Project Number	IST-2006-033789
Project Title	Planets
Title of Deliverable	**Specification of basic metric and evaluation framework**
Deliverable Number	**PP5/D1**
Contributing Sub-project and Work-package	SP/PP/5
Deliverable Dissemination Level	External Planets All
Deliverable Nature	Report
Contractual Delivery Date	31st May 2007
Actual Delivery Date	
Author(s)	Volker Heydegger Christoph Becker

Keyword list

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

Contributors

Person	Role	Partner	Contribution
Christoph Becker	Contributing author	TUWIEN	Main author of chapters 1, 2, 3.1 and 3.3. Contribution for chapter 5.
Volker Heydegger	Contributing author	UzK	Main author of chapters 3.2 and 4. Contribution for chapters 1 and 5.
Manfred Thaller	Lead	UzK	Common coaction

Document Approval

Person	Role	Partner
Andrew Lindley	Reviewer	ARC
Petra Helwig	Reviewer	NANETH
Manfred Thaller	WPL	UzK
Hans Hofman	SPL	NANETH

Distribution

Person	Role	Partner
WP participants		
Hans Hofman	SPL	NANETH
PP subproject mailing list		
Deliverables mailing list		

Revision History

Issue	Author	Date	Description
0.1	Christoph Becker	May 20	Initial draft
0.2	Christoph Becker	May 24	Revised and extended draft
0.3	Volker Heydegger	May 27	Revised and extended draft
0.4	Volker Heydegger	May 29	Final draft for review
1.0	Volker Heydegger	June 26	Final version

References

Ref.	Document	Date	Details and Version
1	Planets PP4/D1 Report on methodology for specifying preservation plans	31st July 2007	available at http://www.planets-project.eu/publications
2	Stephan Strodl, Christoph Becker, Robert Neumayer, Andreas Rauber: How to Choose a Digital Preservation Strategy: Evaluating a Preservation Planning Procedure	June 2007	Proceedings of the ACM IEEE Joint Conference on Digital Libraries (JCDL'07), Vancouver, British Columbia, Canada, June 18-23, 2007. pp 29-38
5	Christoph Becker, Andreas Rauber, Volker Heydegger, Jan Schnasse, Manfred Thaller.	March 2008	In: Proceedings of the ACM Symposium on Applied Computing (SAC'08), Track 'Document Engineering'. Fortaleza, Brazil, March 16-20, 2008.
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. Planets deliverable PC3-D7		
9	Heydegger, V., Neumann, J., Schnasse, J., Thaller, M. Basic design for the extensible characterisation language. PC/2-D1, PC/2-D2 Planets internal	October 2006	Internal report for Planets: http://www.planets-project.eu/private/pages/wiki/index.php/PC/2_Characterisation_Strategy_Development

	deliverable.		
11	PP4-D3 Report on service integration in Plato2. Planets external deliverable	May 2008	
13	PC2-D7: Final Specification of the extensible Characterisation Definition Language (XCDL)	May 2008	available at http://www.planets-project.eu/publications
14	PC2-D8: : Final Specification of the extensible Characterisation Extraction Language (XCEL)	May 2008	available at http://www.planets-project.eu/publications
15	PRONOM Unique Identifiers. The National Archives.		webside available via http://www.nationalarchives.gov.uk/aboutapps/pronom/puid.htm

EXECUTIVE SUMMARY

This document specifies the basic metric and evaluation framework developed in Planets PP5.

The practical outputs of other Planets work-packages, mainly those of PP4, PC2 and PC4 are connected within a common framework: Significant properties as they are defined in the objective trees in PP4 are mapped to the technical characteristics described by XCL in PC2/4. The *validation framework* consists of two main building blocks:

1. Comparison metrics as basis for computing simple measurements or aggregated measurements over several low-level characteristics ('summary metrics'), and
2. A mapping mechanism for connecting criteria to both measurements and partly also directly to technical characteristics ('evaluation framework').

From the bottom-up, objects are characterised, and based on the characteristics described by 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 from one side closed by the comparison metrics. From the other side the bridge is formed by a mapping between the requirements and these comparison metrics, which connects the objective tree and the tree of extracted characteristics and thus supports the automatic evaluation and validation of preservation actions.

TABLE OF CONTENTS

1. Introduction 5
2. Preservation Planning and Significant Properties....7
3. Evaluation Framework 10
3.1 Introduction....10
3.2 Interface definitions 11
3.2.2 Metrics Toolbox....12
3.2.2.1 FPM Box 13
3.2.2.1.1 FPM File....13
3.2.2.1.2 FPM Unit 15
3.2.2.1.3 FPM Request....15
3.2.2.2 Comparison Box....16
3.2.2.2.1 Plato-to-Comparator Request (PCR)....17
3.2.2.2.2 Comparator-to-Plato Response (CPR)....18
3.3 Summary 19
4. Basic Metrics Set for Comparing Properties....20
4.1 Categories of Comparison....20
4.2 Metrics Specification....20
4.2.1 Group A: Arithmetical Metrics 20
4.2.2 Group B: Similarity / Distance Metrics....24
4.2.3 Group C: Statistical Metrics....29
5. Summary and Outlook 30

1. Introduction

The digital preservation landscape is evolving at an accelerating pace. Various migration tools are available for standard file formats such as office documents; the picture is less positive for more exotic and complex compound objects. However, even within migration tools for common object types, variation regarding the quality of conversion is very high. Some tools fail to preserve the proper layout of tables contained in a document; others miss footnotes or hyperlinks. Finding out which information has been lost during a conversion, and whether this loss threatens the value of the object for a given purpose, is a very time-consuming task. Some losses might be acceptable, while others threaten the authenticity of documents. For example, if migrating a collection of Word documents results in a loss of page breaks, this might be irrelevant if the textual content is the only thing of interest. However, if there are page references in the text, this loss might be unacceptable.

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 mean that this decision is a complex task. 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 for a long time be impossible: 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 [9,13,14] 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 straight forward way. This differentiates the XCL approach from the approach used by JHove and similar projects. The XCL does not attempt to extract a set of characteristics from a file, but it proposes to express the complete informational content of a file in a format independent model. Thus it supports the comparison of objects from different sources as illustrated by Figure 1.

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 these comparison metrics, which 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.

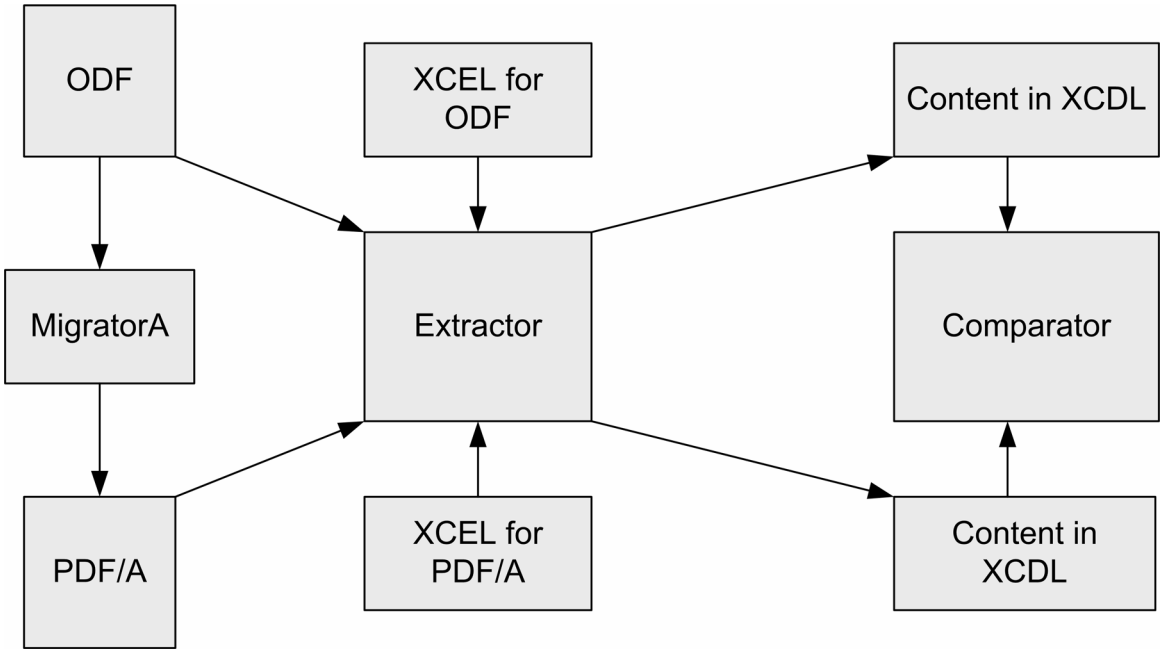

Figure 1: Using XCL to compare migrated documents

Related to objectives, requirements and characteristics, we distinguish the following terms:

Objectives describe high-level goals of a preservation endeavor. These are related to aspects such as the process, significant properties of objects, etc.

These objectives are broken down in the objective tree to **requirements**. Thus a requirement is an objective made more explicit and concrete.

Finally, a **criterion** is a requirement that can be quantified, thus it refers to the lowest level of the objective tree, where a measurement scale is defined. To make requirements quantifiable and measurable is one of the main goals of the work done in the Preservation Planning subproject.

On the other hand, we distinguish between technical characteristics and significant properties in the following way:

Significant properties are defined by the user in the objective tree. They start at an intellectual level such as “object appearance” and are broken down to the criteria level.

Technical characteristics on the other hand can be extracted from an object by an algorithm and thus a software tool. In the context of this document they will usually come from an XCDL description.

This document is structured as follows.

Section 2 describes the hierarchical structuring of requirements, particularly of object characteristics, in preservation planning, and demonstrates how on a conceptual level a mapping can be achieved to connect requirements to technical characteristics and comparison metrics.

Section 3 defines the implementation of the above mentioned mapping as it is being carried out in PP5.

Section 4 describes a set of basic comparison metrics.

Section 5 takes conclusions and defines the next work steps.

2. Preservation Planning and Significant Properties

The Planets preservation planning approach as it is described in [1,2] and implemented in the planning tool Plato [6,7,11,12] defines requirements on preservation strategies in a hierarchical form, using a tree structure called 'objective tree'. A significant part of these objectives naturally is concerned with preserving the significant properties of the objects in question.

These properties are usually defined starting with the five aspects

- Content,
- Appearance,
- Structure,
- Behaviour, and
- Context.

Figure 2 provides an example of overall preservation requirements for web pages, as they were defined in an earlier case study described in [1]. Figure 3 describes the object characteristics for the same scenario in detail, providing the measurement scales that were assigned to the criteria.

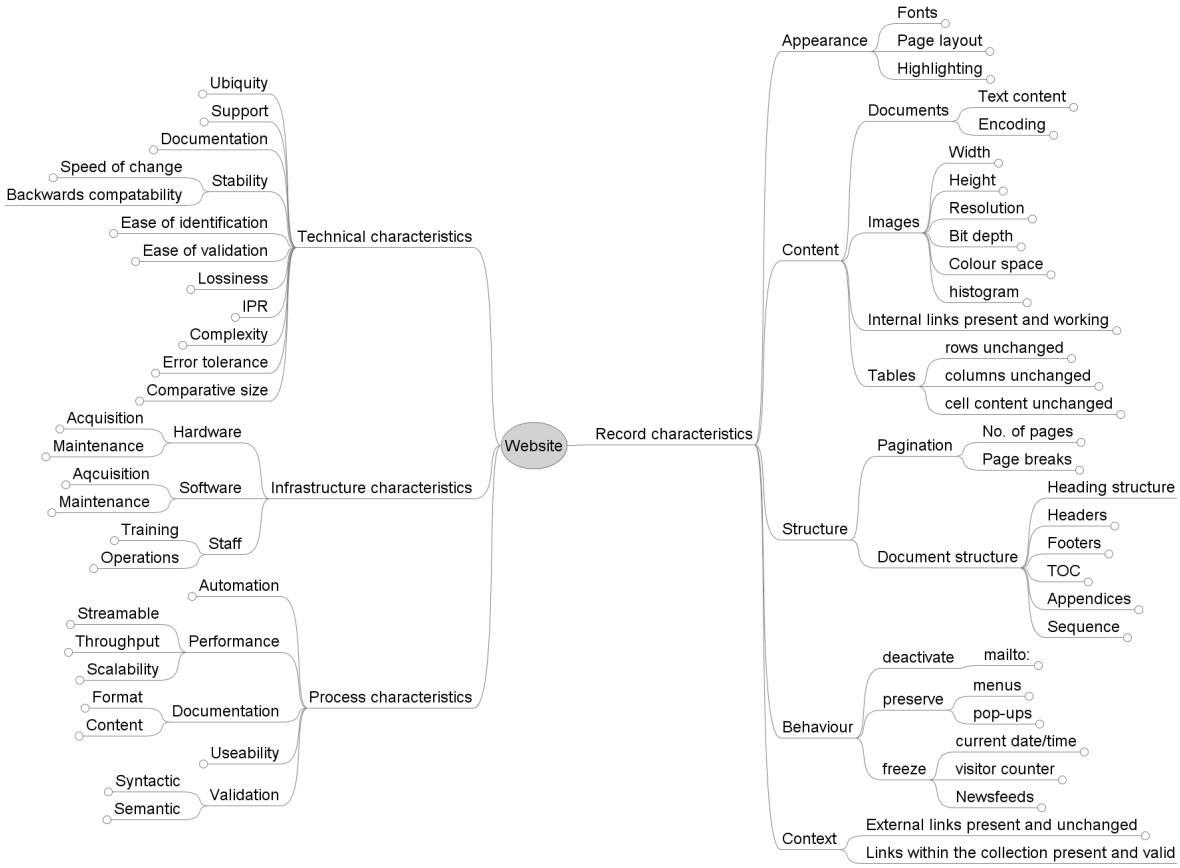

Figure 2: Requirements for preserving web pages (TNA)

Until now, evaluation of how a preservation action has preserved – or destroyed – these criteria has to be done manually, and entered into the decision support software during the planning process.

One of the main goals 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.

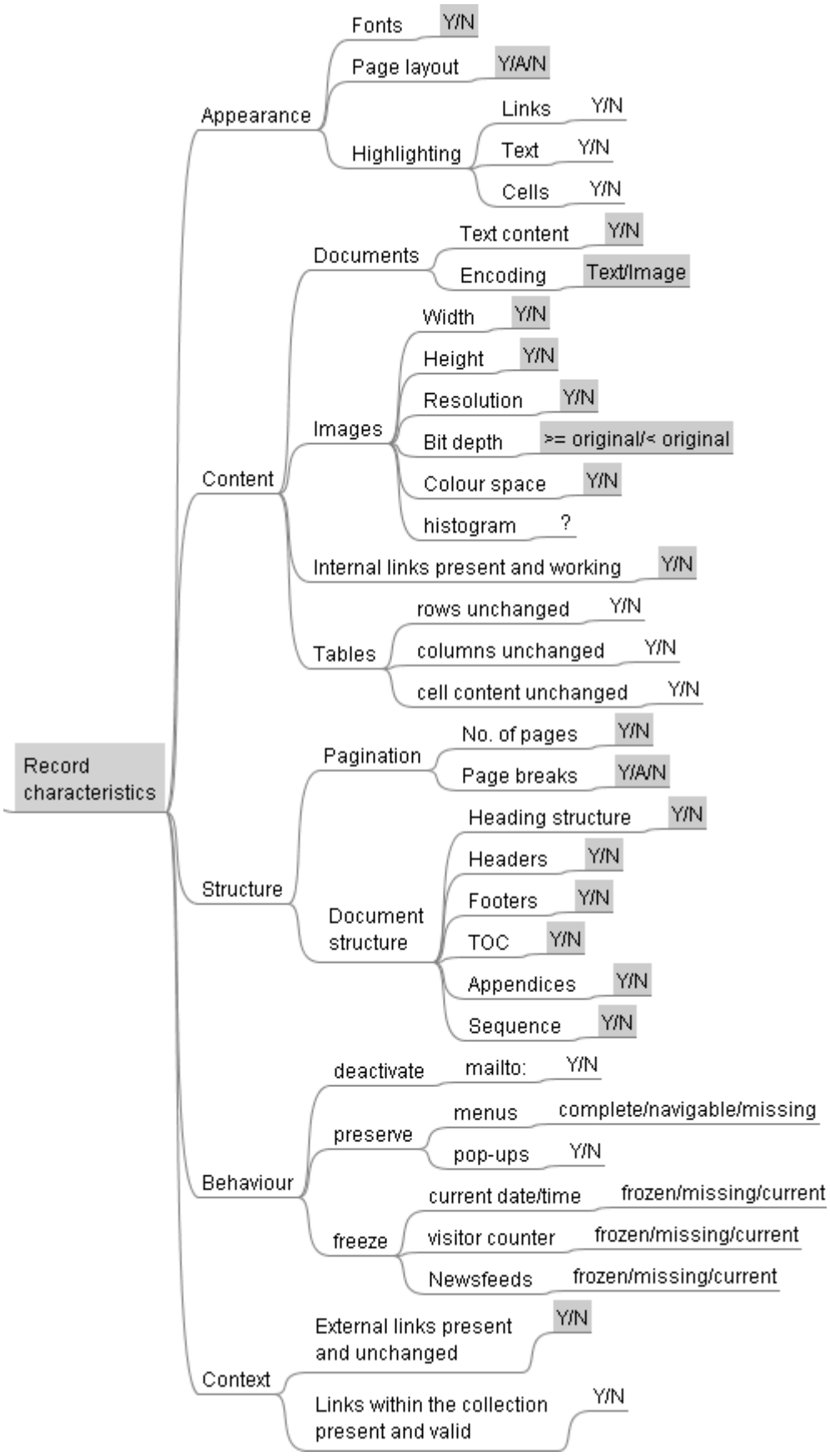

Figure 3: Significant properties of static web pages (TNA)

Figure 4 schematically illustrates the conceptual layers of the validation framework and their relationships, putting it in context with other work done in Planets. On the top, PP4 is defining 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 may be found such as

- *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*.

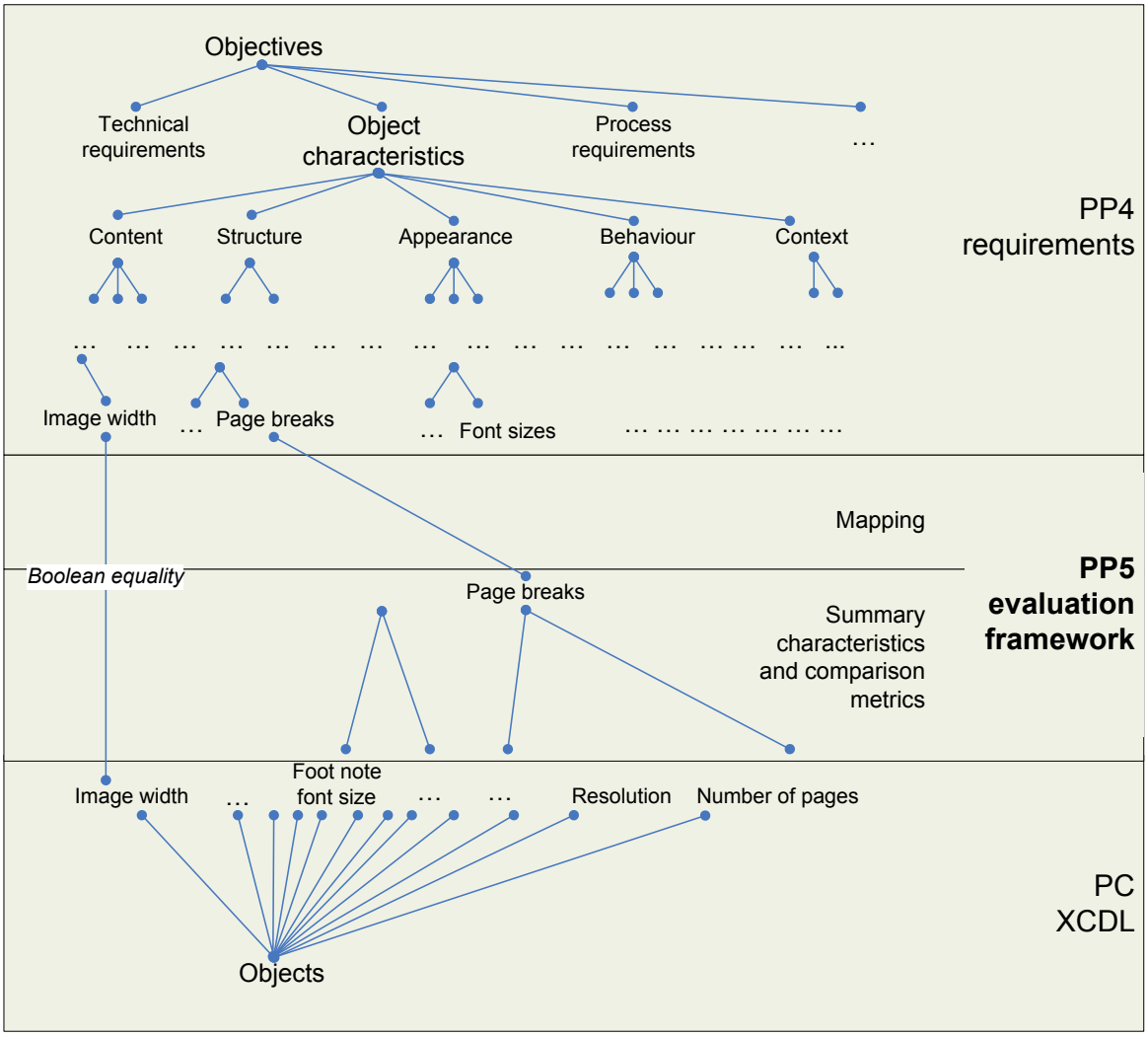

Figure 4: From objects to objectives: Mapping requirements to technical characteristics

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, for example, an algorithm is developed that is able to 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 come 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.

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 validation framework consists of two main building blocks:

1. Comparison metrics as basis for computing simple measurements or aggregated measurements over several low-level[1] characteristics ('summary metrics'), and
2. A mapping mechanism for connecting criteria to both measurements and partly also directly to technical characteristics ('evaluation framework').

Section 3 describes the mapping mechanism as a whole and how these layers and components are integrated on a technical level ('Metrics Toolbox').

Section 4 describes the basic set of comparison metrics in detail.

3. Evaluation Framework

3.1 Introduction

As Figure 4 did show, 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 first part of 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.

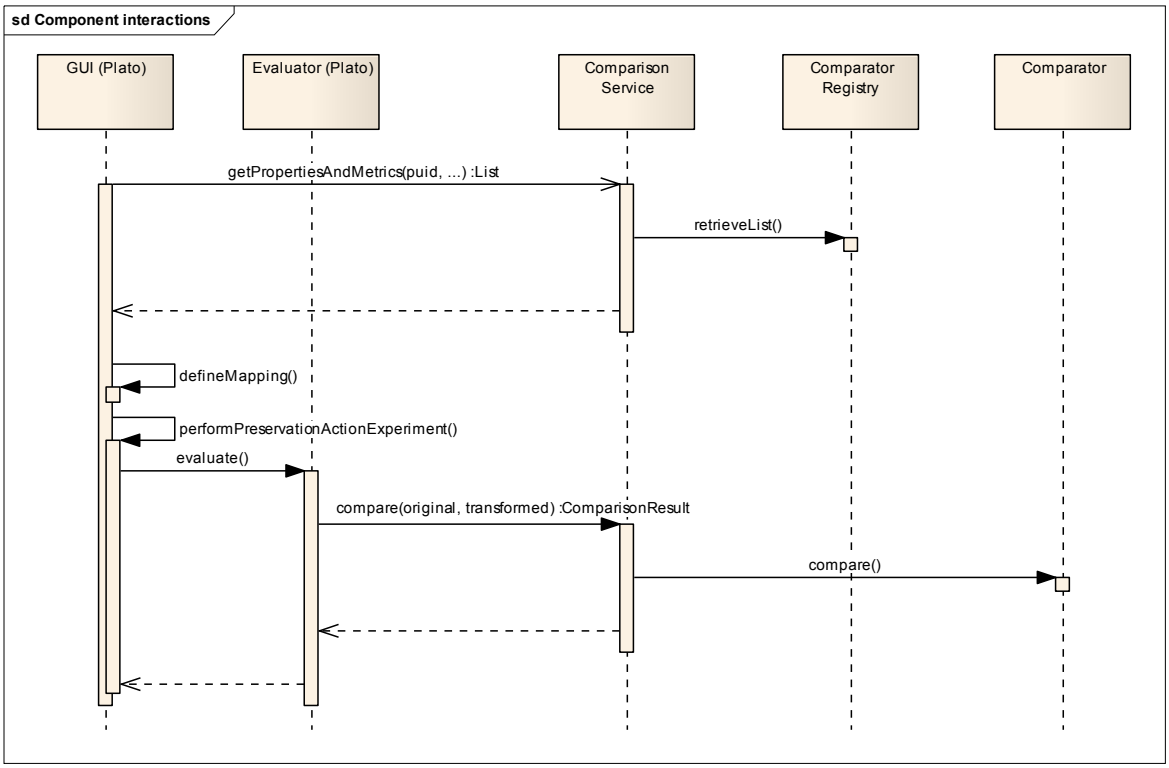

Figure 5: Sequence of interaction between the components of the validation framework

[1] i.e. technical properties

The simple sequence diagram shown in Figure 5 illustrates this process in its basic form. The preservation planner, who is operating 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. A screenshot of an early prototypical implementation that supports the manual mapping procedure in the planning tool is provided in Figure 6. The user can map criteria such as the colour depth of an image to a technical characteristic named 'bitDepth'. In the future, the planner will be able to select which comparison metric shall be used to compare the transformed object to the original. This might be a simple binary test for equality, but also a test for proportional deviation.[2]

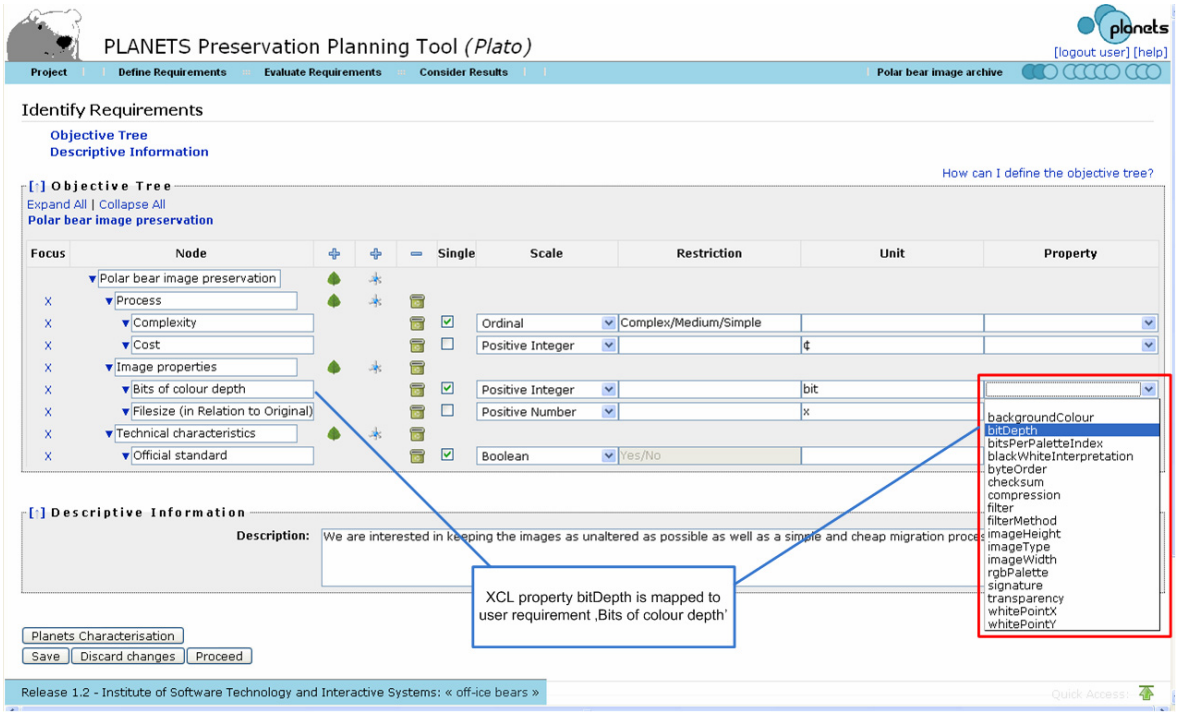

Figure 6: Prototype of Plato mapping requirements to technical characteristics

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.

3.2 Interface definitions

One task within the validation framework is to verify the consistency of objects with respect to file format migration of the object. For compliance of that task it is necessary to define appropriate metrics which enable to measure the differences in file characteristics resulting from such preservation actions.

The eXtensible Characterisation Definition Language (XCDL) [13] allows for representation of these file characteristics as object *properties*, derived from extraction of a given file using the eXtensible Characterisation Extraction Language (XCEL) [14], which, as a generic language, is designed to approve the description of file formats from any content domain.

2 Cp. chapter: 4.2 Metrics Specification

As a consequence, one of the first steps towards an automated comparison of these file format based characteristics is to define a set of appropriate metrics, general enough to meet the requirements for such comparisons on file format level and particularly with regard to the specific representation of properties provided by the XCDL. Therefore the *basic metrics for comparison* defined in here are in most cases tried and tested in a wide field of scientific application. Additionally they are extended to the needs of format specific properties as well as to the given XCDL representation.

In a second step, the characteristics extracted to the XCDL representation have to be compared using a subset of the defined set of metrics. Within the evaluation framework it is necessary to define a set of single software units and additional handles (e.g., XML-styled descriptions) to achieve this objective. We call this *Metrics Toolbox*. Currently the Metrics Toolbox consists of two logical subsets, *FPM box* and *Comparison box,* both customized to the demands of the evaluation framework. Nevertheless it is possible to extend and/or customize the Metrics Toolbox in such a way that the core comparison software module can be used by other applications or in a different framework[3].

The Metrics Toolbox is implemented as a bundle of software modules, called *Comparator*. There are two main factors having a main impact on the design and functionality of the Comparator: Firstly, in its current state, the Metrics Toolbox mainly contains tools customized to the requirements of the evaluation framework, therefore this also applies to the Comparator. Secondly, since the Comparator works on XCDL descriptions of objects ('XCDL file') for comparison, it is also reflects closely the structure of XCDL files and other implications of the XCL concepts [13,14][4].

The final specification of the Metrics Toolbox as well as of the Comparator software will be given in a final report in the last planning period.

3.2.2 Metrics Toolbox

The concept of a Metrics Toolbox as well its implementation (Comparator), are work in progress. Therefore this report contains descriptions of the current elements which may *not* be read as final versions. This is especially true for one part of the Metrics Toolbox, the various XML-style descriptions contained in the following sections. We avoid to use the term 'XML file' and use the term 'XML-style description' instead. The reasons for that lie in the dynamic structure of the evaluation framework and the restrictions on certain elements which arise out of this. Some of the structures described below are indeed no persistent entities, i.e. stored as files, but exist as part of dynamic processes within the evaluation framework 'only' in a virtual state, i.e. during the processing of software. According to the status of the overall framework and the comparison software as work in progress, these descriptions successively will be adjusted, therefore we give a detailed description of the *current* state of the art.[5] We also would like to emphasize that the single elements of the Metrics Toolbox are both in their combinability and functionality targeted at the needs of the evaluation framework in general and the work package specific interdependencies in particular. It is intended to extend or generalise its concepts (as well as the implementation) to reflect the requirements of different purposes and frameworks, e.g. also the direct integration into the Planets Testbed.

Figure 7 shows a model of the current design of the Metrics Toolbox. The individual elements of the model are described in the next sections[6].

3 E.g., within the Planets Testbed application

4 Thus, for a better understanding of this report, it is recommended to go through these documents first.

5 Hence the underlying XML schemas will be outlined in the final report (if appropriate to the 'nature' of the particular description).

6 except element 'Comparator Interface' which is simply part of the actual Comparator implementation, nevertheless inserted for better understanding of the model

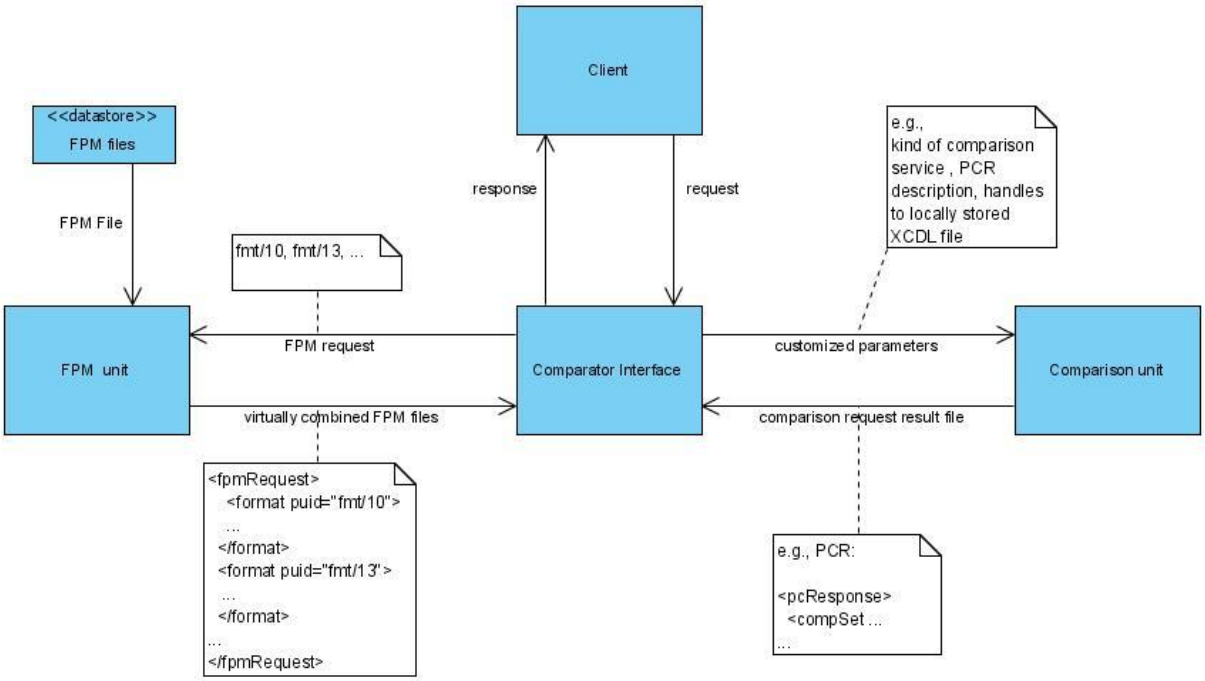

Figure 7: Model of Metrics Toolbox.

3.2.2.1 FPM Box

The FPM Box is a set of tools which are specifically defined and implemented to meet the requirements of the first step of the above described three-step-process.[7] In its core functionality, the tools of the FPM Box provide information on the relations of file formats, properties and metrics[8]. They give us answers to the question: Which properties can be compared by the Comparator for a specific file format and which metrics can be applied to these properties?
The technical representation of the answer to that question is called a FPM file[9].

3.2.2.1.1 FPM File

The basic metrics described in the next section have to be implemented as algorithms to be applied on the extracted XCDL properties. However, within the evaluation framework this is not sufficient. According to the first step of the overall three-steps-process of the evaluation framework[10] it is necessary to assign the defined metrics to certain properties. This is done manually, using a database structure[11] which holds the information about file properties as defined in the XCL system. Furthermore, within this step the evaluation framework requires to relate the properties with their assigned metrics to specific file formats, i.e. to those which have been previously selected as evaluation formats. This is also done manually by relating the defined properties to the file format that supports the property in question[12]. In a semi-automated process a database query is run to produce an XML file containing the property-metrics allocated for each file format that is supported for comparison (short: *FPM file*). Figure 'COMP-01' shows a schematic representation of the structure of such a XML file.

```
<format puid="a_pronom_puid">                    → the file format in question
   <property>                                    → a single supported property
      <id>a_unique_property_id</id>
      <name>a_property_name</name>
```

7 cp. section 'Introduction' in this chapter
7 FPM stands for '(file) Format', 'Property', 'Metric'
9 in this case, the described structure is actually made persistent on local storage, therefore we use the term 'file'.
10 cp. Section ,Introduction' in this chapter
11 using MySQL
12 (this work is part of the PC4 work-package)

```
    <description>the_definition_of_the_property</description>
    <unit>the_unit_of_the_property_if_present</unit>
    <type>the_data_type_of_the_property</type>
    <metrics>                                          → set of assigned metrics
      <m>                                              → a single assigned metric
        <mId>a_unique_metric_id</mId>
        <mName>a_metric_name</mName>
        <mDescription>the_description_of_the_metric</mDescription>
        <mType>the_data_type_used_by_the_metric</mType>
      </m>
      …                                                → more assigned metrics
    </metrics>
  </property>
  …                                                    → more supported properties
</format>
```

Figure 8: Schematic representation of a Format-Property-Metric file (FPM file)

The functions of the single tags within a FPM file are pretty self-describing. Those parts put in italics describe the functionality of the corresponding tag; they are place-holders for a specific value. The format in question is identified using the PRONOM unique identifiers (PUID) [15]. The property and metric IDs are unique within the XCL system and the basic metrics set respectively. <Property> and <m> tags are repeatable since usually more than one single property can potentially be compared and more than one metric can be assigned to a specific property. Figure 9 shows a part of a 'real world' FPM file for the TIF file format[13].

```
<format puid="fmt_10">          → 'fmt_10' stands for TIF file format, version 6.0
<property>
    <id>2</id>
    <name>imageHeight</name>
    <description>Height of an image. Corresponds to the y-axis of a Cartesian coordinate
        system.
    </description>
    <unit>pixel</unit>
    <type>int</type>
    <metrics>
      <m>
        <mId>2</mId>
        <mName>intDiff</mName>
        <mDescription>arithmetical difference of two integer values (A - B)
        </mDescription>
        <mType>int</mType>
      </m>
      <m>
        <mId>10</mId>
        <mName>percDeviation</mName>
        <mDescription>Percental deviation of two values (A,B) of type integer or rational,
            according to the equation: PercDev= d/A*100. A is the source
            value, d is the difference of values B and A (B-A). The output
            value indicates the percental deviation of value B from value A. An output
            value of PercDeviation=0.0 indicates equal values.
        </mDescription>
        <mType>rational</mType>
      </m>
    </metrics>
  </property>
  <property>
    <id>18</id>
    <name>compression</name>
```

13 version 6.0

```
      <description>algorithm applied to image data for the purpose of minimizing storage size
      </description>
      <unit/>
      <type>XCLLabel</type>
      <metrics>
         <m>
            <mId>1</mId>
            <mName>equal</mName>
            <mDescription>simple comparison of two values (A, B) of any type on equality. Type of
output value: Boolean (true, false)
            </mDescription>
            <mType>bool</mType>
         </m>
      </metrics>
   </property>
   …                                                    → more properties to follow
</format>
```

Figure 9: 'Real world' example for a FPM file related to TIF file format, version 6.0 (example is shortened).

3.2.2.1.2 FPM Unit

FPM files are the basic elements handled by a dedicated software unit of the metrics toolbox. We call it a *FPM unit*. Depending on the request for 'FPM information' of the specified file formats (FPM request) the software modules of the FPM unit process one or more locally stored FPM files to virtually[14] create a single XML structure containing the information for all of the requested file formats (*FPM response*). (Figure 10).

```
<fpmResponse>
         <format puid="fmt/10">
         …
         </format>
         <format puid="fmt/13">
         …
         </format>
         …
</fpmResponse>
```

Figure 10: Logical structure of the virtually created FPM Response; actually it is a list of all requested FPM relations (locally stored as FPM files), embedded in XML-style syntax.

3.2.2.1.3 FPM Request

With respect to the intended integration of the Comparator into the Planets Interoperability Framework we have currently developed a test application, simulating a request to the FPM unit as a web service. Figure 11 shows the software components and data flow of the *FPM request* within this test application. In this case, *FPM request service* is the actual interface to the client, in the general Comparator structure this would be the Comparator Interface module (cp. Figure 7). It is implemented as a SOAP-based web service interface[15], to that, in a final step, the XML structure is committed and embedded in the SOAP-envelope to be sent back to the client.

14 i.e., it is not made persistent (written to a storage medium)

15 data is currently handed over by value

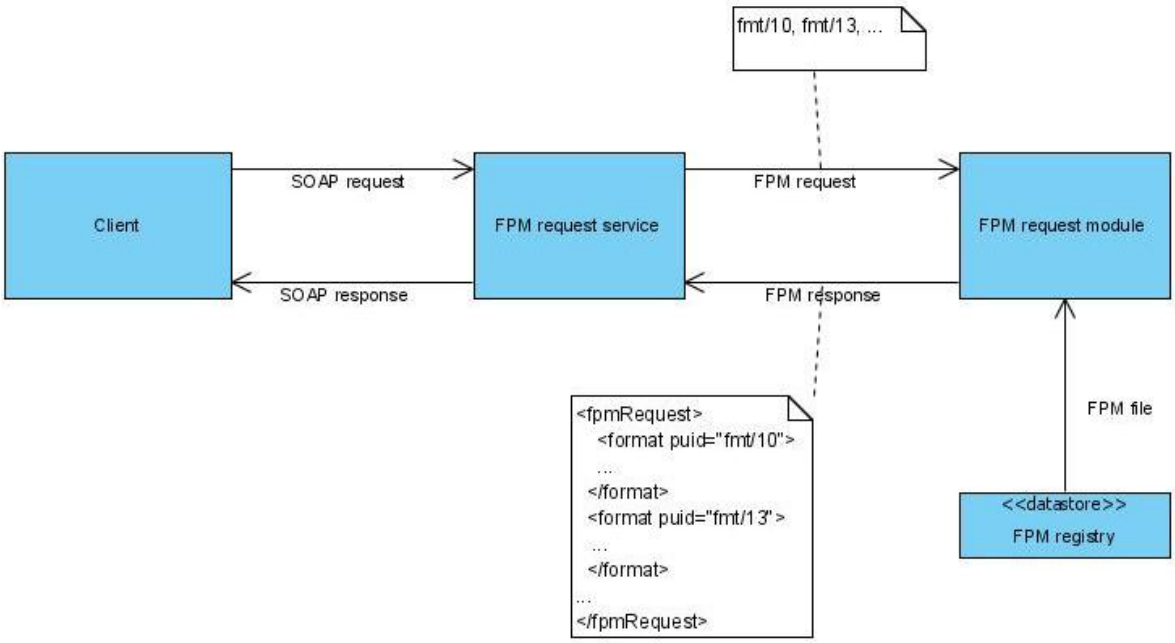

Figure 11: Test implementation of FPM request.

3.2.2.2 Comparison Box

The Comparison Box is that part of the Metrics Toolbox which is related to the actual comparison of the extracted properties. Once the correlations between file format, properties and metrics (FPM file) are provided by the tools of the FPM Box within the evaluation framework, this information is processed according to step two of the overall three-steps-process.[16] In the third and final step, Plato and its processing components of the evaluation framework can use the Comparison Box. It basically comprises a couple of software modules and handles to process the XCDL extracted properties and to finally deliver a result file including the results of the comparison. In the current version the tools of the Comparison Box are customized to the specific interaction of the evaluation framework components.

The core software component for comparison is called *Comparison unit.* It is a bundle of single software modules with different comparison tasks. For the interaction of the evaluation framework components it expects a list of customized input parameters for processing. Each of these parameters is a handle (URIs) to exactly one so called *Plato-to-Comparator Request (PCR)* description plus a number of XCDL files to be compared. Whereas the basic XCDL file for comparison is called source XCDL file, the other XCDL files are called target XCDL files. The terminology refers to one of the overall tasks of the evaluation framework, to give indications on the expected quality of intended file migration in the context of preservation planning. In this case we always have files of an initial file format (the 'source') which are intended to be migrated to a different file format (the 'target'). The Comparison Unit allows for multiple comparisons of files of different formats within a single call, therefore *n* handles to target files are allowed to be passed to the Comparator within any one call.

With respect to the integration within the Planets Interoperability Framework, the Comparison Unit expects these parameters as handles to the accordant files which are actually stored in the dedicated registries of the Planets overall framework[17].

The Comparison unit delivers the results of the comparison procedure as one integrated output unit. If used within the evaluation framework and the Planets Preservation Tool it is called *Comparator-to-Plato Response* (CPR).

16 cp. section 'Introduction' of this chapter

17 The Planets Interoperability Framework defines service interfaces for categories of preservation actions (e.g., validation, migration, characterisation, comparison) and allows for execution of complex workflows.

3.2.2.2.1 Plato-to-Comparator Request (PCR)

A PCR is an XML-style description created within step two of the overall three-step-process of the evaluation framework[18] to enable a comparison of selected properties by specified metrics on files of specific file formats. All possible relations between these instances are defined within the FPM files. A PCR description is a subset of these relations plus additional information on the files to be compared. In a way it acts like a configuration file. It simply tells the Comparison unit of the Comparator which files to choose for comparison, which properties to extract out of these files and which metrics to apply to the selected properties. Figure 12 shows the structure of the tags of the PCR description, 13 gives an example with concrete values.

```
<pcRequest>
  <compSet>
    <source name="the_name_of _the_source_XCDL_file"/>
    <target name="the_name_of_the_first_target_XCDL_file"/>
    <property id="a_unique_property_ID" name="a_property_name"
              unit="an_optional_property_unit">
      <metric id="a_unique_metric_ID" name="a_metric_name" param="an_opional_parameter"/>
      …                                                    → more metrics to follow
    </property>
    …                                                      → more properties to follow
  </compSet>
  <compSet>                    → another set of source/target XCDLs with different target XCDL
    <source name="the_name_of _the_source_XCDL_file"/>
    <target name="the_name_of_the_second_target_XCDL_file"/>
    …
  </compSet>
  …                            → more sets of source/target XCDLs
</pcRequest>
```

Figure 12: Plato-to-Comparator Request description (PCR description, schematic representation).

The parts put in italics describe the functionality of the corresponding tag; they are place-holders for a specific value. The <set> tags are repeatable. They define a pair of source/target files to be compared by the Comparison unit. The absolute location of these files is specified by additional input parameters (see previous section). The <property> tagsets within the <set> tags are also repeatable. They define the single properties which are to be compared. Each opening <property> tag carries two mandatory attributes, indicating name and ID of the property and an additional optional attribute 'unit' that is used if a unit is defined for a property (e.g., property 'imageHeight' is defined to be measured in pixel[19]). Each property can be associated with one or more metrics, specified in <metric> tagsets. In addition to the mandatory attributes 'id' and 'name', an optional parameter 'param' must be added if a metric requires the declaration of one such (see section on specification of metrics below).

```
<pcRequest>
  <compSet>
    <source name="testXCDL1.xml"/>
    <target name="testXCDL2.xml"/>
    <property id="1" name="normData">
      <metric id="11" name="hammingDistance"/>
      <metric id="12" name="simpleMatchCoefficientN" param="3"/>
      <metric id="20" name="RMSE"/>
    </property>
    <property id="2" name="imageHeight" unit="pixel">
      <metric id="1" name="equal"/>
      <metric id="10" name="percDeviation"/>
    </property>
    <property id="18" name="compression">
      <metric id="1" name="equal"/>
```

18 Cp. section: 3.1 Introduction
19 according to XCL names definition

```
      </property>
      <property id="12" name="resolutionUnit">
         <metric id="1" name="equal"/>
      </property>
   </compSet>
</pcRequest>
```

Figure 13: 'Real world' example for PCR declaration, telling the Comparison unit of the Comparator to compare four properties of two XCDL files by the metrics defined in the according <metric> tagsets.

3.2.2.2.2 Comparator-to-Plato Response (CPR)

The results of a comparison initiated by usage of a PCR description are also packed in a single XML-style description (Comparator-to-Plato Response, short: CPR). Figure 14 shows the structure of a CPR description.

```
<cpResponse>
   <compSet source="name_of_compared_source_XCDL_file"
               target="name_of_compared_target_XCDL_file">          → a single comparison set
      <property id="a_unique_property_ID" name="a_property_name"
                  unit="an_optional_property_unit" compStatus="a_predefined_comparison_status">
            <values type="datatype_of_values">
                  <src>a_value_according_to_values_attribute_'type'</src>
                  <tar>a_value_according_to_values_attribute_'type'</tar>
            </values>
            <metrics>
                  <metric id="a_unique_metric_ID" name="a_metric_name"
                     result="result_value_of_comparison_according_to_assigned_metric_datatype"/>
                  <metric id="a_unique_metric_ID" name="a_metric_name"
                           error="a_specific_error_code_for_failed_comparison_on_this_metric"/>
                  …                                  → more metrics to follow
            </metrics>
      </property>
      …                                              → more properties to follow
   </compSet>
   …                                                 → more comparison pairs to follow
</cpResponse>
```

Figure 14: Structure of CPR description.

The <compSet> tags are repeatable, as required by the number of requested comparisons (<compSet> tagsets in PCR). Two attributes are required, the names of the source and target XCDL files. For each property which is listed for comparison in the PCR, a counterpart comprising the results of the comparison for that property is listed within the <compSet> tagsets of the CPR.

For the <property> tag, three attributes are required: an identifier ('id', predefined through the XCL system), the name of the property ('name') and a comparison status ('compStatus') with predefined values as follows: value '*failed*' is set if the entire comparison for property X failed. This is always the case if one of the two XCDLs does not provide the requested property. Value '*partial*' is set if property X is contained in both XCDLs but not all of the required metrics could be provided. In this case, the attribute 'error' in the <metric> tag is set (see below). Value '*complete*' is set if the entire comparison for property X could be performed. The attribute 'unit' is optional for the case of a property that requires a unit (e.g., 'imageHeight' is measured in unit 'pixel').

Within the <property> tagset, two child tagsets are defined: <values> contains all the information related to compared data. The elements <src> and <tar> contain the values of the compared properties, as indicated in the XCDL files. Attribute 'type' of tag <values> takes the data type for the source and target values. The <values> tagset occurs exactly once for each <property> tagset.

The second child tagset <metrics> contains the repeatable <metric> element, according to the requested metrics in PCR. Four attributes are defined for <metric>: 'name' and 'id' are required, 'result' and 'error' are used depending on the context. If the Comparison unit is able to process the

requested metrics, the ‘result’ attribute is used, indicating the result for the comparison based on the specified metric. If not so, the ‘error’ attribute is used, listing a specific error code.

Figure 15 shows an example for a CPR description (the actual response to the PCR example in Figure 13). The both XCDL files on which the comparison in this example is based on are listed in Appendix A.

```
<cpResponse>
   <compSet source="testXCDL1.xml" target="testXCDL2.xml">
      <property id="18" name="compression" compStatus="complete">
         <values type="XCLLabel">
            <src>zlibDeflateInflate</src>
            <tar>uncompressed</tar>
         </values>
         <metrics>
            <metric id="1" name="equal" result="false"/>
         </metrics>
      </property>
      <property id="2" name="imageHeight" unit="pixel" compStatus="complete">
         <values type="int">
            <src>32</src>
             <tar>32</tar>
          </values>
          <metrics>
             <metric id="1" name="equal" result="true"/>
             <metric id="10" name="percDeviation" result="0.000000"/>
          </metrics>
      </property>
      <property id="1" name="normData" compStatus="complete">
         <values type="int">
            <src>00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17
                  18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
                  ...      → 976 hex numbers more to follow
            </src>
            <tar>00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17
                  18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
                  ...      → 976 hex numbers more to follow
            </tar>
         </values>
         <metrics>
           <metric id="11" name="hammingDistance" result="0"/>
           <metric id="12" name="simpleMatchCoefficientN#3" result="0.000000"/>
           <metric id="20" name="RMSE" result="0.000000"/>
         </metrics>
      </property>
      <property id="12" name="resolutionUnit" compStatus="failed"/>
   </compSet>
</cpResponse>
```

Figure 15: Example for a CPR description (the response to the PCR example of COMP-06). The XCDL files which are the basis for the example are listed in Appendix A. For a more convenient representation, the content of <src> and <tar> are shortened (see related XCDL files in Appendix A for full representation).

3.3 Summary

This section described the basic evaluation framework as it is currently defined. We outlined the main component involved and provided the interface definitions detailing how these components are or will be interacting, illustrating the exchanged messages with sample requests and responses. The next section describes the metrics that are used for property comparison.

4. Basic Metrics Set for Comparing Properties

4.1 Categories of Comparison

We have already pointed out the main factors having an influence on the design and functionality of the Metrics Toolbox and Comparator. In a similar way, this is also true for the specification of the basic comparison metrics. In this case, the specifications of the XCL system in general and the structure of the XCDL[13] in particular have to be considered.

According to this we can identify a couple of different *categories of comparison*. These are comparison on level of

1) single properties (one-to-one)
2) sets of properties (one-to-many, many-to-many)
3) references of properties to XCL normalised data
4) complex objects
5) XCDL data categories (labelled values, normalised data)
6) XCDL data types
7) 'nature' of property values (single value, set of values)
8) multiple relations on the categories just mentioned

In this report we specify a basic set of metrics which meet the categories 1,5,6 and 7, at least in parts. To meet the requirements for comparison on the rest of the identified levels, this set of basic metrics has to be extended. New metrics have to be added, existing metrics have to be combined. Particularly the realisation of comparisons on the last category mentioned above is essential to enable the Comparator to make general statements on the nature of the migrated objects (e.g., similarity of objects as a whole and in parts) compared to the source object. Nevertheless, the currently supported levels meet the requirements on the evaluation framework components already quite well. A final specification of all supported levels of comparison as well as the specification of the final metrics set will be given during the last planning period of the Planets project.

We have divided the basic set of metrics into three sub-groups (A-C). Group A is dedicated to arithmetical metrics, which give us basic results on comparisons of properties in a simple and intuitive way. Group B is a set of metrics referring to distance / similarity measurement. They are also fundamental in this scope. Group C is an initial set of basic statistical measures[20].

4.2 Metrics Specification

In the following section all metrics of the basic metrics set are specified. Each metric is listed by its name. The specification parts contain at least a verbal explanation, if necessary we also show the formula for calculation. The data types of input and output values are listed as well. Additionally we give an example for the metric, either as table-style examples, where the original XCDL parts and the expected CPR output parts are exposed or as an example for computation.

The variable 'A' always stands for the source value of comparison, 'B' relates to the target value.

4.2.1 Group A: Arithmetical Metrics

Metric name: equal

Explanation: Metric 'equal' is a simple comparison of two values (A, B) of any XCL data type on equality.
Data type of input value: Any XCL data type
Data type of output value: XCL: boolean (true, false)

20 All of the three groups (especially groups B and C) are only a sample within their domain. We consider them to be one good choice for this basic metrics set among other also appropriate alternatives.

Example:

Value for property X of XCDL1 (src)	Value for property X of XCDL2 (tar)
<labVal> <val>32</val> <type>int</type> </labVal>	<labVal> <val>32</val> <type>int</type> </labVal>
CPR output: … <property id="2" name="imageHeight" unit="pixel" compStatus="complete"> <values type="int"> <src>32</src> <tar>32</tar> <values> <metrics> <metric id="1" name="equal" result="true"/> </metrics> </property> …	

Metric name: intDiff

Explanation: Arithmetical difference of two values (A, B) of data type XCL:int.
Formula:
$A - B$

where A is the source value, B is the target value.

Data type of input value: XCL:int
Data type of output value: XCL:int

Example:

Value for property X of XCDL1 (src)	Value for property X of XCDL2 (tar)
<labVal> <val>32</val> <type>int</type> </labVal>	<labVal> <val>32</val> <type>int</type> </labVal>
CPR output: … <property id="2" name="imageHeight" unit="pixel" compStatus="complete"> <values type="int"> <src>32</src> <tar>32</tar> <values> <metrics> <metric id="2" name="intDiff" result="0"/> </metrics> </property> …	

Metric name: ratDiff

Explanation: Arithmetical difference of two values (A, B) of data type XCL:rational.
Formula:
$A - B$

where A is the source value, B is the target value.

Data type of input value: XCL:rational

Data type of output value: XCL:rational
Example: (see metric 'intDiff')

Metric name: intSum

Explanation: Arithmetical sum of two values (A, B) of data type XCL:int.
Formula:
$A + B$

where A is the source value, B is the target value.

Data type of input value: XCL:int
Data type of output value: XCL:int
Example: (cp. metric 'intDiff')

Metric name: ratSum

Explanation: Arithmetical sum of two values (A, B) of data type XCL:rational.
Formula:
$A + B$

where A is the source value, B is the target value.

Data type of input value: XCL:rational
Data type of output value: XCL:rational
Example: (cp. metric 'intDiff')

Metric name: intRatio

Explanation: Quotient of two values (A, B) of data type XCL:int.
Formula:
A / B

where A is the source value, B is the target value.

Data type of input value: XCL:int
Data type of output value: XCL:rational

Example:

Value for property X of XCDL1 (src)	Value for property X of XCDL2 (tar)
<labVal> <val>32</val> <type>int</type> </labVal>	<labVal> <val>32</val> <type>int</type> </labVal>
CPR output: … <property id="2" name="imageHeight" unit="pixel" compStatus="complete"> <values type="int"> <src>32</src> <tar>32</tar> <values> <metrics> <metric id="6" name="intRatio" result="1.0"/> </metrics> </property> …	

Metric name: ratRatio

Explanation: Quotient of two values (A, B) of data type XCL:rational.
Formula:
A / B

where A is the source value, B is the target value.

Data type of input value: XCL:rational
Data type of output value: XCL:rational
Example: (cp. metric 'intRatio')

Metric name: intProd

Explanation: Product of two values (A, B) of data type XCL:int.
Formula:
$A * B$

where A is the source value, B is the target value.

Data type of input value: XCL:int
Data type of output value: XCL:int
Example: (cp. previous examples)

Metric name: ratProd

Explanation: Product of two values (A, B) of data type XCL:rational.
Formula:
$A * B$

where A is the source value, B is the target value.

Data type of input value: XCL:rational
Data type of output value: XCL:rational
Example: (cp. previous examples)

Metric name: percDeviation

Explanation: Percental deviation of two values (A,B) of data type XCL:int or XCL:rational. The output value indicates the percental deviation of *value B from value A*. An output value of PercDeviation=0.0 indicates *equal* values.
Formula:
$d / A * 100$

where A is the source value, d is the difference of values B and A (B - A) .

Data type of input value: XCL:int, XCL:rational
Data type of output value: XCL:rational

Example:

Value for property X of XCDL1 (src)	Value for property X of XCDL2 (tar)
<labVal> <val>32</val> <type>int</type> </labVal>	<labVal> <val>8</val> <type>int</type> </labVal>
CPR output: …	

```
<property id="2" name="imageHeight" unit="pixel" compStatus="complete">
   <values type="int">
      <src>32</src>
      <tar>8</tar>
   <values>
   <metrics>
      <metric id="10" name="percDeviation" result="-75.0"/>
   </metrics>
</property>
…
```

In this example, the result indicates that the target value B is 75% less of value A; or in other words, B is a forth of value A.

4.2.2 Group B: Similarity / Distance Metrics

The following measures basically provide information on similarity or dissimilarity (distance) of two *sets* of values.

Metric name: hammingDistance

Explanation: The hamming distance counts the number of non-corresponding symbols of two ordered sequences. The sequences must have equal length; hammingDistance=0 indicates equal sequences, i.e. no substitutions are required to change a sequence into the other. Hamming distance is often used for string comparison.
Data type of input value: XCL:int, XCL:rational, XCL:string
Data type of output value: XCL:int

Example:
1.
Let A and B be sets of values with sequentially ordered integer values:

A={255, 80, 0, 255, 80, 0}
B={255, 80, 0, 255, 80, 255}

Comparison scheme:

Value pos#	Set A	Set B	Partial Comp. Result
1	255	255	true
2	80	80	true
3	0	0	true
4	255	255	true
5	80	80	true
6	0	255	false

The values on position #6 differ, all others equal. Thus, the result for the hamming distance is: d_{ham}= 1.

2.
Let A and B be strings (set of characters) with sequentially ordered char values:

A= Heydegger
B= Heidecker

Comparison scheme:

Value pos#	Chars of string A	Chars of string B	Partial Comp. Result
1	H	H	true
2	e	E	true
3	y	I	false
4	d	D	true
5	e	E	true

6	g	C	false
7	g	K	false
8	e	E	true
9	r	R	true

The values on position #3, 6 and 7 differ, all others equal. Thus, the result for the hamming distance is: $d_{ham} = 3$.

Metric name: simpleMatchCoefficientN

Explanation: Simple match coefficient is a distance measure that uses the hamming distance. In addition, the ratio of non-corresponding values (hamming distance) to the total number of compared pairs of values is calculated. N indicates the n-grams used for comparison. Default setting is N=1. [21] The result is a rational number from range 0.0 to 1.0, indicating maximum similarity (0.0) to minimum (1.0).
Formula:
$d(A,B)_{sm} = d_{ham} / t$

where
$d(A,B)_{sm}$ is the simple match coefficient,
d_{ham} the hamming distance,
t the total number of compared pairs of values.

Data type of input value: XCL:int, XCL:rational, XCL:string
Data type of output value: XCL:rational

Example:
1. N=1

Let A and B be sets with sequentially ordered integer values:

A={255, 80, 0, 255, 80, 0, 255, 80, 0, 255, 80, 0}
B={255, 80, 0, 255, 80, 0, 255, 80, 0, 0, 255, 80}

$d(A,B)_{sm} = 3/12 = 0.25$ (The two sets differ in 25%, they are similar to 75%.)

2. N=2
In addition to simpleMatchCoefficientN where N=1, it is presumed that there are always pairs of two[22] that form a comparing unit. The comparison of each dyad of set A and set B matches, if the single values of the dyad match. The order of the values within the dyad is significant for the comparison. The formula for computing simple match coefficient still remains the same.

Let A and B be sets with sequentially ordered integer values:

A={255, 80, 0, 255, 80, 0, 255, 80, 0, 255, 80, 0}
B={255, 80, 0, 255, 80, 0, 255, 80, 255, 0, 255, 80}

In this example 6 pairs of set A are compared with the corresponding dyads of set B. The last two pairs do not match. The comparison of the fifth pairs of two fail although the two single values are the same in both, since the order of the two values within the dyad is significant for this measure. The last (sixth) comparison fails as well. Hence, the result is:

$d(A,B)_{sm2} = 2/6 = 0.3333$ (The two sets differ in a third of their values, based on N=2)

21 Note: In the PCR, N is transmitted to the Comparator via the 'param' attribute within the <metric> tag (cp. section on PCR description above).
22 In the following we use the terms 'dyad' and 'bigram' as synonyms for 'pairs of two'

Metric name: simpleMatchCoefficientExtN

Explanation: This measure is an extension of the simpleMatchCoefficient. In addition it is presumed that the order of the values of the sets is not fixed. Thus, the hamming distance is not restricted to the positions of the single values within the set. N indicates the n-grams used for comparison. Default setting is N=1. Type of output value: rational.
Data type of input value: XCL:int, XCL:rational, XCL:string
Data type of output value: XCL:rational

Example:
1. N=1

Let A and B be sets with integer values:

A={255, 80, 0, 255, 80, 0, 255, 80, 0, 255, 80, 0}
B={80, 80, 80, 80, 255, 255, 255, 255, 0, 0, 0, 0}

Sets A and B contain exactly the same values in the same quantity: each with four times of value 255, 80 and 0. Each single value of set A has exactly one counterpart in set B. In this case, the order within the sets is not significant. Thus,

$d_{ham} = 0$ and
$d(A,B)_{sme} = 0/12 = 0.0$ (The two sets are maximum similar.)

2. N=2
In addition to simpleMatchCoefficientExtN where N=1 it is presumed that there are always pairs of two that form a comparing unit. The comparison of each dyad of set A and set B matches, if the single values of the dyad match. The order of the values within the pair of two is significant for the comparison, the order within the total set *still not*. The formula for computing simple match coefficient still remains the same.

Let A and B be sets with integer values:

A={255, 80, 0, 255, 80, 0, 255, 80, 0, 255, 80, 0}
B={80, 80, 80, 80, 255, 255, 255, 255, 0, 0, 0, 0}

Sets A contains the pairs of two:
A1={255, 80} two times
A2={00, 255} two times
A3={80, 00} two times

Set B contains the pairs of two:
A1={80, 80} two times
A2={255, 255} two times
A3={00,00} two times

Although the single values in both set are exactly the same in quantity, the result for this example is

$d_{ham} = 6$ and
$d(A,B)_{sme2} = 6/6 = 1.0$ (The two sets are maximum dissimilar.)

because unit of the comparison is in this case pair of two. None of the dyads in set A match with any of those in set B.

Metric name: levenshteinDistance

Explanation: The Levenshtein distance[23] is a distance measure for strings. Two strings are compared with respect to the three basic operations insert, delete and replace in order to

23 syn.: Edit distance

transform string A into string B. The value for this metric is the number of operations needed for transformation.
Data type of input value: XCL:string
Data type of output value: XCL: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: $d_{lev} = 2$.

Metric name: blockDistance

Explanation: Block distance[24] is a distance measure. The distance of two sets is the summed up absolute differences of the values of two sets. Type of output value: integer.
Data type of input value: XCL:int, XCL:rational
Data type of output value: XCL:rational

Formula:

$$d_{ij} = \sum_{k=1}^{n} | x_{ik} - x_{jk} |$$

Example:

Let there be two sets
A={255, 0, 100, 100, 0, 255}
B={255, 100, 0, 0, 100, 255}

The block distance for this example is
$D_{block(A,B)}$ = |255-255| + |0-100| + |100-0| + |100-0| + |0-100| + |255-255|
= 0 +100 +100 + 100 + 100 + 0
= 400

Metric name: euclidDistance

Explanation: Euclidian distance[25] is a distance measure for the shortest distance between two points.
Formula:

$$d_{euclid} = \sqrt{\sum_{k=1}^{n} \left(x_{ik} - x_{jk} \right)^2}$$

where
d_{euclid} is the Euclidian distance of two sets i and j,
x_{ik} and x_{jk} are the set values at position k.

Data type of input value: XCL:int, XCL:rational
Data type of output value: XCL:rational
Example:

24 syn.: L1 distance, City block distance, Manhattan distance, Taxicab distance
25 syn.: L2 distance

Let there be two sets
A= {3, 5, 7, 2}
B= {2, 5, 4, 3}

The Euclidian distance is square root of
(3-2)^2 + (5-5)^2+(7-4)^2+(2-3)^2

D_{euclid} = 3.3166 (rounded to four decimal places)

Metric name: JaccardDistance

Explanation: Measure for dissimilarity of two sets. The difference of the sizes of the union and the intersection of two sets is devided by the size of the union. The compared sets are maximum dissimilar at value 1.0.
Formula:

$d_{Jacc} = (|A \cup B| - |A \cap B|) / |A \cup B|$

Data type of input value: XCL:int, XCL:rational, XCL:string
Data type of output value: XCL:rational
Example:

Let A and B be sets
A={ 1, 3, 5, 7, 9}
B={ 2, 3 , 4, 5, 6}

$A \cup B$ = {1,2,3,4,5,6,7,9}
$A \cap B$ = {3,5}

The Jaccard distance is:
d_{Jacc} = (8 – 2) / 8 = 0.75 (The dissimilarity of the two sets is 75%.)

Metric name: diceCoefficient

Explanation: Measure for similarity of two sets. Dice coefficient is similar to Jaccard distance (which uses the Jaccard coefficient); in difference, the agreement is weighted twice. The compared sets are maximum similar at value 1.0.
Formula:

$s_{dice} = (2* |A \cap B|) / (|A| + |B|)$

Data type of input value: XCL:int, XCL:rational, XCL:string
Data type of output value: XCL:rational
Example:

1.
Let A and B be sets

A= {40, 80 ,40, 255, 10, 100}
B= {40, 80, 40, 0, 10, 255}
and let us assume that A and B contains triplets of values:
A1={40, 80, 40}
A2={255, 10, 100}
B1={40, 80, 40}
B2={0, 10, 255}

Dice coefficient for this example is:
s_{dice}= (2*1) / (2+2) = 0.5 (A and B are in 50% similar.)

2.
Let A and B be strings (sets of characters)

A= byte
B= beyl

The Dice coefficient for this example is:
s_{dice}= (2*3) / (4+4) = 0.75 (Similarity of 75%)

4.2.3 Group C: Statistical Metrics

Metric name: meanAbsoluteDeviation

Explanation: Mean absolute deviation (MAD) is a measure for statistical spread of a univariate set (sample). It is defined through the distances of each single value from the mean. These distances are then used for measuring by calculating the average distance as arithmetic mean.
Formula:

$$\frac{1}{n}\sum_{i=1}^{n}|x_i - m(x)|$$

where
xi – m(X) is the distance of value i from the mean,
n the number of values in the set.

Data type of input value: XCL:int, XCL:rational
Data type of output value: XCL:rational

Metric name: meanAbsoluteDeviationRat

Explanation: Mean absolute deviation Ratio is the ratio of the mean absolute deviations of two compared sets (A, B).
Formula:
$d_{MAD(A,B)} = MAD_A / MAD_B$

Data type of input value: XCL:int, XCL:rational
Data type of output value: XCL:rational
Example:

Let A and B be sets

A= {8, 8 ,4, 12, 12, 4}
B= {8, 12, 6, 6, 8, 8}

The mean of set A is 8, of set B also 8.

$d_{MAD(A,B)}$ = ((|8-8| + |8-8| + |4-8| + |12-8| + |12-8| + |4-8|) / 6) - ((|8-8| + |12-8| + |6-8| + |6-8| + |8-8| + |8-8|) /6)
= (16/6) / (8/6) = 2

(The deviation of set A values from their mean is twice as high as for set B values.)

Metric name: standardDeviation

Explanation: Standard deviation is a statistical measure for the distribution of the values of a random variable from the arithmetical mean.

Formula:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2}{n}}$$

Data type of input value: XCL:int, XCL:rational
Data type of output value: XCL:rational

Metric name: RMSE

Explanation: Root mean squared error is a widely used measure for deviation. It is obtained by the square root of the average of the squared differences of single values of two sets.
Formula:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\left(x_{Ai} - x_{Bi}\right)^2}{n}}$$

where
x_{Ai} is a single value of value set A
x_{Bi} is a single value of value set B
n is the number of the values in the sets (must have same number)

Data type of input value: XCL:int, XCL:rational
Data type of output value: XCL:rational
Example:

Let A and B be sets

A= {8, 8 ,4, 12, 12, 4}
B= {8, 12, 6, 6, 8, 8}

$$RMSE = \sqrt{\frac{(0+16+4+36+16+16)}{6}}$$

$$= \sqrt{\frac{88}{6}}$$

= 14,666667

5. Summary and Outlook

This document described the basic metric and evaluation framework developed in the Planets work package PP5. We outlined the basic concepts, described significant properties in the context of preservation planning and referenced the work done in characterisation that enables the validation framework to compare digital objects and compute metrics on technical characteristics.

We then did show how the components are interconnected to provide an evaluation framework for preservation planning and, finally, described the basic methods for comparison.

The following steps will be taken during the next project period:

1. **Extension to other characteristics.**

In analogy to the layers shown in Figure 4, a similar approach is envisioned for mapping requirements to other measurable criteria. One example for these is the risk factors that can be assessed by the risk assessment service developed in PP4. It is intended to extend the validation framework to cover these characteristics.

2. Extension/ Adjustment of the Metrics Toolbox and Comparator.

The Metrics Toolbox is, as well as the actual realisation as software (Comparator), work in progress. Therefore this report contains descriptions of the current elements which may *not* be read as final versions. According to the status of the overall framework and the comparison software as work in progress, the Metrics Toolbox successively will be adjusted. We also would like to emphasize that the single elements of the Metrics Toolbox are both in its combination and functionality connected to the needs of the evaluation framework in general and the work package specific interdependencies in particular. It is intended to extend or generalise it (as well as the software) to the implications of different purposes and frameworks, e.g. the integration into the Planets testbed. Therefore we also will specify an interface to the Comparator software which will be customized for such extensions. This will also ease the integration of the Comparator into the Planets Interoperability framework.

Depending on the status of embedding of the Comparator and the required related software ('Extractor', migration tools) into the Planets Interoperability framework and the availability of XCDL documents via the Planets PC registry we also intend to extend the Metrics Toolbox for a solution which enables to make use of cached objects (e.g., a key pointing to an internal cache, where the XCDL document in question is being held already; ideally this will be the PC registry as described in the overall Planets work plan).

3. Further development of metrics

Currently we have defined and implemented a basic set of comparison metrics which enable to operate on a certain number of categories of comparison.[26] In the next stage of the evaluation framework we will extend this set to arrive at definitions and implementations of summary metrics which enable to compare a set of properties using a particular set of basic metrics. These summary metrics should finally be of such comprehensiveness that they allow for general statements on migration results (fulfilling category #8 as defined in section 4.1 above).

[26] cp. section 4.1 Categories of Comparison

Appendix 1

1. XCDL files, used for examples of section 'Metrics Toolbox'.[27]

testXCDL1.xml :


```
<?xml version='1.0' encoding='UTF-8'?>
<xcdl xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.planets-
project.eu/xcl/schemas/xcl" xsi:schemaLocation="http://www.planets-project.eu/xcl/schemas/xcl
XCDLCore.xsd" id="0" >
   <object id="o1" >
      <normData id="nd1" >00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16
17
18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f 40 41 42 43 44 45 46 47
48 49 4a 4b 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77
78 79 7a 7b 7c 7d 7e 7f 80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f
90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f a0 a1 a2 a3 a4 a5 a6 a7
a8 a9 aa ab ac ad ae af b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf
c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce cf d0 d1 d2 d3 d4 d5 d6 d7
d8 d9 da db dc dd de df e0 e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed ee ef
f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff fe fd fc fb fa f9 f8 f7
f6 f5 f4 f3 f2 f1 f0 ef ee ed ec eb ea e9 e8 e7 e6 e5 e4 e3 e2 e1 e0 df
de dd dc db da d9 d8 d7 d6 d5 d4 d3 d2 d1 d0 cf ce cd cc cb ca c9 c8 c7
c6 c5 c4 c3 c2 c1 c0 bf be bd bc bb ba b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 af
ae ad ac ab aa a9 a8 a7 a6 a5 a4 a3 a2 a1 a0 9f 9e 9d 9c 9b 9a 99 98 97
96 95 94 93 92 91 90 8f 8e 8d 8c 8b 8a 89 88 87 86 85 84 83 82 81 80 7f
7e 7d 7c 7b 7a 79 78 77 76 75 74 73 72 71 70 6f 6e 6d 6c 6b 6a 69 68 67
66 65 64 63 62 61 60 5f 5e 5d 5c 5b 5a 59 58 57 56 55 54 53 52 51 50 4f
4e 4d 4c 4b 4a 49 48 47 46 45 44 43 42 41 40 3f 3e 3d 3c 3b 3a 39 38 37
36 35 34 33 32 31 30 2f 2e 2d 2c 2b 2a 29 28 27 26 25 24 23 22 21 20 1f
1e 1d 1c 1b 1a 19 18 17 16 15 14 13 12 11 10 0f 0e 0d 0c 0b 0a 09 08 07
06 05 04 03 02 01 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11
12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27 28 29
2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f 40 41
42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59
5a 5b 5c 5d 5e 5f 60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71
72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f 80 81 82 83 84 85 86 87 88 89
8a 8b 8c 8d 8e 8f 90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f a0 a1
a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af b0 b1 b2 b3 b4 b5 b6 b7 b8 b9
ba bb bc bd be bf c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce cf d0 d1
d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de df e0 e1 e2 e3 e4 e5 e6 e7 e8 e9
ea eb ec ed ee ef f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff fe fd
fc fb fa f9 f8 f7 f6 f5 f4 f3 f2 f1 f0 ef ee ed ec eb ea e9 e8 e7 e6 e5
e4 e3 e2 e1 e0 df de dd dc db da d9 d8 d7 d6 d5 d4 d3 d2 d1 d0 cf ce cd
cc cb ca c9 c8 c7 c6 c5 c4 c3 c2 c1 c0 bf be bd bc bb ba b9 b8 b7 b6 b5
b4 b3 b2 b1 b0 af ae ad ac ab aa a9 a8 a7 a6 a5 a4 a3 a2 a1 a0 9f 9e 9d
9c 9b 9a 99 98 97 96 95 94 93 92 91 90 8f 8e 8d 8c 8b 8a 89 88 87 86 85
84 83 82 81 80 7f 7e 7d 7c 7b 7a 79 78 77 76 75 74 73 72 71 70 6f 6e 6d
6c 6b 6a 69 68 67 66 65 64 63 62 61 60 5f 5e 5d 5c 5b 5a 59 58 57 56 55
54 53 52 51 50 4f 4e 4d 4c 4b 4a 49 48 47 46 45 44 43 42 41 40 3f 3e 3d
3c 3b 3a 39 38 37 36 35 34 33 32 31 30 2f 2e 2d 2c 2b 2a 29 28 27 26 25
24 23 22 21 20 1f 1e 1d 1c 1b 1a 19 18 17 16 15 14 13 12 11 10 0f 0e 0d
0c 0b 0a 09 08 07 06 05 04 03 02 01 00 01 02 03 </normData>
      <property id="11" source="raw" cat="descr" >
         <name>bitDepth</name>
```


27 Note: Both examples apply to the initial XCDL specification and may slightly differ to the final XCDL specification (version 1.0, 31th May 2008). We used the inital version since at the time of preparing this report the XCDL specification was also in preparation.

```
    <valueSet id="i_i1_s4" >
      <labValue>
        <val>8</val>
        <type>int</type>
      </labValue>
      <dataRef ind="normAll"/>
    </valueSet>
  </property>
  <property id="18" source="raw" cat="descr" >
    <name>compression</name>
    <valueSet id="i_i1_s6" >
      <labValue>
        <val>zlibDeflateInflate</val>
        <type>fixedLabel</type>
      </labValue>
      <dataRef ind="normAll" />
    </valueSet>
  </property>
  <property id="2" source="raw" cat="descr" >
    <name>imageHeight</name>
    <valueSet id="i_i1_s3" >
      <labValue>
        <val>32</val>
        <type>int</type>
      </labValue>
      <dataRef ind="normAll" />
    </valueSet>
  </property>
  <property id="20" source="raw" cat="descr" >
    <name>imageType</name>
    <valueSet id="i_i1_s5" >
      <labValue>
        <val>greyscale</val>
        <type>fixedLabel</type>
      </labValue>
      <dataRef ind="normAll" />
    </valueSet>
  </property>
  <property id="30" source="raw" cat="descr" >
    <name>imageWidth</name>
    <valueSet id="i_i1_s2" >
      <labValue>
        <val>32</val>
        <type>int</type>
      </labValue>
      <dataRef ind="normAll" />
    </valueSet>
  </property>
  <property id="39" source="raw" cat="descr" >
    <name>filter</name>
    <valueSet id="i_i1_s7" >
      <labValue>
        <val>adaptive</val>
        <type>fixedLabel</type>
      </labValue>
      <dataRef ind="normAll" />
    </valueSet>
  </property>
  <property id="41" source="raw" cat="descr" >
    <name>gamma</name>
    <valueSet id="i_i1_s10" >
      <labValue>
```

```
            <val>100000</val>
            <type>int</type>
          </labValue>
          <dataRef ind="normAll" />
        </valueSet>
      </property>
      <property id="45" source="raw" cat="descr" >
        <name>interlace</name>
        <valueSet id="i_i1_s8" >
          <labValue>
            <val>adam7</val>
            <type>fixedLabel</type>
          </labValue>
          <dataRef ind="normAll" />
        </valueSet>
      </property>
   </object>
</xcdl>
```

testXCDL2.xml :

```
<?xml version='1.0' encoding='UTF-8'?>
<xcdl xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.planets-
project.eu/xcl/schemas/xcl" xsi:schemaLocation="http://www.planets-project.eu/xcl/schemas/xcl
XCDLCore.xsd" id="0" >
   <object id="o1" >
      <normData id="nd1" >00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16
17
18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f 40 41 42 43 44 45 46 47
48 49 4a 4b 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77
78 79 7a 7b 7c 7d 7e 7f 80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f
90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f a0 a1 a2 a3 a4 a5 a6 a7
a8 a9 aa ab ac ad ae af b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf
c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce cf d0 d1 d2 d3 d4 d5 d6 d7
d8 d9 da db dc dd de df e0 e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed ee ef
f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff fe fd fc fb fa f9 f8 f7
f6 f5 f4 f3 f2 f1 f0 ef ee ed ec eb ea e9 e8 e7 e6 e5 e4 e3 e2 e1 e0 df
de dd dc db da d9 d8 d7 d6 d5 d4 d3 d2 d1 d0 cf ce cd cc cb ca c9 c8 c7
c6 c5 c4 c3 c2 c1 c0 bf be bd bc bb ba b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 af
ae ad ac ab aa a9 a8 a7 a6 a5 a4 a3 a2 a1 a0 9f 9e 9d 9c 9b 9a 99 98 97
96 95 94 93 92 91 90 8f 8e 8d 8c 8b 8a 89 88 87 86 85 84 83 82 81 80 7f
7e 7d 7c 7b 7a 79 78 77 76 75 74 73 72 71 70 6f 6e 6d 6c 6b 6a 69 68 67
66 65 64 63 62 61 60 5f 5e 5d 5c 5b 5a 59 58 57 56 55 54 53 52 51 50 4f
4e 4d 4c 4b 4a 49 48 47 46 45 44 43 42 41 40 3f 3e 3d 3c 3b 3a 39 38 37
36 35 34 33 32 31 30 2f 2e 2d 2c 2b 2a 29 28 27 26 25 24 23 22 21 20 1f
1e 1d 1c 1b 1a 19 18 17 16 15 14 13 12 11 10 0f 0e 0d 0c 0b 0a 09 08 07
06 05 04 03 02 01 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11
12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27 28 29
2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f 40 41
42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59
5a 5b 5c 5d 5e 5f 60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71
72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f 80 81 82 83 84 85 86 87 88 89
8a 8b 8c 8d 8e 8f 90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f a0 a1
a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af b0 b1 b2 b3 b4 b5 b6 b7 b8 b9
ba bb bc bd be bf c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce cf d0 d1
d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de df e0 e1 e2 e3 e4 e5 e6 e7 e8 e9
ea eb ec ed ee ef f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff fe fd
fc fb fa f9 f8 f7 f6 f5 f4 f3 f2 f1 f0 ef ee ed ec eb ea e9 e8 e7 e6 e5
e4 e3 e2 e1 e0 df de dd dc db da d9 d8 d7 d6 d5 d4 d3 d2 d1 d0 cf ce cd
```

cc cb ca c9 c8 c7 c6 c5 c4 c3 c2 c1 c0 bf be bd bc bb ba b9 b8 b7 b6 b5
b4 b3 b2 b1 b0 af ae ad ac ab aa a9 a8 a7 a6 a5 a4 a3 a2 a1 a0 9f 9e 9d
9c 9b 9a 99 98 97 96 95 94 93 92 91 90 8f 8e 8d 8c 8b 8a 89 88 87 86 85
84 83 82 81 80 7f 7e 7d 7c 7b 7a 79 78 77 76 75 74 73 72 71 70 6f 6e 6d
6c 6b 6a 69 68 67 66 65 64 63 62 61 60 5f 5e 5d 5c 5b 5a 59 58 57 56 55
54 53 52 51 50 4f 4e 4d 4c 4b 4a 49 48 47 46 45 44 43 42 41 40 3f 3e 3d
3c 3b 3a 39 38 37 36 35 34 33 32 31 30 2f 2e 2d 2c 2b 2a 29 28 27 26 25
24 23 22 21 20 1f 1e 1d 1c 1b 1a 19 18 17 16 15 14 13 12 11 10 0f 0e 0d
0c 0b 0a 09 08 07 06 05 04 03 02 01 00 01 02 03 </normData>
<property id="11" source="raw" cat="descr" >
<name>bitDepth</name>
<valueSet id="i_i1_s13" >
<labValue>
<val>8</val>
<type>int</type>
</labValue>
<dataRef ind="normAll" />
</valueSet>
</property>
<property id="18" source="raw" cat="descr" >
<name>compression</name>
<valueSet id="i_i1_s16" >
<labValue>
<val>uncompressed</val>
<type>fixedLabel</type>
</labValue>
<dataRef ind="normAll" />
</valueSet>
</property>
<property id="2" source="raw" cat="descr" >
<name>imageHeight</name>
<valueSet id="i_i1_s10" >
<labValue>
<val>32</val>
<type>int</type>
</labValue>
<dataRef ind="normAll" />
</valueSet>
</property>
<property id="20" source="raw" cat="descr" >
<name>imageType</name>
<valueSet id="i_i1_s18" >
<labValue>
<val>blackIsZero</val>
<type>fixedLabel</type>
</labValue>
<dataRef ind="normAll" />
</valueSet>
</property>
<property id="22" source="raw" cat="descr" >
<name>resolutionUnit</name>
<valueSet id="i_i1_s33" >
<labValue>
<val>inch</val>
<type>fixedLabel</type>
</labValue>
<dataRef ind="normAll" />
</valueSet>
</property>
<property id="23" source="raw" cat="descr" >
<name>resolutionX</name>
<valueSet id="i_i1_s28" >

```
            <labValue>
               <val>72</val>
               <type>rational</type>
            </labValue>
            <dataRef ind="normAll" />
         </valueSet>
      </property>
      <property id="24" source="raw" cat="descr" >
         <name>resolutionY</name>
         <valueSet id="i_i1_s31" >
            <labValue>
               <val>72</val>
               <type>rational</type>
            </labValue>
            <dataRef ind="normAll" />
         </valueSet>
      </property>
      <property id="26" source="raw" cat="descr" >
         <name>samplesPerPixel</name>
         <valueSet id="i_i1_s21" >
            <labValue>
               <val>1</val>
               <type>int</type>
            </labValue>
            <dataRef ind="normAll" />
         </valueSet>
      </property>
      <property id="30" source="raw" cat="descr" >
         <name>imageWidth</name>
         <valueSet id="i_i1_s7" >
            <labValue>
               <val>32</val>
               <type>int</type>
            </labValue>
            <dataRef ind="normAll" />
         </valueSet>
      </property>
   </object>
</xcdl>
```