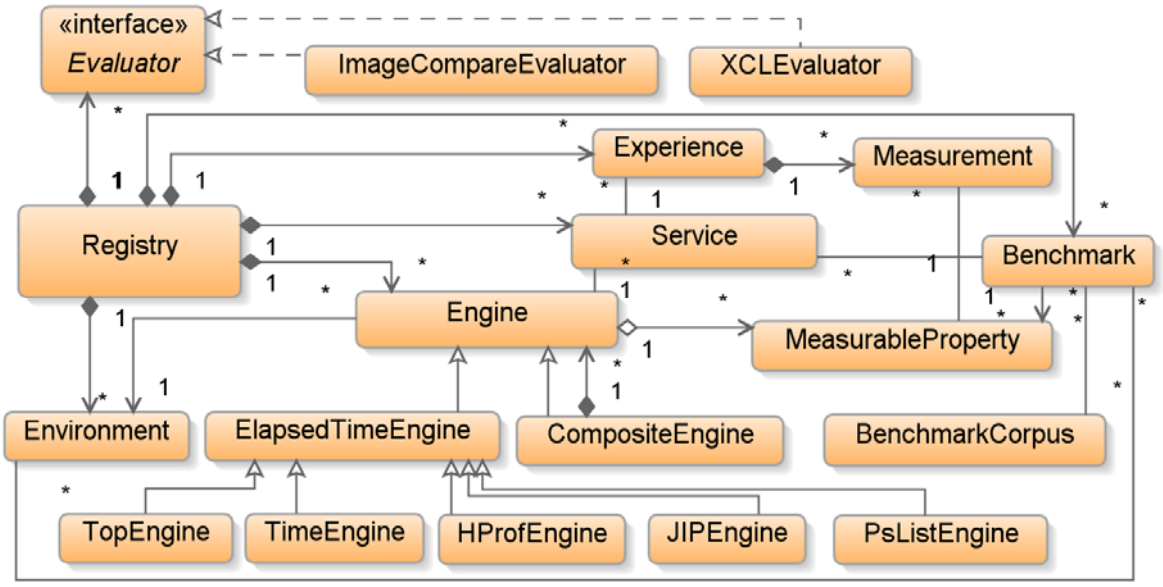


**Figure 10: Core elements of the monitoring framework**

To cover other aspects of interest, such as performance, we have developed a framework for quality-aware migration. The minimal migration engine MiniMEE is integrated in the planning tool

[10] It is intended to move Planets gForge to the SourceForge platform (http://sourceforge.net/) at the end of the Planets Project, May 2010.

and can be accessed through the planning workflow. It has been presented and described in detail in [8].

Figure 10 shows a simplified abstraction of the core elements of the monitoring design and their relations.

The key elements are *Services*, *Engines*, and *Evaluators*, which are all contained in a *Registry*. Each *Engine* specifies which aspects of a service it is able to measure in its *MeasurableProperties*. The property definition includes the scale and applicable metrics for a property, which are used for creating the corresponding *Measurements*. Each *Engine* is deployed within a specific hardware *Environment* that shows a certain performance. This performance is captured by the score of a *Benchmark,* which is a specific configuration of services, and *Data*, aggregating measurements over these data to produce a representative score for an environment. The benchmark scores of the engines' environments are provided to the clients as part of the service execution metadata and can be used to normalise performance data of migration tools running on different hardware platforms.

The *Services* contained in a registry are not invoked directly, but run inside a monitoring engine to enable performance measurements. This monitoring accumulates *Experience* for each service, which is collected in each successive call to a service and used to aggregate information over time. It thus enables continuous monitoring of performance and migration quality. *CompositeEngines* are a flexible form of aggregating measurements obtained in different monitoring environments. This type of engine dispatches the service execution dynamically to several engines to collect information. This is especially useful in cases where measuring code in real-time actually changes the behaviour of that code. For example, measuring the memory load of Java code in a profiler usually results in a much slower performance, so that simultaneous measurement of memory load and execution speed leads to skewed results. Fortunately, in this case there is a way around this uncertainty relation – forking and distributing the execution leads to correct results.

The bottom section of Figure 10 illustrates some of the currently deployed performance monitoring engines.

1. The *ElapsedTimeEngine* is a simple default implementation measuring elapsed (wall-clock) time.
2. The *TopEngine* is based on the Unix tool top[11] and used for measuring the memory load of wrapped applications installed on the server.
3. On Windows servers, the *PsListEngine* relies on PsList[12] to obtain the same information.
4. The *TimeEngine* uses the Unix call time[13] to measure the CPU time used by a process.
5. Monitoring the performance of Java tools is accomplished by a combination of the *HProfEngine* and *JIPEngine*, which use the HPROF[14] and JIP[15] profiling libraries for measuring memory usage and timing characteristics, respectively.

Additional engines and composite engine configurations can be added dynamically at any time. Notice that while the employed engines focus on performance measurement, in principle any category of dynamic QoS criteria can be monitored and benchmarked.

In contrast to performance-oriented monitoring through engines, *Evaluators* are used for comparing input and output of migration tools to compute similarity measures and judge migration quality. The *ImageCompareEvaluator* relies on the compare utility of ImageMagick to compute distance metrics for pairs of images. The more generic *XCLEvaluator* uses the eXtensible Characterisation Languages and compares different XCL documents for degrees of equality as described above. The measurements obtained through evaluators are deposited in the metadata of the migration result and are analysed in the analysis phase of the planning workflow.

11 http://unixhelp.ed.ac.uk/CGI/man-cgi?top

12 http://technet.microsoft.com/en-us/sysinternals/bb896682.aspx

13 http://unixhelp.ed.ac.uk/CGI/man-cgi?time

14 http://java.sun.com/developer/technicalArticles/Programming/HPROF.html

15 http://jiprof.sourceforge.net/

## 2.4 Risk Assessment

Risk assessment of file formats is another key aspect of evaluation. Criteria about file formats, which have since long been in the focus of analysis in the digital preservation community, are suitable to be described in publicly accessible registries maintained by institutions with long-term commitment and substantial resources for evaluating certain aspects of formats.

Tarrant [3] presented the P2 registry[16] which uses Semantic Web technologies to combine the content of PRONOM, represented as RDF, with additional sources such as DBpedia. The P2 fact base currently contains about 44.000 RDF statements about file formats and preservation tools.

Figure 11 shows some of the facts known about PDF 1.4. P2 contains a risk calculation model, where the organisation's preferences are captured in a *risk profile* that models the sensitivity of the organisation to certain risk factors. The outcome is a numerical risk score alongside a summarising analysis of the factors [3]. In our approach in Planets, the *utility function* fulfills this role by transforming the measurement of each criterion according to the acceptance thresholds of an organisation and thus together with the weighting of factors models the risk aversion curve of the institution. This in turn is influenced by the policy settings defined by the organisation.

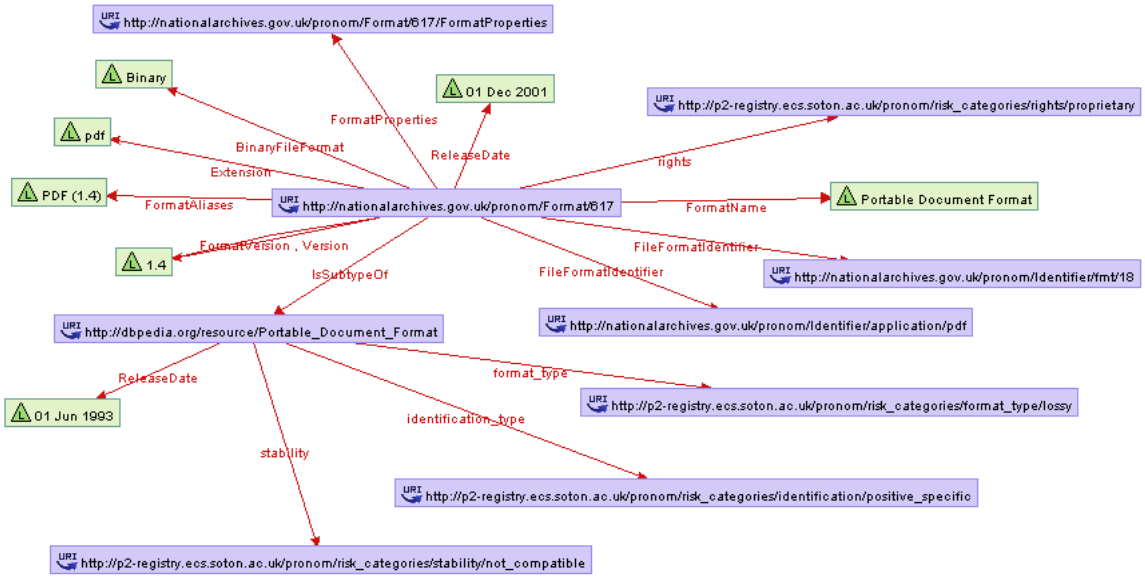


**Figure 11: RDF graph showing some of the facts known about PDF 1.4**

The P2 knowledge base has been integrated into the planning tool Plato and allows the automated evaluation of certain risk factors. For example, to extract the *disclosure* of a file format, the following SPARQL code fragment is used.

```
prefix pronom: <http://pronom.nationalarchives.gov.uk/#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?d WHERE {
?format pronom:FormatDisclosure ?d .
?format pronom:FileFormatIdentifier ?ident .
?ident pronom:IdentifierType "PUID" .
?ident pronom:Identifier $PUID$
}
```

This further contributes to the coverage of automation in the preservation planning procedures and to the completeness of the validation framework.

Further information about these aspects as well as a full release report of the planning tool will be included in the final deliverable of PP4, the release 3.0 of the planning tool Plato.

[16] http://p2-registry.ecs.soton.ac.uk/