Document Description

Descriptor	Descriptions
Project Identifier	IST-2006-033789
Project Title	Planets
Sub-Project Title	Preservation Planning
Work Package Title	Preservation Policy and Strategy Models
Work Package Identifier	PP2
Deliverable Title	Report on the Conceptual Aspects of Preservation, Based on Policy and Strategy Models for Libraries, Archives and Data Centres
Deliverable Identifier	**PP2-D3**
Dissemination Level	PP
Deliverable Type	External Deliverable
Contractual Delivery Date	31 May 2009
Actual Delivery Date	9 July 2009
Author(s)	Angela Dappert

Abstract

Abstract

This last deliverable of work-package PP2 reports on the final version of a conceptual model for expressing the core concepts and requirements that appear in the digital preservation context. In addition, we present a machine-interpretable XML representation to support automated preservation planning tools.

Report PP2-D2 [PP2-D2 2008] discussed the methodology applied in deriving this conceptual model and presented an intermediate version. This iteration presents the final version after application in and validation against other Planets work-packages.

This conceptual model is suitable for

- modelling the full range of preservation services, such as monitoring, characterization, comparison of characteristics, and evaluation of candidate preservation actions,
- modelling technical as well as organizational properties,
- modelling the full range of preservation actions from emulation to migration and bit-preservation,
- modelling a full range of entities from logical to physical entities, including actions and environments.

Risks and requirements are first class objects within this model. The model lines up actions against the risks they mitigate.

Keyword List

Keywords	
Preservation policy	Property
Preservation strategy	Preservation object
Preservation planning	Preservation risk
Conceptual model	Preservation opportunity
Machine interpretable model	Preservation action
Characteristic	Preservation requirement

Contributors

Person	Role	Partner	Contribution
Angela Dappert	Work-package leader PP2	BL	Author
Bart Ballaux	PP2 partner	NANETH	Contributor
Peter Bright	Planets partner	BL	Developer (xsd)
Michaela Mayr	PP2 partner	ÖNB	Contributor
Sara van Bussel	PP2 partner	KB	Contributor

Document Approval

Person	Role	Partner
Adam Farquhar	Project Leader	BL
Hans Hofman	Sub-Project Leader	NANETH
Ross King	Scientific Board Member	ARC

Distribution

Person	Role	Partner

Revision History

Issue	Author	Date	Description
V 0.1	Angela Dappert	5 June 2009	1st draft
V 0.2	Angela Dappert	11 June 2009	2nd draft distributed to Hans Hofman, Ross King, Adam Farquhar
V 0.3	Angela Dappert	9 July 2009	3rd draft after feedback from Hans Hofman, Ross King, Adam Farquhar, Bart Ballaux

References

Ref.	Document	Date	Details and Version
[PP2-D2 2008]	Report on policy and strategy models for libraries, archives and data centres. Planets external report PP2-D2	June 2008	Angela Dappert, Bart Ballaux, Michaela Mayr, Sara van Bussel. 24 June 2008. http://www.planets-project.eu/docs/reports/Planets_PP2_D2_ReportOnPolicyAndStrategyModelsM24_Ext.pdf
[PP2-PresGuid 2008]	*Modelling Organisational Goals to Guide Preservation*	August 2008	Angela Dappert, Adam Farquhar. iPRES 2008: The Fifth International Conference on Preservation of Digital Objects, page 5. http://www.bl.uk/ipres2008/ipres2008-proceedings.pdf Appended as PP2-D3a
[PP2-SigChar 2009]	*Significance is in the Eye of the Stakeholder*.	June 2009	Angela Dappert, Adam Farquhar (2009). European Conference on Digital Libraries (ECDL) http://planets-project.eu/docs/papers/Dappert_Significant_Characteristics_ECDL2009.pdf Appended as PP2-D3b
[PP2-PresServ 2009]	*Implementing Metadata which Guides Digital Preservation Services.*	June 2009	Angela Dappert, Adam Farquhar. Appended as PP2-D3c

Table of Contents

Table of Tables

Table of Figures

Executive Summary

Previous preservation models have focused on preserving technical properties of digital files. Such an approach limits the choices of preservation actions and does not fully reflect preservation activities in practice.

Organizations consider properties that go beyond technical aspects and that encompass a wide range of factors that influence and guide preservation processes, including organizational, legal, and financial ones. Because of this, it is necessary to handle objects more general than just files, such as sets of files which create renditions of logical objects, and even abstract objects, such as intellectual entities and collections. In addition, we find that not only the digital objects' properties, but also the properties of the environments in which they exist, guide digital preservation processes.

Furthermore, organizations use risk-based analysis for their preservation strategies, policies and preservation planning. They combine information about risks with an understanding of actions that are expected to mitigate the risks. Risk and action specifications can be dependent on properties of the actions, as well as on properties of objects or environments which form the input and output of those actions.

The model presented in this report supports this view explicitly. It links risks with the actions that mitigate them and expresses them in stakeholder specific requirements. Risk, actions and requirements are first class objects in this model.

In addition, digital objects and environments are first class objects on an equal level. Models that don't have this property limit the choice of preservation actions to ones that transform a file in order to mitigate a risk. Establishing environments as first class objects enables us to treat risks to objects, environment, or even a combination of them.

1. Introduction

A set of successful end-to-end digital preservation services needs an underlying conceptual model and vocabulary on which it can build. A shared notion of the metadata that needs to be exchanged between these services is essential in order to correctly put together the pieces. Additionally this metadata must be able to capture the strategy and policy goals and constraints of the stakeholder who undertakes them.

The overall aim of the work-package is to produce such a conceptual model of organisational digital preservation policies and strategies. It should incorporate all relevant organisational characteristics and strategic directions, and cover the full life cycle of documents and records from the moment of creation. The original intention was to support automated digital preservation planning, but it evolved that a model suitable for this purpose was also suitable for supporting other preservation services, such as monitoring, characterization, comparison of characteristics, and execution and evaluation of preservation actions. The model was developed by analysing how stakeholders – implicitly or explicitly – define and materialize their commitment and effort to digital preservation.

Over-arching goals for developing such a conceptual model are

- To identify common features and systematic differences in the policies and strategies of different types of organisation (See a discussion of our results in report PP2-D2).
- To enable parts of the preservation planning process and decision support to be based on organisational policy and strategy requirements.
- To add to the scientific understanding of preservation.

The concrete deliverables are

- A conceptual model which can be reused by related work-packages.
- A specific vocabulary for the concepts in the model that stakeholders can use to model their own policies and strategies.[1]
- A machine interpretable model that can be used by preservation planning tools.

The conceptual model presented in this paper can

- support a full range of preservation services,
- model technical as well as organizational properties,
- model a full range of preservation actions from emulation to migration and bit-preservation,
- model a full range of entities from logical to physical entities, including actions and environments.

Risks and requirements are first class objects within this model. The model lines up preservation actions against the risks they mitigate. The resulting conceptual model presents a simple yet expressive representation of the digital preservation domain.

Planets report PP2-D2 [PP2-D2 2008] describes the analysis work that was performed to arrive at the conceptual model. It uses a worked example to give an overview of how the model and vocabulary in this report can be used. It covers a first draft of the conceptual model and data dictionary, lists the collected vocabulary for concepts, gives background information on our modelling approaches, and detailed reports on some interviews and a selection of resources. We will not repeat these discussions in this report. Please refer back to that document.

In this last iteration we validated the model through application in and alignment against other Planets work-packages and international preservation metadata models and frameworks to ensure its practical applicability for preservation services.

This report delivers

- a summary of our validation efforts
- an explanation of changes that resulted from this validation effort

[1]The specific vocabulary for file format properties, is being developed in Planets work-packages PC2, PC3, and TB3.

- an updated list of terminology
- a description of the updated conceptual model
- a data dictionary for this model
- a machine-interpretable XML representation

Additionally, work resulting from this work-package has been described in external publications (delivered, accepted or submitted). They should be considered part of the deliverable D3 for this work-package. Please see

- [PP2-PresGuid 2008] *Modelling Organisational Goals to Guide Preservation*.

This paper introduces the conceptual model and requirements categories found in preservation guiding documents, such as policy and strategy documents.

- [PP2-SigChar 2009] *Significance is in the Eye of the Stakeholder*.

This paper discusses a particular category of preservation guiding requirements which we encountered during this research: Significant Characteristics. The concept of significant characteristics has become increasingly prominent in the field of digital preservation [Hockx 2008]. As is often the case in an emerging field, however, the term has become over-loaded and remains ill-defined. In this paper, we unpack the meaning that lies behind the phrase, analyze the domain, and introduce clear terminology based on the PP2 conceptual model.

- [PP2-PresServ 2009] *Implementing Metadata which Guides Digital Preservation Services*.

Effective digital preservation depends on a set of preservation services which work together to ensure that digital objects can be preserved for the long-term. These services need digital preservation metadata, in particular, descriptions of the properties that digital objects may have and descriptions of the requirements that guide digital preservation services. This paper analyzes how these services interact and use this metadata. Based on this, the paper develops the part of the PP2 data dictionary presented in this report, which supports them.

2. Methodology

To perform the analysis, we initially used a combination of top-down and bottom-up methods. We examined the scientific literature to create a top-down model from first principles. To complement this, we analyzed actual preservation guiding documents, such as policy and strategy documents (called *PreservationGuidingRequirementsSets* in the model) for their content and interviewed decision makers to determine factors that influence their preservation choices. This work is reported in Planets report [PP2-D2 2008].

In this last iteration we refined the model through

- Continued expansion of the requirements base
- Continued clean-up of the conceptual model and data dictionary through co-ordination activities
- Design of a corresponding machine-interpretable model (XML schema)

In order to ensure its practical applicability for preservation services we validated the model's concepts and vocabulary through application in and alignment against other Planets work-packages that are dependent or have a close relationship with PP2. For example, we assisted in the use of the model in the preservation planning tool PLATO [Plato 2008], and continued our effort to integrate with the Core Planets model [Core 2008]. We also aligned our terminology with PREMIS [PREMIS 2008] which is the leading digital preservation data dictionary, and OAIS [OAIS 2002], the accepted framework for archival information systems.

In order to align the model with this other work, we made the changes to our model that are listed in the Appendix. Please also see the Appendix for a discussion of the evaluation of the alignment and validation effort. We have now arrived at a stable version of the model which is aligned with the investigated work. Further proof of concept will now require use of its features in implemented systems.

For a brief explanation of our modelling tools, UML, OCL, and XML, please refer to the Appendix 7.1. in report PP2-D2.

3. Terminology

Some key terms are defined here. Currently many of these terms are used inconsistently across various preservation research efforts.

The source of the definition is given in square brackets. Terms marked as [Planets] have been taken from the Planets Wiki in June 2009.

Many examples for and explanations of the terms are contained in the section on the conceptual model.

3.1 General Concepts

Term	Definition
Preservation	[ALA[2]] Digital preservation combines policies, strategies and actions that ensure access to digital content over time. For a more detailed definition please follow the link in the footnote. In the context of this paper we limit the scope to preservation aspects that maintain digital objects that are at preservation risk by mitigating those risks through preservation actions. Preservation requirements are used to determine the presence of those risks and to guide the choice of acceptable preservation actions.
Preservation Policy	[PP2 based on InterPARES2[3]] A formal statement of direction or guidance as to how an organization will carry out its preservation mandate, functions or activities, motivated by determined interests or programs.
Preservation Strategy	[Planets 2009-06-01] The strategy is a procedure of preservation actions to preserve a collection of digital objects. It treats only technical aspects. The preservation strategy thus contains a detailed description of the preservation action(s) to be taken, including ▪ used hardware and software, ▪ parameter settings for used tools and actions, and ▪ input and output file format, and ▪ available metadata about the action(s) In a preservation strategy, different tools and parameter settings can be defined for different file formats. Appropriate characterisation tools allow even different tools and parameter setting for the same file format with different characteristics.

[2]Association for Library Collections & Technical Services of the American Library Association, Definitions of Digital Preservation http://www.ala.org/ala/mgrps/divs/alcts/resources/preserv/defdigpres0408.pdf

[3] See the InterPARES2 glossary at: http://www.interpares.org/ip2/display_file.cfm?doc=ip2_glossary.pdf&CFID=243105&CFTOKEN=70677126, p. 20 (accessed: 23 May 2008). A similar definition can be found in R. Pearce-Moses, *A glossary of archival & records terminology*. Chicago, 2005, p. 300: “An official expression of principles that direct an organization’s operations.”

3.2 Concepts from the Conceptual Model

Bitstream	[PP2] A *Bitstream* is contiguous or non-contiguous data within one or more files that has meaningful common properties for preservation purposes.
Bytestream	[PP2] An ordered sequence of bytes. A file is a special *Bytestream*
Characteristic	[PP2] A *Characteristic* of an *Entity* is the concrete *Value* which this *Entity* has for an abstract *Property* in a defined context (a concrete *Property/Value* pair). In the model it is the *Characteristic* of a *PreservationObject, Environment* or *PreservationAction*.
Component	[PP2] A part of an *IntellectualEntity* for which *Values* for *Characteristics* can be determined. The largest possible *Component* is the whole of the *IntellectualEntity* itself.
Environment	[PP2] A set of factors which constrain a *PreservationObject* or *PreservationAction* and that are necessary to interpret it.
File	[PREMIS] A file is a named and ordered sequence of bytes that is known by an operating system. A file can be zero or more bytes and has a file format, access permissions, and file system characteristics such as size and last modification date.
IntellectualEntity	[PP2 combined with PREMIS] A set of content that is considered a single intellectual unit for purposes of management and description; a distinct intellectual or artistic creation.
PreservationAction	[PP2] The execution of an action that mitigates a *PreservationRisk* to the continued viability, renderability, understandability, and authenticity of a *PreservationObject* across time and changing *Environments*. It ensures the satisfaction of their *PreservationRequirements*, and transforms the *PreservationObject* itself, the *Environment* required to support access to the *PreservationObject*, or a combination thereof. A *PreservationAction* is an event resulting from the execution of a *PreservationService*.
PreservationGuiding RequirementsSet	[PP2] Representations that specify *Requirements*, which are logical constraints that make a stakeholder's values, goals or constraints explicit and influence a preservation process. They include oral representations, as well as written representations, in traditional documents, databases, source code, web sites, etc., such as policy, strategy, or business documents, as well as applicable legislation, guidelines, rules, or even a choice of temporary runtime parameters during a *PreservationAction*.
PreservationObject	[PP2] Any digital object that is directly or indirectly at risk and needs to be preserved.
PreservationRequirement	[PP2] A constraint which limits the space of allowable preservation activities.
PreservationRisk	[PP2] A *PreservationRisk* arises when a *Characteristic* of a *PreservationObject* or an *Environment* of a *PreservationObject* conflicts with the stakeholder's *RiskSpecifyingRequirements*.
PreservationService	[PP2] A core service supporting the goal of digital preservation. Examples are preservation monitoring, planning, characterization, comparison of characteristics, and execution and evaluation of *Preservation Actions*. Service is a subclass of *Environment*. Services are realized manually or through software tools and are associated with hardware and other *Environments*.
Property	[PP2] An abstract attribute, trait or peculiarity suitable for describing *PreservationObjects*, *PreservationActions* or *Environments*.

Representation	[PP2] The physical embodiment of a *Component*. A collection of all *Bitstreams* that are needed to create one rendition of a *Component* together with the necessary structural information.
RepresentationBitstream	[PP2] A *Bitstream* that is part of a *Representation*.
Significant Property / Significant Characteristic	[Wilson 2007] The characteristics of digital objects that must be preserved over time in order to ensure the continued accessibility, usability, and meaning of the objects, and their capacity to be accepted as evidence of what they purport to record. [PP2]*Requirements* in a specific context, represented as constraints, expressing a combination of *Characteristics* of *PreservationObjects* or *Environments* that must be preserved or attained in order to ensure the continued accessibility, usability, and meaning of preservation objects, and their capacity to be accepted as evidence of what they purport to record.
Value	[PP2] Every characteristic has a *Value* which can either be assigned or be inherent in the object. The *Value* can be looked up if it is stored explicitly or measured with an associated measuring tool, or deduced with a given logic if it is inherent in the object.
ValueOrigin	[PP2] The *ValueOrigin* concept provides a way to specify where a specific *Value* comes from or how it can be obtained. There can be multiple ways of obtaining the *Value* of a *Property* that do not conflict, measured by a different technique, using a different tool, or by a different agent.

Table 1 PP2 Terminology

4. A Conceptual Model, Specific Vocabulary and Data Dictionary

4.1 Introduction

This section proposes a conceptual model, vocabulary, and data dictionary for concepts captured in *PreservationGuidingRequirementsSet*s. It can be shared across all of the work-packages within preservation planning and other work packages within the Planets project, as well as outside of the project.

The core of a preservation planning model are the *Requirement*s which are expressed in *PreservationGuidingRequirementsSet*s, and all *PreservationObjects*, *PreservationActions* or *Environment*s and *Characteristics* to which those *Requirement*s refer. Besides these *Requirement*s, however, there are some general aspects which should be contained in *PreservationGuidingRequirementsSet*s. We borrow some basics from a model called StratML.

The PP2 model also draws from the OAIS model [OAIS 2002], PREMIS [PREMIS 2008], and from the Planets core conceptual model [Core 2008], which define key concepts related to digital objects.

It is important to note that the terminology regarding *Characteristics*, *Properties*, *Values*, etc. throughout the preservation literature is very inconsistent. The literature, for example, refers to significant properties just as it refers to essential characteristics. We have tried to ensure that the terminology in this report is internally consistent. Efforts have been made to unify the use with other work.

The description of the model and vocabulary are combined in this section. In each section, a new concept with its elements and relationships is introduced. This is supported by vocabulary for possible subclasses of the primary concepts introduced. It is assumed that an implementation knows for each instance to which subclass of the primary concept it belongs, so that preservation processes can be customized for particular subclasses. For example, if a *Requirement* is created as an instance of a *RiskSpecifyingRequirement* then the implementation knows that it is an instance of this subclass and can perform monitoring processes, rather than, say, preservation planning processes, guided by this *Requirement*.

4.2 Modelling the Context of Preservation Planning

Preservation planning is a process which identifies and mitigates risks to current and future access to digital objects. Preservation planning involves information about a stakeholder's policies and goals, its infrastructure, its user community, and the external environment in addition to information about the digital objects held within a collection.

Preservation planning goals are to

- Identify which parts of the collection present the greatest risks or the greatest opportunities for improvement.
- Identify candidate *PreservationActions* (alternatives) that could be taken to mitigate the risks.
- Evaluate the candidate *PreservationActions* to determine their potential costs and benefits.
- Weigh the cost/benefit of candidate *PreservationActions*. The cost may comprise the cost of executing the action, the cost of needed infrastructure for sustaining preservation output, the cost of essential *Characteristic*s lost in the *PreservationAction* (i.e. loss of authenticity) etc.. The benefit of the *PreservationAction* is the benefit of mitigating the risk in terms of the value of the object, the severity of the risk, etc.. Obviously these costs and benefits are not necessarily monetary.
- Provide justified recommendations for which actions to execute on which collections.

The result of the preservation planning process is a set of justified prioritised recommendations for actions that mitigate the risks presented to aspects of a collection. These recommendations can, in some cases, be executed automatically by a preservation plan execution engine.

An essential aspect of this preservation planning model is that it takes into account the goals and limitations of the stakeholder, features of its user community, and the environment in which its users access digital content. Thus, the scope of preservation planning extends beyond merely considering file formats and preserving characteristics of individual digital objects.

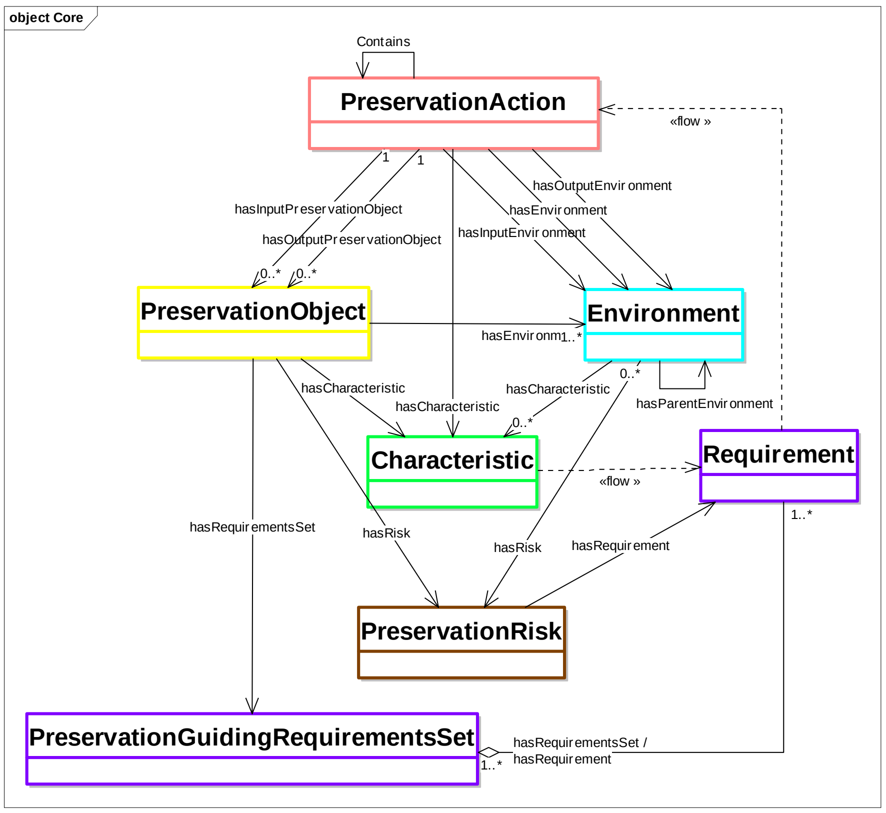

Figure 1: Preservation Planning Conceptual Model as UML Class Diagram

The key conceptual data model in the context of preservation planning is summarised in Figure 1. It shows the concepts and relationships which are explained in detail in the following sections.

In summary:

Any *PreservationObject*, such as a logical *IntellectualEntity* down to an individual *Bitstream,* has one or more *Environments*.

Every *Environment* in which the *PreservationObject* is embedded consists of a number of sub-*Environments*, such as hardware and software components, the legal system, and other internal and external factors.

Whenever changes occur to the *PreservationObject* or an *Environment*, such as obsolescence of hardware or software components, decay of data carriers, or changes to the legal framework, this may introduce a *PreservationRisk*.

PreservationRisks are specified in *Risk Specifying Requirements.* Whenever *Characteristics* of a *PreservationObject* or its *Environment* take on certain *Value*s which are specified in the *Requirement* then the *PreservationObject* is considered at risk.

Once a *Risk Specifying Requirement* is violated, a preservation monitoring process should trigger the preservation planning process. It, in turn, determines the optimal *PreservationAction* to mitigate this *PreservationRisk*. This preservation monitoring process is outside the scope of our model.

When a *PreservationAction* is applied to a *PreservationObject* and its *Environment,* then a new copy of the *PreservationObject* and/or a new *Environment* is created in which the *PreservationRisk* is mitigated. Every *PreservationAction*, therefore, does not only have an *Input PreservationObject* and an *Input Environment*, but also an *Output PreservationObject* and an *Output Environment*. For example, if a Microsoft Word *File* is migrated to an Adobe PDF *File*, not only do we create a new *PreservationObject*, which might have slightly new *Characteristics*, but we also need to embed it in a new *Environment* in which it can be used – in this case the platform needs to at least contain an Adobe PDF viewer. This approach works equally for migration, emulation, and hardware solutions.

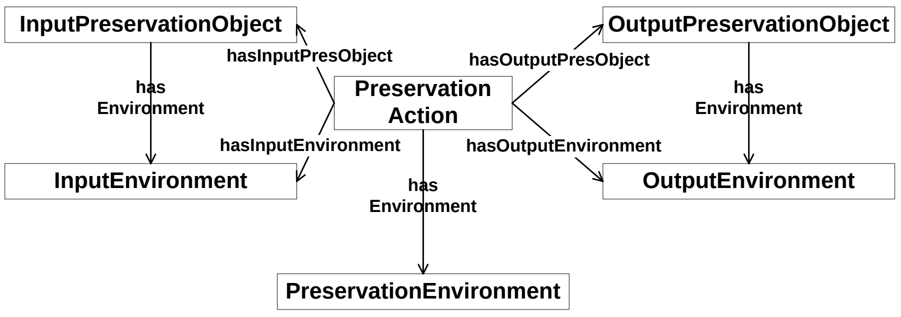

Figure 2: PreservationAction's Relationships

For any given *PreservationObject* and its *Environment,* there are multiple possible *PreservationActions* which might mitigate the *PreservationRisk*. Which of these *PreservationActions* is the most suitable for the *PreservationObject* can be derived from the information in *Requirements*. These *Requirements*

- define acceptable *Characteristics* of the *PreservationAction* itself (such as that PDF may, for a given stakeholder, not be an acceptable preservation output format of a *PreservationAction*)
- define acceptable output *Characteristics* of the *PreservationObject*, which may be dependent on input *Characteristics* (such as that the size of the *PreservationAction*'s output *PreservationObject* should not exceed a maximal size set by the stakeholder or should be in a specified relationship to that of the input *PreservationObject*).
- describe the preservation process itself independent of the *Characteristics* of the *PreservationObject* as well as of those of the *PreservationAction* (such as that a preservation planning process should be executed for every data object at least every 5 years, independent of the *PreservationRisks* that are established for this data object).

Events, Agents and *Rights* are concepts in the model and may be taken from PREMIS [PREMIS 2008].

4.3 Basic Concepts and Elements

The following concepts and elements are basic to the model.

- *Event*, *Agent*, and *Right* are concepts that are assumed in the model. They can be modelled in the way they are defined in PREMIS [PREMIS 2008], where *Events* and *Rights* describe *PreservationObjects* and *Agents* refer to either *Events* or *Rights*.
  A *PreservationAction*, is a special kind of *Event.*
- *Description* (optional, repeatable): a human readable meaningful description of a concept
  - *descriptionDefinition* (optional, repeatable): A verbal definition of the concept (data constraint: string)
  - *descriptionJustification* (optional, repeatable): Why this concept is needed for preservation planning (data constraint: string)
  - *descriptionExample* (optional, repeatable): Examples (data constraint: string)
  - *descriptionNote* (optional, repeatable): Notes (data constraint: string)
  - *descriptionUsage* (optional, repeatable): How this concept is to be used (data constraint: string)
- Version information which is used to manage the history of objects, is not included in this model. It is assumed that the system which implements this model will manage versions according to its own needs. Version information that is part of the name of the object (such as a software version or document version) are included.

4.4 PreservationGuidingRequirementsSet

Definition of PreservationGuidingRequirementsSet

Representations that specify *Requirements*, which are logical constraints that make a stakeholder's values, goals or constraints explicit and influence a preservation process. They include oral representations, as well as written representations, in traditional documents, databases, source code, web sites, etc., such as policy, strategy, or business documents, as well as applicable legislation, guidelines, rules, or even a choice of temporary runtime parameters during a *PreservationAction*.

Observations for PreservationGuidingRequirementsSet

Observation 1

The term goes beyond and refines the notion of "organisational policy and strategy" documents that were originally foreseen as basis for the analysis.

PreservationGuidingRequirementsSets are representations which

- may have any institutional scope (corporate, departmental, project related, etc.),
- may have any business focus (policy, strategy, mission, process, etc.),
- provide an input to the business process of preservation planning. Preservation plans are the output of a preservation planning process and are not considered *PreservationGuidingRequirementsSet*s.

Observation 2

The core of *PreservationGuidingRequirementsSet*s are the *Requirement*s which are expressed in them. Besides these *Requirement*s, however, there are some general aspects which should be contained in *PreservationGuidingRequirementsSet*s. We borrow some basics from a model called Strategy Markup Language (StratML). It is a basic conceptual model for describing the essential contents of a strategy document. For more information please see Appendix 6.4.

Elements of PreservationGuidingRequirementsSet

- *setIdentifier* (mandatory, non-repeatable): a unique identifier of the *PreservationGuidingRequirementsSet* (data constraint: *PreservationGuidingRequirementsSet ID*)
- *setName* (optional, repeatable): a human readable meaningful descriptor for the *PreservationGuidingRequirementsSet* (data constraint: string)
    - *setVersion* (optional, non-repeatable): Version of the *PreservationGuidingRequirementsSet* (data constraint: none)
- *StratML:Organization* (mandatory, non-repeatable): a unique identifier of the organization (data constraint: *Agent ID*)
- *setApproval* (optional, repeatable)
    - *status* (mandatory, non-repeatable): (data constraint: one of *proposed*, *approved*, *superseded, withdrawn*)
    - *initiator* (optional, repeatable): Person who proposed, approved or withdrew the *PreservationGuidingRequirementsSet*. (data constraint: *Agent ID*) (N.B. This subsumes the *StratML:submitter* element)
    - *statusDate* (mandatory, non-repeatable): Date on which the *PreservationGuidingRequirementsSet* was proposed, approved or withdrawn. (N.B. This subsumes the *StratML:Date* attribute)
- *setApplicability* (mandatory, non-repeatable)
    - *startDate* (optional, non-repeatable): The date the *PreservationGuidingRequirementsSet* is projected to become valid (data constraint: date)
    - *endDate* (optional, non-repeatable): The date the *PreservationGuidingRequirementsSet* is projected to cease, if it is not subsequently extended (data constraint: date)
- *StratML:Source* (optional, non-repeatable) The Web address (URL) for the authoritative source of this *PreservationGuidingRequirementsSet* (data constraint: anyURI)

- *StratML:Vision* (optional, repeatable): Vision statements are distinguished from goals in that they are the focus of constant pursuit but can never be satisfied in the sense of being met or completed. A concise and inspirational description of a state the organization will strive to approach over a relatively long span of years but which can ultimately never be fully achieved. (data constraint: string)
- *StratML:Mission* (optional, repeatable): Mission Statement. A brief description of the basic purpose of the organization. An agency's goals should flow from the mission statement. (data constraint: string)
- *StratML:Value* (optional, repeatable) A principle that is important and helps to define the essential character of the organization.
  - *StratML:Name* (optional)
  - *StratML:Description* (optional, repeatable)
- *Goal* (mandatory, repeatable)
  - *StratML:SequenceIndicator* (optional, non-repeatable)
  - *StratML:Name* (optional, non-repeatable)
  - *StratML:Description* (mandatory, non-repeatable) (data constraint: *Description*)
  - *StratML:Stakeholder* (optional, repeatable) (data constraint: *Agent ID*)
  - ***hasRequirement*** (optional, repeatable): a unique identifier of the *Requirements* included in this *RequirementsSet* (data constraint: *Requirement ID*)
  - *StratML:OtherInformation* (optional, non-repeatable)
- *references* (optional, repeatable)
  - *hasCollection* (optional, repeatable): unique identifiers for each of the organization's *collections* (ID of or specification of a set of *PreservationObjects*) to which the *PreservationGuidingRequirementsSet* refers (data constraint: ID of or specification of a set of *PreservationObjects*)
  - *hasRegistryReference* (optional, repeatable): unique identifiers for each of the registries and inventories to which the *PreservationGuidingRequirementsSet* refers (data constraint: *Registry ID*)
  - *hasPredecessor RequirementsSet* (optional, repeatable): unique identifiers for each of the predecessor *RequirementsSet*(s) of the *PreservationGuidingRequirementsSet* (data constraint: *PreservationGuidingRequirementsSet ID*)
  - *hasRelatedRequirementsSet* (optional, repeatable): unique identifiers for each of other related *RequirementsSets* (data constraint: *PreservationGuidingRequirementsSet ID*)

Other relationships of PreservationGuidingRequirementsSet

PreservationObject has a *hasRequirementsSet* relationship with *PreservationGuidingRequirementsSet.*

4.5 PreservationObject

Definition of PreservationObject

A *PreservationObject* is any digital object that is directly or indirectly at risk and needs to be digitally preserved.

Vocabulary for PreservationObject Subclasses

A *Bitstream* is the primary *PreservationObject.* If it is at risk, it becomes the object of preservation activity. We create and execute preservation plans to preserve it.

A *Bitstream* is, however, embedded in a larger context, as illustrated in Figure 3. Since higher-level objects, such as the *Representation* (one complete rendition of an *IntellectualEntity*) that includes the affected *Bitstream*, and the *IntellectualEntity* which is rendered by this *Representation*, are indirectly affected by its preservation need, they also need to be considered during preservation planning and are, therefore, indirectly *PreservationObjects*. Conversely, a stakeholder can not consider the preservation of each individual data object in isolation. Stakeholders need to take a global look at all their collections and resources in order to prioritise their *PreservationActions* and co-ordinate preservation activity. In order to facilitate this we are devising a model for *PreservationGuidingRequirementsSets* as a basis for preservation planning, which goes well beyond planning for the individual data object.

Vocabulary for *PreservationObject* Subclasses:

PreservationObject subclasses are *IntellectualEntity, Component, Representation, Bitstream.*

Figure 3 illustrates the 3-tiered *PreservationObject* hierarchy of our model.

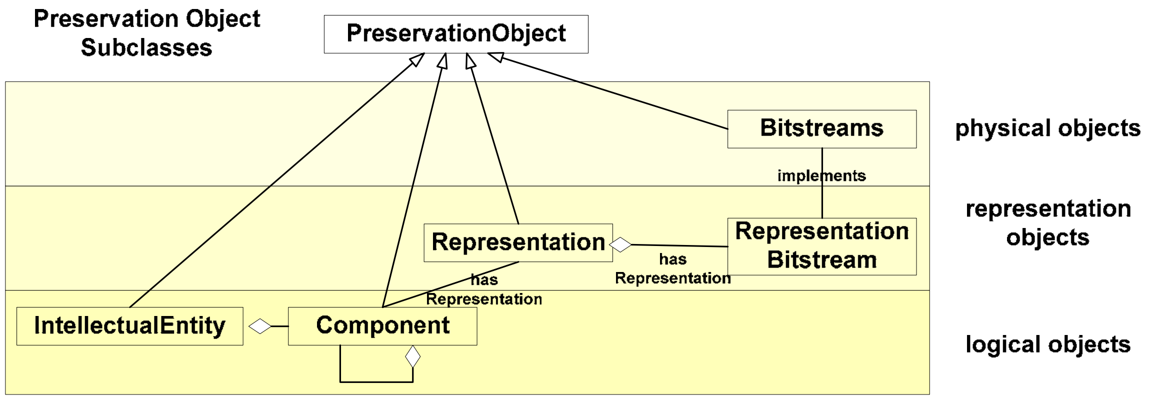

Figure 3: PreservationObject Subclasses

The *PreservationObject* concept corresponds to those objects directly or indirectly in need of preservation. It has subclasses on three tiers, as illustrated in Figure 3.

The top two tiers are associated with specific physical representations of digital objects. The top tier comprises physical objects, such as *Bitstreams* and its subclasses including *Bytestreams* and *Files*. The middle tier comprises *Representations* of logical objects consisting of *RepresentationBitstreams* that are needed to create a single rendition of a logical object (e.g., the set of *html* and *gif* files[4] needed to render the web version of a journal article). The bottom tier comprises logical objects such as *IntellectualEntities* and *Components*.

An *IntellectualEntity* is a distinct intellectual or artistic creation. PREMIS [PREMIS 2008] defines it as a set of content that is considered a single intellectual unit for purposes of management and description. The *IntellectualEntity* entity can be extended in ways to meet the needs of stakeholders. For example, in the library setting, common subclasses include *Collection* and *SubCollection* or *Work* and *Expression* to capture useful FRBR distinctions [FRBR 1998]. In an archival setting, subclasses such as *Fonds* and *Series* are relevant. Most repositories support discovery and delivery of *IntellectualEntities* such as *Books*, *Videos*, and *Articles*. But *IntellectualEntity* may also correspond to larger structures, such as *Collections* (=different levels

[4] The formal definition of such a statement would of course contain a persistent unique identifier of the exact version of the file formats. For improved readability of examples we casually refer to file formats by their file extension.

of aggregation), which may not be of interest to the end-user, but may be significant in preservation decisions.

During preservation, it is often necessary to consider fine-grained *Components* of an *IntellectualEntity*. Examples include *Table*, *Image*, *Title*, *Substring,* or even an individual *Character*. The *Component* entity can be decomposed in several ways, such as by the type of content (e.g., *TextComponent, ImageComponent, TableComponent)*, or by structure (e.g., *HeaderComponent* or *TableOfContentComponent). Values* for *Characteristics* of *Components* can be measured from their associated *Representations* (e.g. the *font* of a *CharacterComponent* can be extracted from its *RepresentationBitstream*.). In general, a *Representation* is made up of a set of *RepresentationBitstreams*.

Properties can be applicable to objects in every tier. For example:

- *fileSize* or *encoding* are applicable to *Files* (physical objects).
- *numberOfFilesInTheRepresentation, totalRepresentationSize, resolution, hasRepresentationBitstreamSequence*, or *preservationLevel* are applicable to *Representations* (representation objects).
- *pageCount* or *frameRate* are properties applicable to *IntellectualEntities* such as a *JournalArticle* or *Video*. *alignment* is a *Property* applicable to a *TextComponent*. *semanticInterpretation* can be a *Characteristic* of any *Component* (logical objects).

Example eJournal:

In this scenario the *Organization*, its *Collections*, their *JournalTitles*, their *Issues*, and their *Articles* can be types (i.e. subclasses) of *IntellectualEntities*. The primary logical object of preservation is an *IntellectualEntity* of type *Article*. The article can have several *Representations* that render it, such as an HTML *Representation* and a PDF *Representation.*

There are also smaller components of this *Article*, such as a *TextStringComponent* or a *TitleComponent*. They can have several *Representations* with possibly slightly different *Characteristics of their own*, such as their *fontSize Values*. Each *Representation* is captured in one or more *RepresentationBitstreams*.

Vocabulary for PreservationObject Subclasses

- Extensible vocabulary including *IntellectualEntity*, *Component*, *Representation* and *Bitstream*
- They can be further categorized as illustrated earlier in this section. An orthogonal categorization of *PreservationObjects* could be, for example, the intellectual content, the semantic and syntactic interpretation which are necessary to interpret the content, the format in which the content is encoded, or the physical realisation of the content.

Elements of PreservationObject and its Subclasses

- *preservationObjectIdentifier* (mandatory, non-repeatable): a unique identifier of the *PreservationObject* (data constraint: *PreservationObject ID*)
- *preservationObjectName* (optional, repeatable): a human readable meaningful descriptor for the *PreservationObject* (data constraint: string)
- *preservationObjectDescription* (optional, repeatable): a human readable meaningful description for the *PreservationObject* (data constraint: *Description*)
- *hasParent* (optional, repeatable): a unique identifier of the parent object (data constraint: *PreservationObject ID*)
- *hasEnvironment* (optional, repeatable): unique identifiers of each of the *PreservationObject*'s *Environment*s (data constraint: *Environment ID*)
- *hasCharacteristic* (optional, repeatable): unique identifiers of each of the *Characteristics* of the *PreservationObject* (data constraint: *Characteristic ID*). Every *PreservationObject* has none or more *Characteristics* with associated *Values* which may influence the choice of *PreservationAction*.
- *hasRisk* (optional, repeatable): unique identifiers of each of the *PreservationRisks* which have arisen as the *PreservationObject's Characteristics* violate a *Risk Specifying Requirement* (data constraint: *PreservationRisk ID*).
- *hasRequirementsSet* (optional, repeatable): unique identifiers of each of the *PreservationObject*'s *PreservationGuidingRequirementsSets* (data constraint: *PreservationGuidingRequirementsSet ID*)

- *hasStakeholder* (optional, repeatable): unique identifiers of each of the *PreservationObject*'s stakeholders (data constraint: *Agent ID*)
- *hasRight* (optional, repeatable): unique identifiers of each of the *PreservationObject*'s *Rights* objects (data constraint: *Rights ID*)
- *hasEvent* (optional, repeatable): unique identifiers of each of the *PreservationObject*'s *Event* objects (data constraint: *Event ID*) [5]

Other relationships of PreservationObject

- *PreservationAction* has a *hasInputPreservationObject* and a *hasOutputPreservationObject* relationship with *PreservationObject.*

4.5.1 IntellectualEntity

Definition of IntellectualEntity

A set of content that is considered a single intellectual unit for purposes of management and description; a distinct intellectual or artistic creation.

Vocabulary for IntellectualEntity Subclasses

Extensible vocabulary, such as: *Fonds, Series*, *Work*, *Expression; Collection, SubCollection, etc.*

Elements of IntellectualEntity and its Subclasses

- Elements inherited from *PreservationObject.*
  - The *hasParent* relationship is a reference to a parent *IntellectualEntity* or may be nil, for top-level *PreservationObjects*. (data constraint: *IntellectualEntity ID*)

4.5.2 Component

Definition of Component

A part of an *IntellectualEntity* for which *Values* for *Characteristics* can be determined.

The largest possible *Component* is the whole of the *IntellectualEntity* itself.

Example:

A *TextStringComponent*, *FootnoteComponent* or *AbstractComponent* in a *JournalArticle*.

Vocabulary for Component Subclasses

Extensible vocabulary for *Component* subclasses (such as *Header*, *Body*, *Footer* / *Title*, *Abstract*, *Appendix* / *SubString*, *Table*) is being developed in preservation characterization research. For text-based systems the vocabulary to specify the *Component* subclasses can, for example, be taken from the NLM DTD [NLM] or TEI [TEI] which uses tags for mark-up of text components. Other *Component* subclasses can be defined for other content-type specific needs, such as sound, video, etc..

The *Component* entity can be decomposed in several ways, such as

- by the type of content (e.g., *TextComponent, ImageComponent, TableComponent)*, or
- by document structure (e.g., *HeaderComponent* or *TableOfContentComponent).*

An example is shown in Figure 4.

[5] For all events the following holds: Whether recording a certain event is mandatory, and which event to record is a business requirement of the institution. It is not made mandatory by the data model.

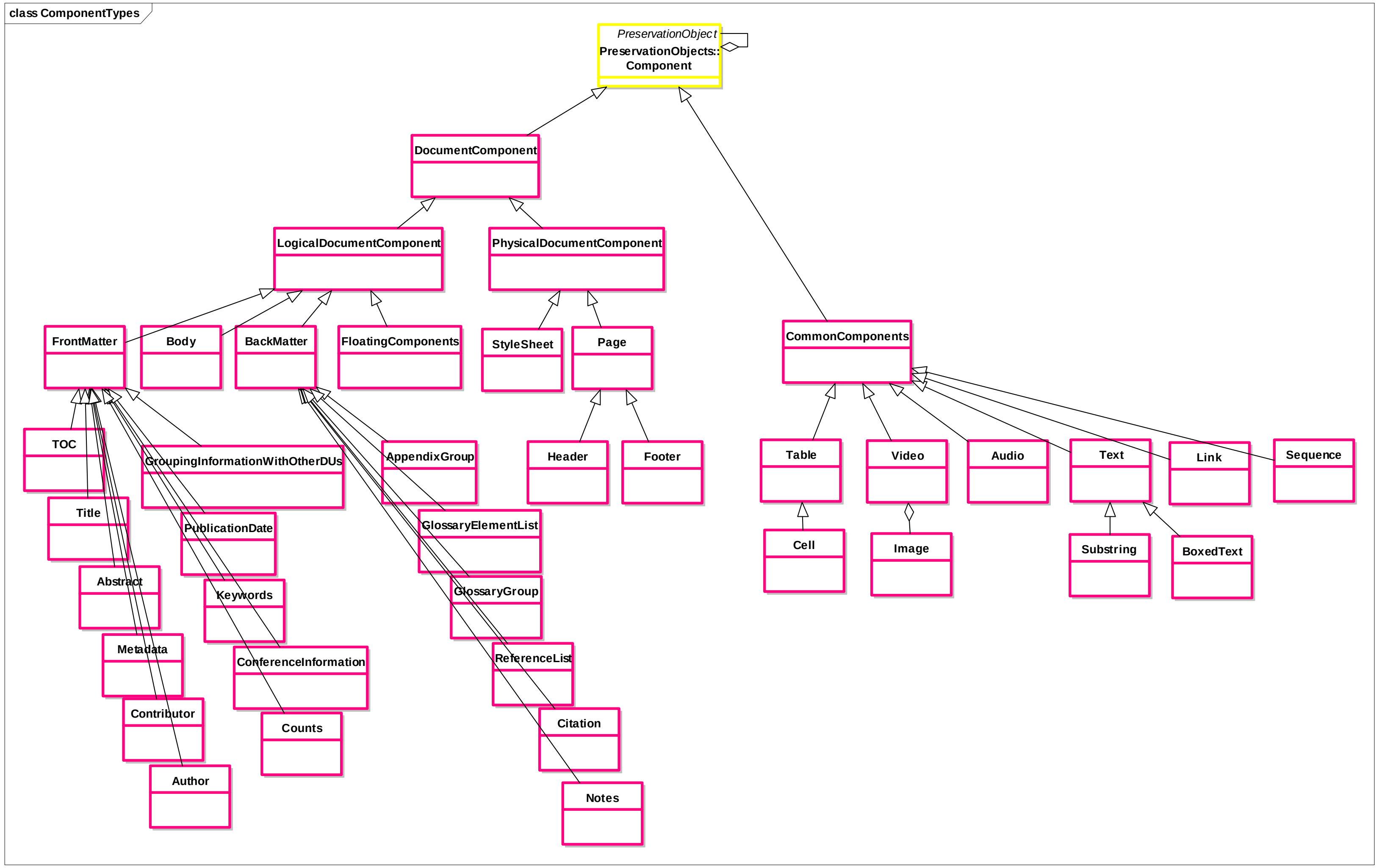

Figure 4: Example of Component Subclasses for Text Applications

Values for *Characteristics* of *Components* can be measured from their associated *Representations* (e.g. the *font* of a *CharacterComponent* can be extracted from its *RepresentationBitstream*.).

Elements of Component

- Elements inherited from *PreservationObject.*
    - The *hasParent* relationship is a reference to a parent *IntellectualEntity* or *component*. (data constraint: *IntellectualEntity* or *Component ID*)
    - One *hasEvent* relationship is a reference to a *ComponentDiscoveryEvent*.
- *hasRepresentation* (optional, repeatable): unique identifier of the *Component's Representation*.

Other Relationships with Component and its Subclasses

- *Component* is a subclass to *PreservationObject*.
- *Representation* has a *rendersComponent* relationship with *Component.*

4.5.3 Representation

Definition of Representation

The physical embodiment of a *Component*.

A collection of all *Bitstreams* that are needed to create one rendition of a *Component* together with the necessary structural information.

Observations for Representation

Observation 1

Components may have multiple *Representations*. For example a journal article may come both in .doc format and an .XML document with associated files. Any set of files that allows authentic rendering of the *Component* within its technical *Environment* is a *Representation* of the *Component*.

Elements of Representation

- Elements inherited from *PreservationObject*
- *rendersComponent* (mandatory, repeatable): unique identifier of the *Component* for which the *Representation* serves as physical embodiment. (data constraint: *Component ID*)
- *hasRepresentationBitstream* (mandatory, repeatable): unique identifier of the *Bitstreams* that make up the *Representation*. (data constraint: *RepresentationBitstream ID*)
- *hasRepresentationBitstreamStructmap* (mandatory, repeatable): information to capture the physical and logical structural relationships of the *RepresentationBitstreams* that make up the *Representation*. See the METS structMap definition for comparison. [METS].

Other Relationships with Representation

- *Representation* is a subclass to *PreservationObject*.
- *Component* has a *hasRepresentation* relationship with *Representation*.
- *RepresentationBitstream* has a *hasRepresentation* relationship with *Representation*

4.5.4 Bitstream

Definition of Bitstream

A *Bitstream* is contiguous or non-contiguous data within one or more files that has meaningful common properties for preservation purposes.
It can be a digital *File* or embedded within a digital *File*.

A non-file *Bitstream* can be transformed into a standalone file with the addition of file structure (headers, etc.) and/or reformatting the *Bitstream* to comply with some particular file format, by giving it the required metadata (name, create date, ...), a path, and placing it into a file system.

A *File* is a named and ordered sequence of bytes that is known by an operating system. A *File* can be zero or more bytes and has a file format, access permissions, and file system characteristics such as size and last modification date [PREMIS 2008].

Vocabulary for Bitstream Subclasses

- *Bytestream* is a *subclass* of *Bitstream*.
- *File* is a *subclass* of *Bytestream.*

Extensible vocabulary. An example is shown in Figure 5**Error! Reference source not found.**.

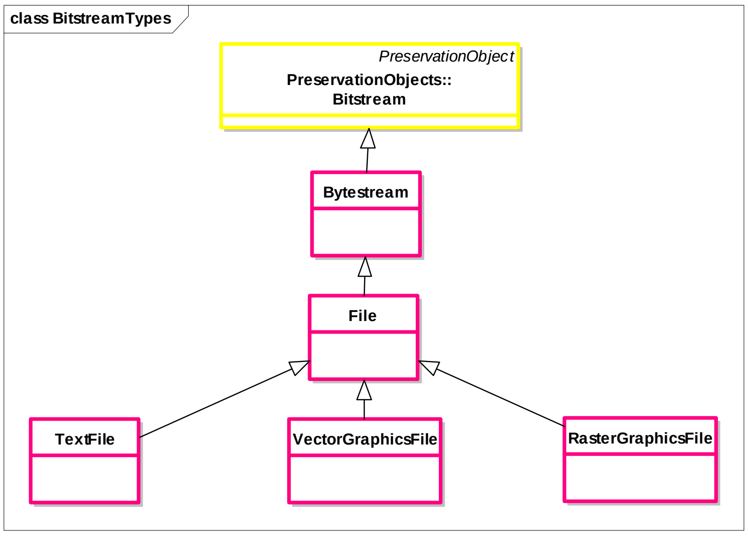

Figure 5: Example of Some Bitstream Subclasses

Elements of Bitstream

- Elements inherited from *PreservationObject*
- *implements* (optional, repeatable): unique identifier of the *RepresentationBitstreams* which are realized by the *Bitstream* (data constraint: *RepresentationBitstream ID*). A link between the *RepresentationBitstream* and *Bitstream* is mandatory in at least one direction.

Other Relationships with Bitstream

- *Bitstream* is a subclass to *PreservationObject*.

4.5.5 RepresentationBitstream

Definition of RepresentationBitstream

A Bitstream that is part of a *Representation*.

We use sets of *RepresentationBitstreams* rather than *Bytestreams* or *Files* to capture a *Representation*, since, the bits representing *Characteristics* of *Components* of *IntellectualEntities* may not necessarily align with byte boundaries (e.g. when they are extracted from a compressed file directly or if *Characteristics* are represented as bitmaps). They may span several files (e.g. large files may be split with a Unix "split" command, data may be streamed into containers of a fixed file-size, such as ARC, data may be split over several files to optimize access).

Elements of RepresentationBitstream

- *hasRepresentation* (mandatory, repeatable): unique identifier of each of the *RepresentationBitstream*'s *Representations* (data constraint: *Representation ID*)
- *implementedBy* (optional, non-repeatable): unique identifier of the physical object that implements the *Bitstream* including offset information, etc. (data constraint: *Bitstream ID*) A link between the *RepresentationBitstream* and *Bitstream* is mandatory in at least one direction.

Other Relationships with RepresentationBitstream

- *Representation* has a *hasRepresentationBitstream* relationship with *RepresentationBitstream*

- *Bitstream* has a *implements* relationship with *RepresentationBitstream.*

4.6 Environment

Definition of Environment

A set of factors which constrain a *PreservationObject* or *PreservationAction* and that are necessary to interpret it.

Observations for Environment

Observation 1

Every *PreservationObject* has one or more *Environments* which may be fulfilling different purposes. For example, a *File* or a *Representation* object may have creation, ingest, preservation, and access *Environments*; a *Collection* may have an internal, a physical delivery, and an online delivery *Environment*.

Observation 2

The selection of a *PreservationAction* may depend on the *Characteristics* of these *Environments* and the *Characteristics* which the output *Environment* would have if the given candidate *PreservationAction* was to be executed.

Observation 3

Environments for *PreservationObject*s at a higher level (logical or representation, resp.) also apply to *PreservationObjects* at a lower level (representation or physical, resp.). But lower level *PreservationObjects* may have additional *Environments* or *Characteristics*.

Therefore, the *Environment* for a *File*, for example, can be different from the *Environment* of the *Representation* to which it belongs. As long as the *File* is part of its *Representation*, it will live in the *Representation's Environment*. When it is taken out of the *Representation*'s *Environment*, for example to be used in a migration, then the *File*'s individual *Environment* will influence the *Environment* of its new *Representation*. For example, a website may only render properly in IE6.0, but a jpg image contained within it would render in a simple viewing environment.

Observation 4

It may not be possible to derive the best *Environment* from a *File*’s file format alone. If, for example, a *File* does not make use of the full range of features of the file format then it may be supported by an *Environment*, which in general might not support all *Files* of its file format. Stakeholders may wish to specify the *Environment* together with their intentions (necessary, recommended, acceptable…).

Observation 5

PreservationService is a subclass of *Environment*. It is any core service supporting the goal of digital preservation. Examples are preservation monitoring, planning, characterization, comparison of characteristics, and execution and evaluation of *Preservation Actions*. Services are realized manually or through software tools and are associated with hardware and other *Environments*.

The goal of preservation planning is to determine the preservation (execution) service which will produce the optimal *PreservationAction*.

Vocabulary for Environment Subclasses

Every *Environment* can consists of a number of sub-*Environments*, such as data, software, hardware, and community or other internal and external *Environment*al factors, such as legal or budget restrictions.

The top-level vocabulary to specify the *Environment* subclasses can be taken from Figure 6. Lower-level vocabulary is specified in Figure 7, Figure 8, and Figure 9. They can be extended according to institution-type specific needs.

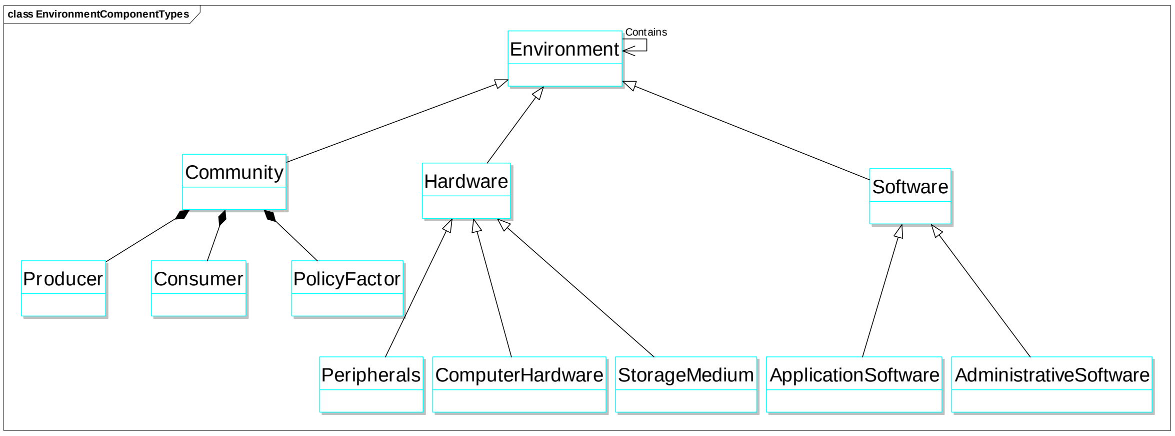

Figure 6: Top-level Vocabulary for Environment Subclasses

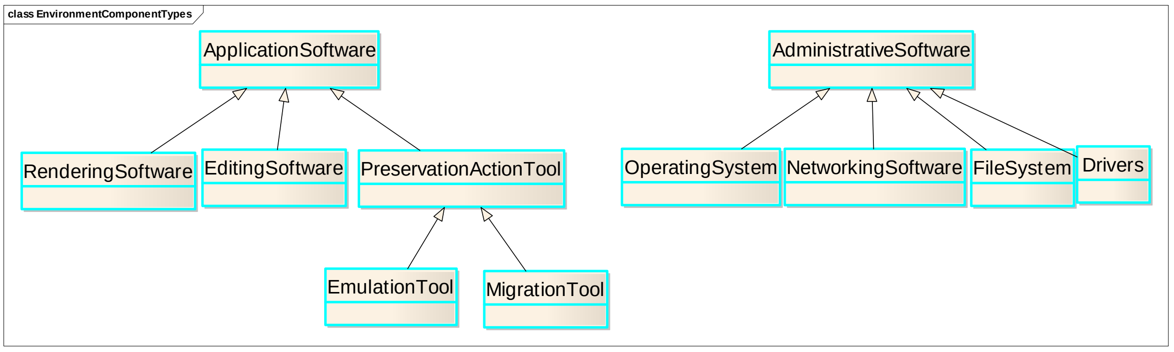

Figure 7: Vocabulary for Software

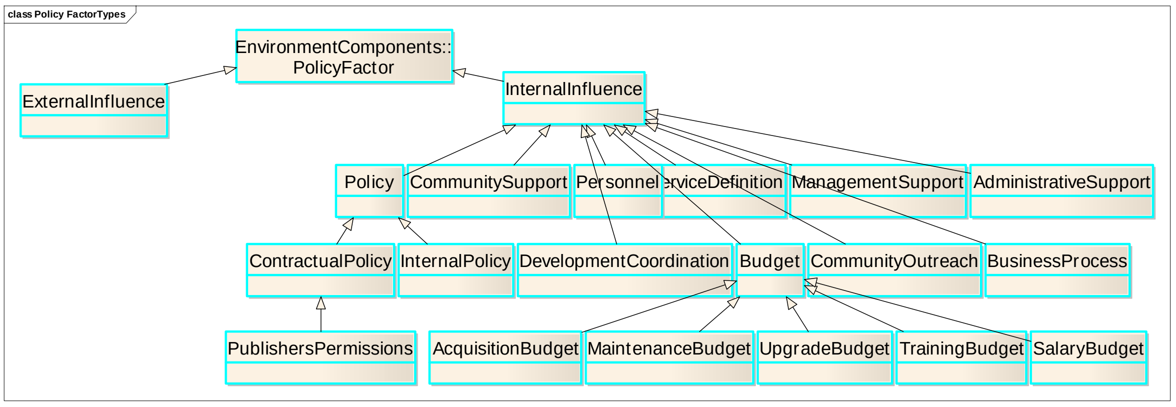

Figure 8: Vocabulary for Internal Influences

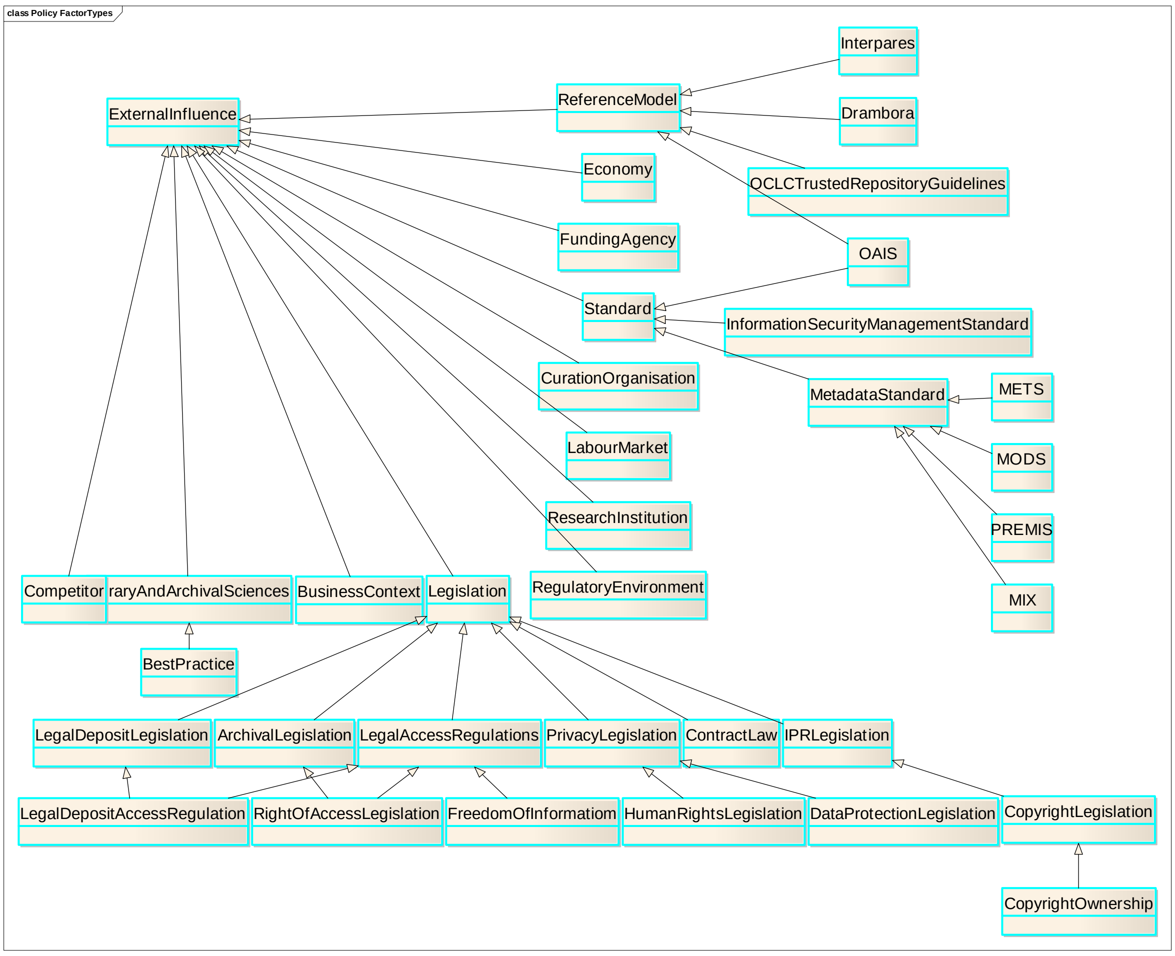

Figure 9:Vocabulary for External Influences

Elements of Environment and its Subclasses

- *environmentIdentifier* (mandatory, non-repeatable): a unique identifier of the *Environment* (data constraint: *Environment ID*)
- *environmentName* (optional, repeatable): a human readable meaningful descriptor for the *Environment* (data constraint: string)
- *environmentDescription* (optional, repeatable): a human readable meaningful description for the *Environment* (data constraint: *Description*)
- *environmentPurpose* (optional, repeatable): (data constraint: extensible vocabulary: one of *creation*, *ingest*, *preservation*, *remote access*, *local access*, *migration*,…)
- *environmentFunction* (optional, repeatable): (data constraint: extensible vocabulary: one of *rendering, editing, executing, printing….)*
- *environmentIntention* (optional, repeatable): (data constraint: extensible vocabulary: one of *necessary*, *recommended*, *acceptable*…)
- *hasParentEnvironment* (optional, repeatable): unique identifiers to each of the parent *Environment* objects (data constraint: *Environment ID*)
- *hasPreservationObject* (optional, repeatable): a unique identifier of the *PreservationObjects* to which the *Environment* belongs; (data constraint: *PreservationObject ID*)
  (Inverse of the *hasEnvironment* relationship from *PreservationObject* to *Environment*).
- *hasCharacteristic* (optional, repeatable): unique identifier of each of the *Characteristics* of the *Environment* (data constraint: *Characteristic ID*). Every *Environment* has none or more *Characteristics* with associated *Values* which may influence the choice of *PreservationAction*.

- *hasRisk* (optional, repeatable): unique identifiers of each of the *PreservationRisks* which have arisen as the *Environment's Characteristics* violate a *RiskSpecifyingRequirement* (data constraint: *PreservationRisk ID*).
- *hasEvent* (optional, repeatable): unique identifiers to each of the *Environment's Event* objects (data constraint: *Event ID*)

Other Relationships with Environment

- *PreservationObject* has a *hasEnvironment* relationship with *Environment*
- *PreservationAction* has a *hasEnvironment*, a *hasInputEnvironment* and a *hasOutputEnvironment* relationship with *Environment.*

4.7 PreservationRisk

Definition of PreservationRisk

A *PreservationRisk* arises when a *Characteristic* of a *PreservationObject* or an *Environment* of a *PreservationObject* conflicts with the stakeholder's *RiskSpecifyingRequirement*s.

Observations for PreservationRisk

Observation 1

Preservation planning is about mitigating *PreservationRisk* to access of digital objects or about taking advantage of opportunities for improvement through *PreservationAction*s.

Observation 2

Specific *PreservationRisks* are associated with a *PreservationObject* or a specific *Environment* of a *PreservationObject*.

Examples of *PreservationRisk* include:

- Data carriers deteriorate and cannot be read.
- The data object becomes corrupted on the carrier and the original byte stream cannot be retrieved.
- Essential hardware components are no longer supported or available.
- Software components are proprietary and this dependence is unacceptable to the stakeholder.
- The community requires new patterns of access, such as access on a mobile phone, rather than a workstation.
- File formats become obsolete.
- The legislative framework changes and the data or access to it has to be adapted to the new regulations.

Examples of *PreservationOpportunities* include:

- Adding features, such as interactivity, provides new usage opportunities.
- Maintaining data becomes cheaper by moving to alternative formats.
- Consolidate support structures (e.g. software or hardware *Environment*s) streamlines the maintenance of the Collection.

In the remainder of this paper when we talk about *PreservationRisks* we implicitly include *PreservationOpportunities*.

Observation 3

These risks are not always inherent, but are relative to considerations such as the stakeholder's goals and the *Characteristics* of individual *PreservationObjects*.

Examples:

- Depending on the stakeholder's goals: One stakeholder might find using proprietary software acceptable, another might not, and, therefore, does or does not consider it a *PreservationRisk*
- Depending on the digital object's individual *Characteristic*s: The digital object uses, or does not use macros and, therefore, is or is not subject to a *PreservationRisk*.

Each stakeholder must therefore specify in *RiskSpecifyingRequirements* which state of the *PreservationObject* or the *PreservationObject's Environment* represents a *PreservationRisk*.

Observation 4

Risks obviously apply to technological *Environments*. But they also apply to community *Environments*. If, for example, consumers request changed services (i.e. considers existing services obsolete) then this may prompt the need for executing a *PreservationAction* which brings the services up to date.

Vocabulary for PreservationRisk Subclasses

Extensible vocabulary. Basic *PreservationRisk* subclasses are (see Figure 10**Error! Reference source not found.**):

NewVersionRisk: A new version of the *PreservationObject* or *Environment* is available. This creates a risk of future obsolescence, or a risk of having to support too many versions.

LackingSupportRisk: The *PreservationObject* or *Environment* is no longer sufficiently supported. This creates a risk that support will cease altogether, rendering the *PreservationObject* or *Environment* non-functional.

DeteriorationOrLossRisk: The *PreservationObject* or *Environment* is deteriorating or has been lost. Reconstruction or replacement become necessary.

ProprietaryRisk: The *PreservationObject* or *Environment* is proprietary. There is a risk that it cannot be replaced since the specifications for it are unknown.

UnmanagedGrowthRisk: The stakeholder's *PreservationObjects* or *Environments* are becoming too diverse to manage. A normalisation *PreservationAction* is needed to simplify or unify them.

Alternatively or additionally, these risk categories can be used to create sub-categories of *RiskSpecifyingRequirements.*

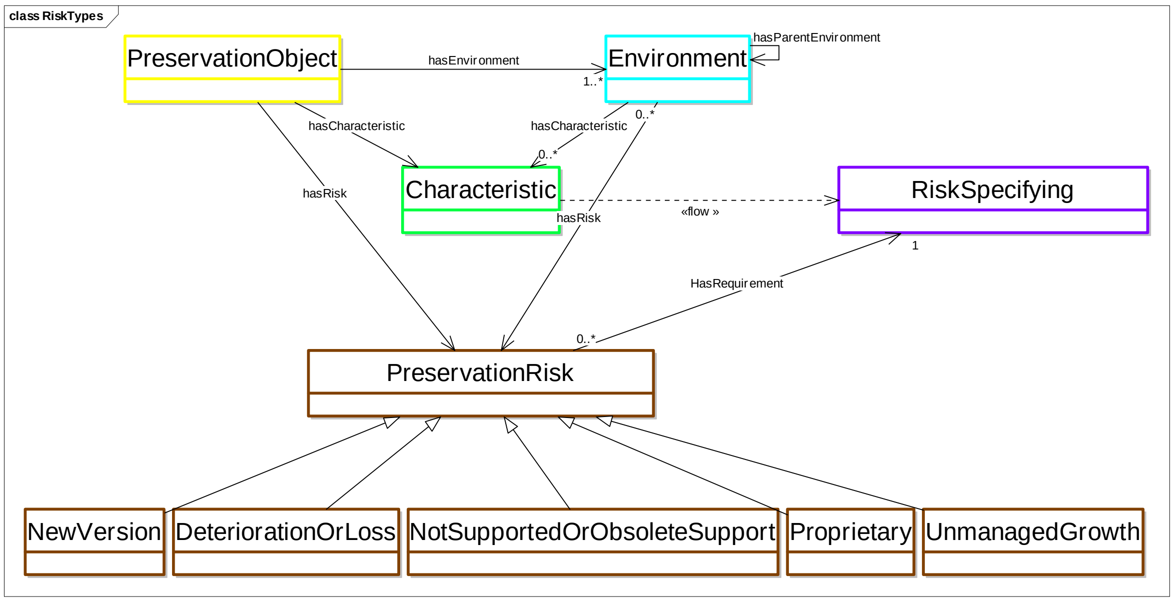

Figure 10: Vocabulary for PreservationRisk Subclasses

Elements of PreservationRisk

- *riskIdentifier* (mandatory, non-repeatable): a unique identifier of the *PreservationRisk* (data constraint: *PreservationRisk ID*)
- *riskName* (optional, repeatable): a human readable meaningful descriptor for the *PreservationRisk* (data constraint: string)
- *riskDescription* (optional, repeatable): a human readable meaningful description for the *PreservationRisk* (data constraint: *Description*)
- *associatedWith* (mandatory, non-repeatable):
  vector of unique identifiers of *PreservationObject* or *Environment Instances* that are at risk
  (data constraint: vector of *PreservationObject* or *Environment IDs*). A *PreservationRisk* may consist of the interplay of 2 or more *PreservationObjects* or *Environments*. Therefore there is a need to express a vector of affected *Entities*.
  (Inverse of the *hasRisk* relationship from *PreservationObject* or *Environment* to *PreservationRisk*).
- *hasRequirement* (mandatory, non-repeatable): a unique identifier of the *RiskSpecifyingRequirement* which is violated by the *PreservationRisk* (data constraint: *RiskSpecifyingRequirement ID*).

- *hasEvent* (optional, repeatable): unique identifiers to each of the *PreservationRisk's Event* objects (data constraint: *Event ID*) This might have specific information about which *Characteristics* of which *PreservationObject* or *Environment* violated the *RiskSpecifyingRequirement* at the time when the *PreservationRisk* was discover

Other Relationships with PreservationRisk

- *Environment* and *PreservationObject* have a *hasRisk* association link to *PreservationRisk*.

4.8 PreservationAction

PreservationActions are included in the model since many *Requirements* in *PreservationGuidingRequirementsSets* refer to desired *Characteristics* of permissible *PreservationActions*.

Definition of PreservationAction

In the Planets glossary, *PreservationAction* is currently defined in the following way:

> A non-destructive action that creates new data from existing data in the archive, with the intent of preserving or increasing access to information stored in the archive

The following is an older Planets definition:

> The execution of an action to ensure the continued accessibility of a digital object across time and changing technical *Environment*s and the preservation of its critical significant properties that transforms the digital object itself, the technical *Environment* required to support access to the object, or a combination thereof.

The newer definition shows a shift of focus within Planets toward *PreservationAction*s on data related actions rather than hardware related actions or emulation. The PP2 model uses the more general older approach (which encompasses the newer Planets definition) slightly adapted to our model.

> The execution of an action that mitigates a *PreservationRisk* to the continued viability, renderability, understandability, and authenticity of a *PreservationObject* across time and changing *Environments*. It ensures the satisfaction of their *PreservationRequirements*, and transforms the *PreservationObject* itself, the *Environment* required to support access to the *PreservationObject*, or a combination thereof.
>
> A *PreservationAction* is an event resulting from the execution of a *PreservationService*.

Observations for PreservationAction

Observation 1

A *PreservationAction* produces a changed version of the *PreservationObject* and/or its *Environment*. The model, therefore, contains an input and output *PreservationObject* and input and output *Environments* for a *PreservationAction* (see Figure 2).

Examples:

- In the case where a corrupted file is recovered from a back-up, there is an input and output *File* while the *Environment* may stay the same.
- In the case of migration, there is an input and output *Representation*. The input and output *Representations* may need different *Environments*.
- In the case of data carrier refresh, the input and output *Files* are the same, but the *Environment* is new.

A *PreservationAction* produces a new *PreservationObject*, if the intellectual content of the *PreservationObject*, the semantic and syntactic interpretation of the content which are necessary to interpret the content, the format in which the content is encoded, or the physical realisation of the content change.

Example: In the case of file reconstruction there is an input and output *File* since the realisation of the *File* changes. If the *File* is part of a *Representation* , then there will also be a new output *Representation* object, or possibly even a new *IntellectualEntity* if *Characteristic*s change sufficiently.

Observation 2

In general a *PreservationAction* may result in the replacement or repair or reconstruction of a combination of *Environments*.

Example: Emulation can be seen as a combination of hardware, software and file format replacement, since it provides a new hardware and/or software *Environment* for the digital object, but it might also be necessary to extract data from the original digital object to feed into the emulation.

Observation 3

A *PreservationAction* always applies to one input and output *PreservationObject*. This *PreservationObject*, however, may consist of several *Components*.

Example:

- Several input *Components*: When migrating an XML *Representation* to a PDF *Representation*, the input *Representation* consists of the XML file and its images. Migrating an Oracle database to an Access database, consumes .dbf, .ctl files, etc. and produces one .mdb file.
- Several output components: When migrating a Word *Representation* to an HTML *Representation*, the output *Representation* consists of the XML file with an accompanying CSS file. Migrating a .zip file to its expanded version leads to multiple formats.

Observation 4

The *PreservationAction* has an *Environment* of its own. The *PreservationService* (e.g. a certain configuration of a migration tool), for example, is one of them.

These *Environments,* and the *PreservationAction,* have *Characteristics* of their own, such as *acceptedInputFormat*, *outputFormats*, *preservationActionCost*. They are used to guide preservation planning through *ActionDefiningRequirements*, which define which kinds of *PreservationActions* are desirable, or they are used to express *PreservationGuidingRequirements* which are conditional on *Characteristics* of *PreservationActions*.

Observation 5

One can extend the scope of the model to preservation activities other than preservation planning. These activities can be described in the same way as here, and have *Requirements* attached to them.

Vocabulary for PreservationAction Subclasses

Extensible vocabulary.

A simple categorization is the following: A *PreservationAction* may result in the *Replacement*, *Repair* or *Reconstruction* of any of the *PreservationObjects* or *Environments* that are at risk. This is illustrated in Figure 11.

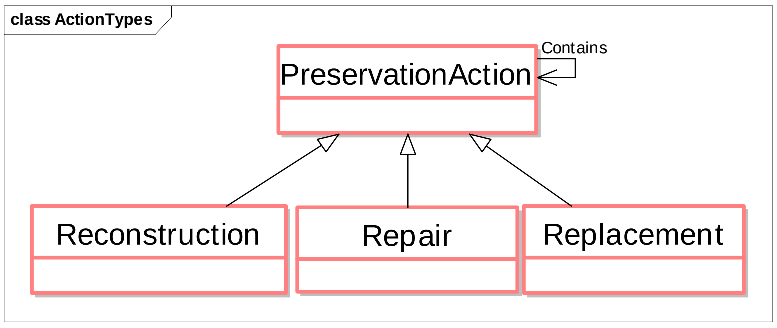

Figure 11: Vocabulary for PreservationAction Subclasses

This specification can, for example, be further refined depending on the combination of

- *PreservationObject* subclass and/or *Environment* subclass,
- *PreservationRisk* subclass
- simple *PreservationAction* subclass.

Table 2 shows some examples of refined *PreservationAction* subclasses. Such *PreservationAction* subclasses may suitably be described in a registry.

Example	*PreservationObject* subclass	*Environment* subclass	*PreservationRisk* subclass (new version, not supported / obsolete, deterioration / loss, proprietary)	Simple *PreservationAction* subclass (reconstruction, repair, replacement)
Data carriers deteriorate and cannot be read		Data Carrier	Deterioration	Replacement
The data object becomes corrupted on the carrier and the original bitstream cannot be retrieved.	*Bitstream*		Deterioration	Reconstruction
Essential hardware components are no longer supported or available		Hardware	Not supported	Replacement
Software components are proprietary and the dependence is unacceptable to the stakeholder.		Software	Proprietary	Replacement
The community requires new patterns of access, such as access on a mobile phone, rather than a workstation		Hardware and Software	Obsolete	Replacement
File formats become obsolete.	*File*		Obsolete	Replacement
The legislative framework changes and the data or access to it has to be adapted to the new regulations		Legislation	New Version	Replacement

Table 3 Examples of refined PreservationAction Subclasses

In other words, corresponding to every *PreservationRisk* and the subclass of the affected *PreservationObject* or *Environment* that needs to be addressed, there are appropriate *PreservationActions* to mitigate the risk.

Examples: The risk of data carrier failure can be mitigated by a carrier refresh. The risk of file format obsolescence can be mitigated by migrating objects to an alternative format.

Figure 12 shows some examples of *PreservationAction* subclasses depending on the subclasses of the *PreservationRisk* and the affected *PreservationObject* or *Environment*.

Most of them are self-explanatory. Some deserve some special comments:

- Modification of *Content* might represent an action such as the reconstruction of a deteriorated file, or a file that is modified in order to satisfy new legal requirements.
- One possible *PreservationAction* is to not do anything (wait and see).
- Migration does not always imply that a different file format is chosen. One might, for example replace an XML file with another XML file. In that case the input and output file formats happen to

be the same. The output *PreservationObject* might nonetheless have different *Characteristics* to the input *PreservationObject* because of the different information captured within the XML tags.

- The needs of the target community might be a deciding factor for the choice of *PreservationActions*, and, conversely, the choice of *PreservationActions* will shape and change the community, just as it changes the other *Environment* subclasses.
- Community consists of producers and consumers. Both types are either technical (e.g. repository or IT staff, publishing staff) or content oriented (authors or readers) and will consider the digital object obsolete under different circumstances and according to their needs.
- Shifting the target community might be a somewhat unintuitive *PreservationAction*, which is parallel to all other forms of *Environment* replacement. An example might be turning a research data centre into a history-of-science repository, as the material contained in the collection seizes to live up to contemporary standards of scientific use.

Elements of PreservationAction

PreservationAction is an *Event* and has general *Event* information (see PREMIS), such as start time / end time, agent, and outcome. It has additional elements.

- *actionIdentifier* (mandatory, non-repeatable): a unique identifier of the concrete *PreservationAction* (data constraint: *PreservationAction ID*)
- *actionName* (optional, repeatable): a human readable meaningful descriptor for the *PreservationAction* (data constraint: string)
- *actionDescription* (optional, repeatable): a human readable meaningful description for the *PreservationAction*. This is not a description of a *PreservationTool* or *PreservationService,* but a description of the actual *PreservationAction Event*. (data constraint: *Description*)
- *hasParentPreservationAction* (optional, repeatable): unique identifiers to each of the parent *PreservationAction* objects (data constraint: *PreservationAction ID*)
- *hasRisk* (optional, repeatable): a unique identifier of the concrete *PreservationRisk* which prompts the *PreservationAction* (data constraint: *PreservationRisk ID*) The *PreservationRisk* object contains the information about the violated *RiskSpecifyingRequirement* and the *Environment or PreservationObject* that is at risk.
- *hasInputPreservationObject* (optional, non-repeatable): a unique identifier of the *PreservationObject* on which the *PreservationAction* is being executed (data constraint: *PreservationObject ID*) It is optional since a *PreservationAction* might only address an *Environment.*
- *hasOutputPreservationObject* (optional, non-repeatable): a unique identifier of the output *PreservationObject* which results from the execution of the *PreservationAction* (data constraint: *PreservationObject ID*)
- *hasInputEnvironment* (optional, non-repeatable): a unique identifier of the applicable *Environment* of the input *PreservationObject* (data constraint: *Environment ID*) including all sub-*Environments* and their *Characteristics* which can be used to evaluate *PreservationGuidingRequirements*
- *hasOutputEnvironment* (optional, non-repeatable): a unique identifier of the *Environment* of the output *PreservationObject* (data constraint: *Environment ID*) including all sub-*Environments* and their *Characteristics* which the *PreservationObject* would have after execution of the candidate *PreservationAction*. These can be used to evaluate *PreservationGuidingRequirement*s.

At least one input or output *PreservationObject* or *Environment* has to exist.

- *hasEnvironment* (optional, repeatable): a unique identifier of each of the *Environments* of the *PreservationAction* itself (data constraint: *Environment ID*) including the preservation tool and service which execute the *PreservationAction* and all other sub-*Environments* which can be used to evaluate *ActionDefiningRequirements*
- *hasCharacteristic* (optional, repeatable): unique identifier of each of the *Characteristics* of the *PreservationAction* (data constraint: *Characteristic ID*). Every *PreservationAction* has none or more *Characteristics* with associated *Values*.

- *hasEventOutcome* (optional, repeatable):
  - *hasRequirementsSet* (optional, repeatable): unique identifier of the sets of *Requirement*s under which this *PreservationAction* has been performed
  - *degreeOfCompliance* (optional, repeatable): specifies for a *Requirement* to what degree and by what measure the *PreservationAction* complied with the *Requirement*A *PreservationAction* can store to what degree the *Requirement*s have been satisfied. Sometimes *Characteristics* that are not referenced by any *Requirement* are lost during a *PreservationAction*; it is not, in general, possible to record their loss as they can not be listed exhaustively.
    - *hasRequirement* (mandatory, non-repeatable):(data constraint: *Requirement ID*)
    - *associatedWith* (mandatory, non-repeatable):
      vector of unique identifiers of affected *PreservationObject* and/or *Environment Instances* (data constraint: vector of *PreservationObject* or *Environment ID*s).
    - *hasMeasure* (): *Measure ID*
    - *hasOutcome* (): (data constraint: none)

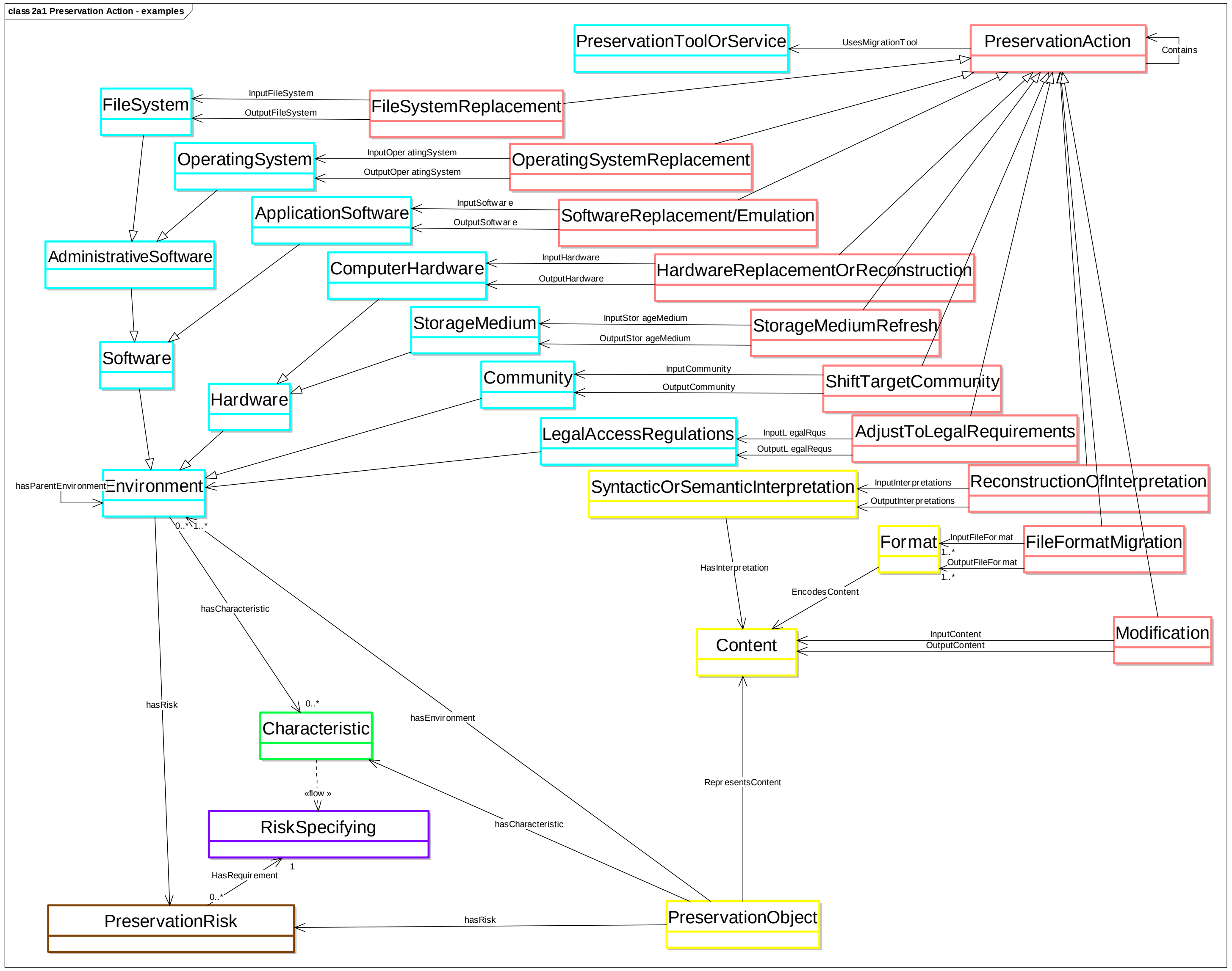

Figure 12: Example PreservationAction Subclass Depending on the Subclasses of PreservationObjects or Environments and of Risk

4.9 Property and Characteristic

In order to write with a reasonable level of precision, we need to introduce a basic vocabulary to talk about *Entities*, *Properties*, *Values*, and so on. We use an object-oriented model with roots in [Chaudhri 1998]. The core terms in this vocabulary are:

- *Entity* – Anything whatsoever.
- *Class* – A *Class* is a set of *Entities*. Each of the *Entities* in a class is said to be an *Instance* of the *Class*.
- *Individual* – *Entities* that are not *Classes* are referred to as *Individuals*.
- *Property* – A *Property* is an *Individual* that names a relationship.
- *Characteristic* – A *Property* / *Value* pair associated with an *Entity*. The *Value* is an *Entity*. This relationship is illustrated in Figure 13.
- *Constraint* – A Boolean condition involving expressions on *Entities*.

Unless otherwise specified, a *Characteristic* is directly associated with an *Entity*. It is sometimes useful to associate a *Characteristic* with all of the *Instances* of a *Class*. We refer to this as a *ClassCharacteristic*. Furthermore, we say that a *Property applies* to a *Class* if it can be meaningfully associated with some *Instances* of the *Class*.

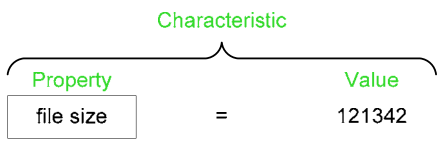

Figure 13: **Properties and Characteristics**

We can use this language in the domain of digital objects and preservation.

For example,

- *File* is a class;
- *f1.txt* is an instance of the class *File*;
- *fileSize* is a *Property*; the *Property fileSize* applies to *File*;
- File *f1.txt* has the characteristic *fileSize* = 131342;
- If every instance of *File* has been virus-scanned, then the class *File* has the class characteristic *isVirusScanned* = ”yes” which applies to all its instances.
- *Collection* is a subclass of *IntellectualEntity;*
- *MyDigitalCollection* is an instance of the class *Collection;*
- *MyDigitalCollection* has characteristics *numberOfObjectsInTheCollection = 850*, *valueOfTheCollection = 2000*.

Important additional information about a *Property* or *Characteristic*, such as how a *Value can be* encoded for a *Property* or *is* encoded for a *Characteristic*, applicable units of a *Property* or the actual unit of a *Characteristic*, or the algorithm or tool that can be used to compute the *Value* for a *Property* or have actually been used for a *Characteristic* are specified in the data dictionary below.

4.9.1 Property

Definition of Property

An abstract attribute, trait or peculiarity suitable for describing *PreservationObjects*, *PreservationActions* or *Environments*.

The model’s scope is limited to *Properties* which are expected to be used in *PreservationGuidingRequirementsSets* and are expected to be useful for preservation planning.

Observations for Property

Observation 1

Unlike the other concepts introduced so far, the *Property* concept is purely abstract and defined as part of the vocabulary of the domain of preservation planning. There are separate efforts in Planets to grow *Property* vocabulary such as in PC2, TB, and PRONOM.

Observation 2

A *Property applies* to a *Class* if it can be meaningfully associated with some *Instances* of the *Class*.

Many properties are applicable to specific subsets of objects. For example, the *Property fontSize* is applicable to *TextComponent PreservationObjects*; it would not be applicable to an *AudioComponent PreservationObjects*.[6] *Properties* are applicable to *PreservationObject*, *Environment* or *PreservationAction* subclasses.

Observation 3

The language that we use to define *Properties* must be expressive enough to refer to a combination of *PreservationObjects*, *Environments*, or *PreservationActions*. E.g. the relative size of two images to each other, the absolute distance of a line from the text, the metrics describing column layout, all refer to several objects. This means that we generalize Observation 2 to say that a *Property applies* to a vector of *Classes* if it can be meaningfully associated with some *Instances* of the *Classes* – i.e. *Properties* can be n-ary.

Observation 4

Every *Property* applies to exactly one vector of *PreservationObject, Environment, or PreservationAction* subclasses. A *Property* with the same name can be defined for other vectors of *Classes*, but will have a different globally unique *PropertyIdentifier*.

Observation 5

If a *Property* applies to *Component* or one of its subclasses, such as *TextComponent* or *ImageComponent*, we can map from the *Component* subclass to file formats to make explicit which *Component* subclasses can be captured in which file formats, and thereby capturing which *Properties* apply to which file format. See Figure 14**Error! Reference source not found.** for an illustration.

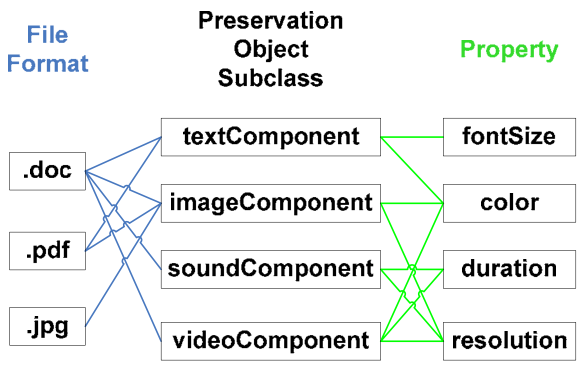

Figure 14: Some Applicable Properties are Mapped to Formats via the PreservationObject Subclass

Observation 6

Properties are related to each other and their relationships have to be modeled explicitly.

Observation 6a

In many cases, it is useful to define one *Property* in terms of others. For example, the *aspectRatio* of an image might be defined as *imageWidth* / *imageHeight*. For example *duration* can be calculated from *dateTimeRange*. As a result, it is essential to record how such *Properties* are defined and derived in order to ensure consistency.

Observation 6b

[6] The association of properties with digital object types of files is discussed in the Planets Testbed [12]. We are refining this to the type of a component of the digital object, since a logical object might well contain, for example, text, sound and image components together.

Some *Properties* are modelled hierarchically. For example *maintenanceSalaryCost* is a kind of *maintenanceCost* which is a kind of *budgetCost*. Furthermore, different file formats have similar, but not identical *Properties*. A data model of *Properties* should be able to capture the relationships between them and specify how to compare or convert them. Figure 15 illustrates this.

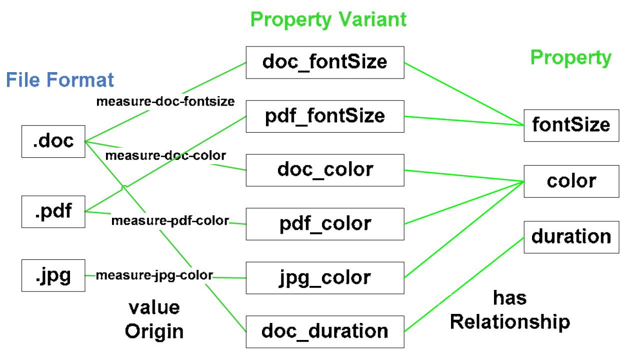

Figure 15: Properties Example: ValueOrigins and Relationships between Properties

Observation 7

For each *Property* it is essential to specify the tool and algorithm that can be used to determine a *Value* and the types of sources from which they can be obtained. We refer to this as the *ValueOrigin*. *Values* originate when they are

- Assigned manually (stored or on demand)

When *Value*s are assigned manually they often need to comply with conventions, such as cataloguing rules, standards, controlled vocabularies, etc. This should be specified as part of the *ValueOrigin*.

- Assigned automatically as a side-effect of a service (stored)

Regular internal operations, such as ingest, digitization, and harvesting of digital objects, purchase of hardware and software, decommissioning of equipment, hiring, training and laying-off of staff, getting and spending money, or executing *PreservationActions*, all change *Characteristics* of *PreservationObjects* or their *Environments*. Equally, external operations, such as introducing a new file format or a new preservation service, change *Characteristics*. These *Value* changes need to be captured if they serve as a basis for making preservation decisions.

E.g. the *contentType* of objects in an eJournal ingest system is always set to “eJournal” upon ingest.

E.g. the *budget* of an institution may be set during the execution of a *PreservationAction*: *preservationBudgetSize* := *preservationBudgetSize – preservationActionCost.*

- Extracted (stored or on demand)

The original source of derived *Value*s must be the *RepresentationBitstreams* of a *Representation* of a logical object. *Values* are extracted using a tool which implements an algorithm. The *ValueOrigin* should specify the algorithms and tools used.

Examples: *bytestream*Size may be extracted from the *Bytestream* object. *colorFidelity* can be measured by *averageColor* or by *histogramShape*. wordCount can count hyphenated words as one or as multiple words. *MIME type* can be extracted using the JHOVE format characterization tool.

NB: Characterization tools are defined to work on *Representations*. Most often we characterize digital object *Representations*, but we can also characterize at a higher level, e.g. Collection profiling tools analyse *Properties* of a *Collection* at a given time and measure their *Values*.

- Inferred (stored or on demand)

Values may be inherited in the *PreservationObject* hierarchy, derived through a function from *Value*s of other *Properties*, or logically inferred.

The *ValueOrigin* should specify the algorithm that can be used to infer it. E.g. the *aspectRatio* of an image may be *imageWidth* / *imageHeight.*

Observation 8

Every *Property* can have several units and data constraints. This is particularly important for preservation characterization. *bitDepth*, for example, is described as one non-negative number in PNG and as three nonNegativeNumbers (one for every colour channel) in TIFF. It is important to be able to specify which data constraint is chosen and also, how this data constraint can be compared to others.

Observation 9

In the model a *Property Value* can be represented with various *Units* and be obtained through various *ValueOrigins*.
Properties are defined to take on exactly one *Value* for every combination of *Unit* and *ValueOrigin*.
The *Value* for a *Property* of a given *Unit* can be converted deterministically to a different compatible *Unit*.
The *Value* for a *Property* of a given *ValueOrigin* may have (possibly systematic) differences that may or may not be related in a deterministic way to the *Value* of a different *ValueOrigin* for specified *Units*.

ValueOrigins can have a choice of *techniques*, *sources* and *agents*.
A *ValueOrigin* is defined to produce the same *Value* for a specified *Unit* for any combination of *techniques*, *sources* and *agents* defined for it.

Different *ValueOrigins* for the same *Property* may produce different *Values*, i.e. *Properties* are abstract and *ValueOrigins* produce concrete *Values* / Measurements.

Elements of Property

- *propertyIdentifier* (mandatory, non-repeatable): a unique identifier of the *Property* (data constraint: *Property ID*).
- *propertyName* (optional, repeatable): a meaningful human readable name for the *Property* (data constraint: string) It is repeatable in order to allow for synonyms. Different *Properties* may have the same names, but must have unique identifiers.
- *propertyDescription* (optional, repeatable): a human readable meaningful description for the *Property* (data constraint: *Description*)
- *appliesTo* (mandatory, non-repeatable): domain specification;
  vector of *PreservationObject*, *Environment* or *PreservationAction* sub-*Classes* to which the *Property applies*. It can be meaningfully associated with *Instances* of these *Classes;*
  "n-ary parameter list" (data constraint: vector of *PreservationObject*, *Environment* or *PreservationAction* subclasses).
  The vocabulary of subclasses can be extensible and include many subclasses not shown in this paper. Some vocabulary for subclasses can be found in Sections 4.6, 4.7, and 4.8.
- *hasRange* (optional, repeatable): range specification; constraints on or enumeration of permissible *Values*; a data type definition for the *Value*; possibly a URI pointing to the defined vocabulary for the *Property*
  - *hasUnit* (optional, non-repeatable): See Section 4.9.3. A unique identifier of the unit.
  - *hasData*Constraint (mandatory, non-repeatable): permissible *Values*; a type definition for the *Value*; possibly a URI for defined vocabulary for the *Property* (data constraint: taken from an extensible set of data constraints) Data constraints are combined with the unit definition, as different units may have different data constraints. (E.g. K: ≥0, °C: ≥ -273.15, °F: ≥ -459.67). It has to be compatible with the *Unit*'s data constraint.
  - *isDefault* (optional, non-repeatable): indicates whether this range specification is the default range for this *Property* (data constraint: “yes” or “no”)
  - *hasDefault Value* (optional, non-repeatable): a default *Value* for this *Property*.
- *hasValueOrigin* (optional, repeatable): How the *Value*s for the *Property* may be obtained or updated (if it is stored)
  - *hasValueOriginID* (mandatory, non-repeatable): a unique identifier of the *ValueOrigins*. See Section 4.9.2.
  - *isDefault* (optional, non-repeatable): indicates whether this *ValueOrigin* is the default for this *Property* (data constraint: “yes” or “no”)
- *hasRelationship* (optional, repeatable): relationship to other *Property* concepts with related semantics (relationships that cannot serve as *ValueOrigin*).
  - *hasRelatedProperty* (mandatory, non-repeatable): (data constraint: *Property ID*)
  - *hasRelationshipType* (mandatory, non-repeatable): a type specification of the relationship to an other *Property* concept (data constraint: local usage, such as generalizationOf, specializationOf, siblingOf, inverseOf, disjointOf, smallerThan, or any association name)

- *hasEvent* (optional, repeatable): unique identifiers to each of the *Property's Event* objects, such as versioning (data constraint: *Event ID*)

Other Relationships with Property

- *Characteristic* has a *hasProperty* relationship with *Property*.

Example for Property

The example in Figure 16 shows a definition of the *Property imageSizeWidth* for an *ImageFile PreservationObject*.

This *Property* definition has 3 types of units: inches, centimetres, and points. They all are valid alternative units, however, if not specified, it is assumed that "points" are the default unit.

The *Value* may be created in two ways: It may be assigned by digitization software DigitizR on creation of the image. Alternatively it may be characterized from an existing file by the JHOVE file format characterization tool.

imageSizeWidth may be derived in two ways. It can be calculated if *aspectRatio* and *imageHeight* are known by using the conversion function associated with these *Properties*. Alternatively it can be derived if the *Property imageSizeWidth*_GIF is known, since they are known to be equivalent *Values*.

propertyIdentifier
    http://ontology.xxx.yyy/1234
propertyName
    imageSizeWidth
propertyDescription
    The width of an image. No default value is provided. The default measurement unit is In.
appliesTo
    ImageFile
range (isDefault="yes")

- ***hasUnit***
  <UnitID for inches>
- ***dataConstraint***
  positive or zero float

range

- ***hasUnit***
  <UnitID for centimeters>
- ***dataConstraint***
  positive or zero float

range

- ***hasUnit***
  <UnitID for points>
- ***dataConstraint***
  positive or zero int

hasDefaultValue
hasValueOrigin

- ***hasValueOriginID***
  <ValueOriginID for JHOVE Version 1.1 extractor of imageWidth>
- ***isDefault***
  "no"

hasValueOrigin

- ***hasValueOriginID***
  <ValueOriginID for DigitizR Version2.5 - imageWidth>
- ***isDefault***
  "no"

hasValueOrigin

- ***hasValueOriginID***
  <ValueOriginID for conversion from aspectRatio and imageSizeHeight>
- ***isDefault***
  "no"

hasValueOrigin

- ***hasValueOriginID***
  <ValueOriginID for conversion from gif_format_imageWidth>
- ***isDefault***
  "no"

hasEvent
<CreationEventID giving creation time and author of this property>

ValueOrigin

- ***valueOriginIdentifier***
  <ValueOriginID for conversion from aspectRatio_imageSizeHeight>
- ***valueOriginName***
  "conversion from aspectRatio and imageSizeHeight to imageSizeWidth"
- ***hasTechnique***
  <conversion function> (aspectRatio(self), imageSizeHeight(self))

ValueOrigin

- ***valueOriginIdentifier***
  <ValueOriginID for conversion from gif_format_imageWidth>
- ***valueOriginName***
  "conversion from gif_format_imageWidth to imageSizeWidth"
- ***hasTechnique***
  "is same"

Figure 16: Example Property and ValueOrigin

Vocabulary for Specifying Properties

In the Appendix of document [PP2-D2 2008] we list an initial collection of *Property* vocabulary for a subset of the *Environment* and *PreservationObject* subclasses. The goal is to have a deep vocabulary that would be generally acceptable and sharable by different stakeholders. For certain subsets one can refer to related work. Work in Planets work-package PC2, TB and PRONOM are creating *Property* ontologies. Metadata initiatives for descriptive metadata [MODS, DC, MARCXML, TEI, NLM etc.], technical metadata [NISO_MIX, TEXTMD] and preservation metadata have elaborated useful vocabularies for Properties. For example, the PREMIS [PREMIS 2008] preservation metadata defines *Properties* for *Representations*, *Files* and *Bitstreams*.

4.9.2 ValueOrigin

Definition of ValueOrigin

The *ValueOrigin* concept provides a way to specify where a specific *Value* comes from or how it can be obtained. There can be multiple ways of obtaining the *Value* of a *Property* that do not produce conflicting results, measured from various sources, measured by various techniques, using various tools, or obtained through various agents.

Elements of ValueOrigin

- *valueOriginIdentifier* (mandatory, non-repeatable): a unique identifier of the *ValueOrigin* (data constraint: none)
- *valueOriginName* (optional, repeatable): a human readable meaningful descriptor for the *valueOrigin* (data constraint: string)
- *valueOriginDescription* (optional, repeatable): a human readable meaningful description for the *ValueOrigin*. (data constraint: *Description*)
- *hasSource* (optional, repeatable): a type specification of the sources from which the *Value* can be measured or derived (data constraint: none). Sources for the *Value* may be registries or inventories, Values for other *Properties* from which the *Value* can be derived (In that case the source would have to be a list of parameter definitions including the *Unit* and *ValueOrigin* of the source-*Properties*), or *Representations* of the *IntellectualEntities* from which the *Value* can be measured. There may be a chain of *ValueOrigins* where one *ValueOrigin* is the source for another.
- *hasTargetUnit* (optional, repeatable): a specification of the Unit of the *Value* to be created by this ValueOrigin. (data constraint: *Unit ID*)

- hasT*echnique* (optional, repeatable): Rule, algorithm or logic used for obtaining the *Value* (e.g. assigned according to Anglo-American Cataloguing Rules, extracted from .tiff file metadata) (data constraint: none) Dependent on whether the *Value* is created manually or automatically different preservation processes need to be used.
  One special *technique* is the specification of a conversion. Conversions specify how a *Value* for the *Property* may be derived from other *Properties* for specified *Units* and *ValueOrigins*, or from the same *Property* obtained by other *ValueOrigins* for specified*Units*, since the same *Property* can have slightly different measurement results when it is measured using different *ValueOrigins*, e.g. through systematic errors.
- has*Agent* (optional, repeatable): For automatically derived *Values*: software tool and version; for manually assigned *Values*: person role (data constraint: *Agent ID*) There may be multiple possible *Agents*.
- *hasTrigger* (optional, repeatable): a trigger for *Value* assignment: e.g. ingest, *PreservationService,* etc. (data constraint: none)

Other Relationships with ValueOrigin

- *Property* has a *hasValueOrigin* relationship *ValueOrigin.*

4.9.3 Unit

Every *Property* can have several *Units*. This is particularly important for preservation characterization. *bitDepth*, for example, is described as one non-negative number in .png and as three non-negative numbers (one for every colour channel) in .tiff. It is important to be able to specify which *Unit* is chosen and also how this *Unit* can be compared to others.

Elements of Unit

- *unitIdentifier* (mandatory, non-repeatable): a unique identifier of the *Unit* (data constraint: none)
- *unitName* (optional, repeatable): (data constraint: string) allows for synonyms; e.g. inches, Zoll
- *unitDescription* (optional, repeatable): a human readable meaningful description for the *Unit* (data constraint: *Description*)
- *hasData*Constraint (mandatory, non-repeatable): permissible *Values*; a type definition for the *Value*; possibly a URI for defined vocabulary (data constraint: taken from an extensible set of data constraints)
- *hasConversion* (optional, repeatable): How *Values* may be converted from another *Unit* to this *Unit* . This is important for preservation characterization and comparison.
  - *hasSource* (mandatory, non-repeatable): Identifier of the source *Unit* (data constraint: *Unit ID*)
  - *hasTechnique* (mandatory, repeatable): Rule, algorithm or logic used for mapping or converting the *Value* (e.g. FFT) (data constraint: none) There may be multiple ways of deriving the *Value*.
  - *hasAgent* (optional, repeatable): conversion software tool and version; (data constraint: *Agent ID*) There may be multiple possible agents.

Other Relationships with Unit

- *Property, Characteristic* and *Requirement* have a *hasUnit* relationship with *Unit*

4.9.4 Characteristic

Definition of Characteristic

A *Characteristic* of an *Entity* is the concrete *Value* which this *Entity* has for an abstract *Property* in a defined context (a concrete *Property/Value* pair).
In the model it is the *Characteristic* of a *PreservationObject, Environment* or *PreservationAction*.

Observations for Characteristic

Observation 1

In our model, each of the concepts *PreservationObject*, *Environment*, and *PreservationAction*, may have Characteristics. This is a key aspect of our model.

PreservationObjects may have *Characteristics*. For example, *alignment*= “left” is a *Characteristic* of a *textComponent. semanticInterpretation*=”body weight” is a *Characteristic* of a *numberComponent.*

PreservationActions may have *Characteristics*. For example, *numberOfIntermediateCopiesProduced* = 2 is a *Characteristic* of a *PreservationAction*. It may, for example, be used to identify *PreservationActions* which violate copyright regulations which limit the number of intermediate copies created.

Environments may have Characteristics.

Examples:

- *memoryUsage* = “low” is a *Characteristic* of a *SoftwareToolEnvironment* that renders the *PreservationObject*.
- *numberOfIntermediateCopies* <=3 and *preservesColorDepth* = “yes” are *Characteristics* of a *PreservationService* which is part of a *PreservationAction*’s environment.[7]

It is essential to always be clear with which *Entity* the Characteristic is associated, i.e. for which *PreservationObject, Environment* or *PreservationAction* this *Characteristic* holds.

Observation 2

Values for *Characteristics* may be stored or derived on demand. On demand derivation can take place through characterization services or through retrieval from registries or inventories[8]. Whether they are stored or derived needs to be recorded since different *PreservationServices* will be chosen based on this *Property*.

Observation 3

There may be multiple *Value*s for a *Property* of an object, since there may be several *Representations* (*hasSources*) which form the basis of measurement for the *Value* and several different measurement techniques (*hasTechnique*) and tools (*hasAgent*). *Characteristics* and *Requirement*s need to specify which *ValueOrigin* is meant.

Observation 4

Characteristics are used to express *Requirements* which then inform the choice of *PreservationAction*.

Elements of Characteristic

- *characteristicIdentifier* (mandatory, non-repeatable): a unique identifier of the *Characteristic*. Having a unique identifier for a *Characteristic* supports different *Values* for the same *Property* at different times. (data constraint: *Characteristic ID)*
- *associatedWith* (mandatory, non-repeatable):
  vector of unique identifiers of *PreservationObject*, *Environment* or *PreservationAction Instances* with which the *Characteristic* is associated. It can be meaningfully associated with *Instances* of the *Classes* defined in the *appliesTo* element of the corresponding *Property* concept.
  (data constraint: vector of *PreservationObject*, *Environment* or *PreservationAction ID*s).
  (This relationship is also established via the has*Characteristic* relationship of *PreservationObject*, *Environment*, or *PreservationAction*)
- *hasProperty* (mandatory, non-repeatable): a specification of the *Property* to which this *Characteristic* refers
  - *propertyIdentifier* (mandatory, non-repeatable): It specifies for which *Property* the *Characteristic’s Value* holds (data constraint: *Property ID*)
  - *annotations* (optional, non-repeatable): chosen from the allowable values specified in the corresponding *Property* definition.
    - *hasUnit* (optional, non-repeatable): a unique identifier of the *Unit* of the *Value* (data constraint: *Unit ID*)
    - *hasValueOrigin* (optional, non-repeatable): a unique identifier of the *ValueOrigin* which specifies how the *Value* is to be obtained on demand or was obtained, if

[7] They are class *Characteristics* which can be captured in a *PreservationServices* registry. If the service *Characteristic* reflects constant behaviour and the *ValueOrigin* can be trusted, it can be inherited to the *PreservationActions* that are executed by this service. In that case, the characteristic *colorDepth* need not be measured and compared for individual *PreservationActions* since it is known to be preserved before hand.

[8] Such as software licenses, hardware inventories, standards and XML schemata in use, staff skills, etc.

stored. (data constraint: *ValueOrigin ID)* The *technique, source* and *agent* employed must be of the types specified for the *ValueOrigin* and *Property*.

- *hasSource* (optional, non-repeatable)
- *hasTechnique* (optional, non-repeatable)
- *hasAgent* (optional, non-repeatable)

- *onDemandP* (optional, non-repeatable): a specification of whether the *Value* is stored locally or should be derived on demand (data constraint: one of *local, onDemand).* Registry look-up can considered an on-demand access.
- *hasValue* (optional, non-repeatable): *Value* of the *Characteristic*, if it is stored locally (data constraint: none)
- *hasCreationEvent* (optional, non-repeatable): a unique identifier of the *Event* which created the *Value* if it is stored locally (data constraint: *Event ID*) including the date the *Value* was set. In addition, information to capture versioning information such as a date range of applicability of the *Value*, previous *Values* for the same *Property* and objects, etc. will be desirable

Other Relationships with Characteristic

- *PreservationObject*, *Environment*, and *PreservationAction* have a *hasCharacteristic* association link with *Characteristic*.

4.10 Requirement

Definition of PreservationRequirement

A constraint which limits the space of allowable preservation activities.

Observations for Requirement

Observation 1

Requirements are constraints that make the stakeholder's values explicit and influence the preservation process.

Requirements are measurable subsets of goals. They express a target level of results expressed in units against which achievement is to be measured. *Requirements* provide the day-to-day support for achieving goals. [adopted from StratML, Objectives]

Observation 2

Requirements are described in *PreservationGuidingRequirementsSets.*

Observation 3

Requirements can be expressed as constraints, such as through OCL [OCL 2003] or other informal or formal languages.

They can be expressed through one or more *Property/Value* constraint specifications on *PreservationObjects*, *Environments* or *PreservationActions* and any of their subclasses.

Observation 3a

In many cases, a stakeholder would like to make *Requirement*s dependent on additional conditions - that is, a context needs to be specified. The conditions involve *Characteristics* of *PreservationObjects, Environments* or *PreservationActions*.

- If *componentType* = "text" then *fontSize* must be preserved.
- If *environmentType* = "archival preservation" then *imageResolution* must be preserved.
- If *preservationActionType* = "bitPreservation" then *fileSize* must be preserved.

As a result, the language that we use to define *Requirement*s must be expressive enough to include conditionals**.**

Observation 3b

Requirements often need to include specifications such as invariants, pre-conditions and post-conditions.

Observation 3c

A stakeholder may only instantiate consistent, non-contradictory sets of *Requirements*.

Observation 4

Stakeholders may like to specify an importance factor which is a measure of the relative importance of the *Requirement* for the stakeholders. I may consider each of two conflicting *Requirement*s important and prioritise one as more important than another. This prioritisation is essential for both decision making and planning.

Equally, *Requirements* may tolerate some deviation or error. For example, an office document migration that produced a result with different hyphenation or pagination might be acceptable in many situations. *Requirements* should contain a tolerance factor which specifies to what degree deviation from the required value can be tolerated.

During the comparative evaluation of candidate *PreservationActions* the importance and tolerance factors can be combined into a weighted measure.

Observation 5

While *Characteristics* capture *Values* at a given moment in time, *Requirements* capture constraints on *Characteristics* across time – before and after a *PreservationAction*.

Observation 6

During preservation planning one determines which of the candidate *PreservationActions* is the most suitable for the *PreservationObject*. This can be derived by considering the *Characteristics* of the *PreservationObject* and *Environment* before and after the execution of a candidate *PreservationAction*, and by comparing them to the stakeholder*'s Requirements*. This process lets us derive to what degree this *PreservationAction* would satisfy the *Requirements*. It amounts to a cost/benefit analysis of the *PreservationAction*.

Observation 7

Requirement evaluators (e.g. the XCDL comparator [Thaller 2008]) determine the degree to which *Characteristics* of *PreservationObjects*, *PreservationActions* and *Environments* comply with *Requirements* before, during and after *PreservationActions*. The importance and tolerance factors can be combined with the degree of compliance into a weighted measure. The output is a combined measure of the degree of compliance with the *RequirementsSet*.

Observation 8

The output *Characteristic* is not necessarily inferior to the input *Characteristic*, i.e. preservation is not always lossy. In many cases, we wish to include the possibility of capturing improvements to an object. A common *PreservationAction* is normalization of digital objects upon ingest. This may be done to reduce the variety of formats held, but may also be done to improve *Characteristics* in the original. For example, we might migrate files which are in formats that are susceptible to degradation to files in a more resilient format, or move static tables to spreadsheets which enable pivot tables. In this case the *Characteristics fileFormatResilience* = "high" or *enablesPivotTables* = "yes" are *SignificantCharacteristics* which were not found in the original and can be captured in a *Requirement*. Another preservation action which improves upon the original is the manual restoration of a file by a curator to the state it was presumed to have had before a corruption. Another common example can be found in CAD drawings or data sets. As technology improves, consumers desire to perform new functions on old data in ways that were previously not possible

Vocabulary for Requirement Subclasses

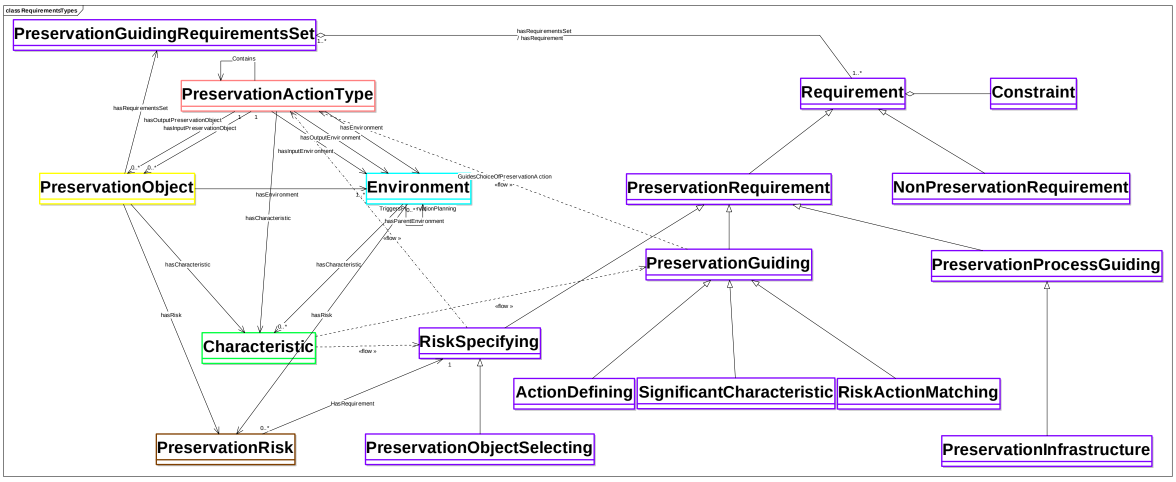

Figure 17: Requirement Subclasses

Degradation to *PreservationObjects* is caused by two things:

- *PreservationRisks*
- Executing *PreservationActions*, which might not preserve all *Characteristics* of the *InputPreservationObject* or *InputEnvironment* in the newly created *OutputPreservationObject* or *OutputEnvironment*.

Acceptable levels of either are described in *Requirements*. *RiskSpecifyingRequirements* capture the first category. *PreservationGuidingRequirements* capture the second category.

Different *Requirements* categories play different roles in different preservation services. Figure 17 illustrates the *Requirement* subclasses.

4.10.1 RiskSpecifyingRequirement

RiskSpecifyingRequirements state explicitly what the perceived risks for *PreservationObjects* and *Environments* are. Whenever *Characteristics* of a *PreservationObject* or its *Environment* violate constraints which are specified in the *Requirement* then the *PreservationObject* is considered at risk.

Once a *RiskSpecifyingRequirement* is violated, a preservation monitoring process should trigger the preservation planning process. It, in turn, determines the optimal *PreservationAction* which should mitigate this *PreservationRisk*.

PreservationObjectSelectingRequirements are a special class of *RiskSpecifyingRequirements* which specifies which subset of *PreservationObjects* is at risk.

PreservationRisk subclasses (see Figure 10**Error! Reference source not found.**) *NewVersion*, *NotSupportedOrObsoleteSupport*, *DeteriorationOrLoss*, *Proprietary*, and *UnmangedGrowth* could also be used to create sub-categories of *RiskSpecifyingRequirement* if this distinction supports the preservation processes.

4.10.2 PreservationGuidingRequirement

PreservationGuidingRequirements determine the cost/benefit of a *PreservationAction* by explicitly stating the stakeholder*'s* values. The degree to which the *PreservationAction* satisfies those *Requirements* determines its cost/benefit for the stakeholder.

They define which kinds of *PreservationActions* are desirable for the *PreservationObject*, dependent on

- which input *Characteristics* of the *PreservationObject* and its *Environment* need to be met to consider the *PreservationAction*
- which output *Characteristics* of the *PreservationObject* and its *Environment* are permissible/ desirable
    - dependent on input *Characteristics*
        - compares the differences between the input and output *Characteristics* and measures to what degree this difference satisfies the required *Characteristics*. (The input *PreservationObject* and its *Environment* might be a derivative or the original submitted to the stakeholder[9].)
        - Example: “The loss of resolution may not exceed 20% of the original resolution”.
    - in absolute terms, independent of input *Characteristics*
        - measures to what degree the output *Characteristic* satisfies the required *Characteristic*.
        - Example: The size of the *PreservationAction*’s output *PreservationObject* should not exceed a maximal size set by the stakeholder.
- which *Characteristics* of the *PreservationAction* itself are desirable
    - Example: “Output file formats need to be platform independent.”

Example: The size of the *PreservationAction*’s output *PreservationObject* should not exceed a maximal size set by the stakeholder.

ActionDefiningRequirements are a special class of *PreservationGuidingRequirements*. They define which kinds of *PreservationActions* are desirable independent of the *Characteristics* of the *PreservationObject*, but dependent only on the *Characteristics* of the *PreservationAction* itself.

Example: PDF may, for a given stakeholder, not be an acceptable preservation output format of a *PreservationAction*).

SignificantCharacteristics are a special class of *PreservationGuidingRequirements*. They are discussed in detail in [PP2-SigChar 2009]. They define which *Characteristics* must be met by output *PreservationObjects* and *Environments*. Their applicability may depend on *Characteristics* of *PreservationActions*.

Example: "If *preservationActionType* = “migration” then *fileFormat* of the output *PreservationObject* must be non-proprietary."

RiskActionMatchingRequirements are a special class of *PreservationGuidingRequirement.* They specify that a candidate *PreservationAction* has to be an appropriate match to a given *PreservationRisk* as was illustrated in Figure 12.

[9] It is important to not accumulate errors in subsequent *PreservationActions*, which implies that it is best to express comparative losses with respect to the original *PreservationObject*.

4.10.3 PreservationProcessGuidingRequirement

PreservationProcessGuidingRequirements are a special class of *PreservationRequirement*. They describe the preservation process itself independent of the *Characteristics* of the *PreservationObject*, its *Environments*, as well as of those of the *PreservationAction*. They may prompt the preservation planning process but do not influence it.

Example: A preservation planning process should be executed for every data object at least every 5 years, independent of the *PreservationRisks* that are established for this data object.

Preservation Infrastructure Requirements are a special class of *PreservationProcessGuidingRequirements* which specifies what *Characteristics* are required of the infrastructure with respect to security, networking, connectivity, storage, etc..

Example: Mirror versions of on-site systems must be provided.

4.10.4 NonPreservationRequirement

NonPreservationRequirements are a special class of *Requirements*. They specify processes relevant to preservation, but not part of preservation itself.

Elements of Requirements

- *requirementIdentifier* (mandatory, non-repeatable): a unique identifier of the *Requirement* (data constraint: Requirement *ID*)
- *requirementName* (optional, repeatable): a human readable meaningful name for the *Requirement* (data constraint: string)
- *requirementDescription* (optional, repeatable): a human readable meaningful description for the *Requirement* (data constraint: Description)
- *hasRequirementsSet* (optional, repeatable): a unique identifier of the *RequirementsSet* to which the *Requirement* belongs (data constraint: *PreservationGuidingRequirementsSet ID*)
- *hasStakeholder* (optional, repeatable): (data constraint: *Agent ID*)
- *requirementSource* (optional, repeatable)
- *requirementApplicability* (optional, non-repeatable): Time range during which the *Requirement* is applicable. If it is not specified explicitly, then it defaults to the *Value* of the *applicability* element of the *PreservationGuidingRequirementsSet* in which the *Requirement* is captured.
  - *startDate* (optional, non-repeatable): The date the *Requirement* is projected to become valid (data constraint: date)
  - *endDate* (optional, non-repeatable): The date the *Requirement* is projected to cease, if it is not subsequently extended (data constraint: date)
- *requirementSpecification* (mandatory, non-repeatable):
  - *context* (optional, repeatable): Specifies the objects for which the constraint holds
  - *pre* (optional, non-repeatable): Specifies a pre-condition for applying the requirement
  - *post* (optional, non-repeatable): Specifies a post-condition for applying the requirement

  The *requirementSpecification* element is a constraint which can be modelled similar to constraint languages such as OCL [OCL 2003]. Each pre- and post-condition is a logical expression which combines constraints and can be evaluated to true or false for a given set of *Characteristics Values*.

  In general a constraint will contain some of the following parts:
  - *operator*: Operator to be applied to determine whether the *Requirement* is satisfied.
    - *operator* (mandatory, non-repeatable): Function to be evaluated. e.g. "=", "one of", "MyBooleanFunction". The function should evaluate to true/false. If a tolerance is specified the function might return the degree to which the constraint is satisfied with respect to the tolerance.
    - *tolerance* (optional, non-repeatable): To what degree deviation from the *Requirement* can be tolerated.

- *property*: It specifies for which *Property* a value should be retrieved. A *Property* is fully specified by the following elements
  - *propertyIdentifier* (mandatory, non-repeatable): It specifies for which *Property* a *Value* should be retrieved.
  - *annotations* (optional, non-repeatable): It specifies which of the annotations listed within the *Property* definition should be used to derive the *Value*.
    - *hasUnit* (optional, non-repeatable): a unique identifier of the *Unit* of the *Value* (data constraint: *Unit ID*)
    - *hasValueOrigin* (optional, non-repeatable): a unique identifier of the *ValueOrigin* which specifies how the *Value* is to be or was obtained. (data constraint: *ValueOrigin ID)* The *technique, source* and *agent* employed must be of the types specified for the *ValueOrigin* and *Property.*
    - *hasSource* (optional, non-repeatable):
    - *hasTechnique* (optional, non-repeatable):
    - *hasAgent* (optional, non-repeatable):
- *constant*: It specifies a constant *Value*. A constant is fully specified by the following two elements
  - *value* (mandatory, non-repeatable)
  - *unit* (mandatory if applicable, non-repeatable)

  Units in *requirementsSpecifications* have to agree with *Units* of *Characteristics Values* and the *Property*'s *hasUnit* (must be the same or have a conversion specified).

- *requirementImportanceFactor:* Measure of the relative significance of the *Requirement* for the stakeholder (data constraint: none)
- *hasEvent* (optional, repeatable): unique identifiers to each of the *Requirement's Event* objects (data constraint: Event *ID*)

The *requirementImportanceFactor* and the *tolerance* elements allow for computing a weighted measure of compliance with the *Requirement*.

Other Relationships with Requirement

- *PreservationGuidingRequirementsSet* has a *hasRequirement* aggregation link to *Requirement*.
- *PreservationRisk* has a *hasRiskSpecifyingRequirement* association link to *RiskSpecifyingRequirement*.

Example of Requirement

The following example illustrates how a *Requirement* may be expressed solely in terms of model elements and vocabulary.

The *Requirement* "**Textual data must be migrated to RTF 1.8"** is being mapped in the following way:

The context of the *Requirement* describes the *Class* to which the precondition, post-condition, or invariant applies. In this example it describes restrictions on eligible *PreservationActions*.

The precondition describes under which circumstances the *Requirement* applies. This is expressed solely in terms of the *hasInputPreservationObject* relationship between *PreservationAction* and *PreservationObject,* and in terms of the *hasCharacteristic* element of *PreservationObject*.

The post-condition, finally, describes which conditions need to be true after a *PreservationAction* is executed under the given circumstances. Again this is expressed using relationships and elements introduced in the above data model.

Context:
PreservationAction: a
  class-of (a): "replacement preservation action"
  hasInputPreservationObject: i
  hasOutputPreservationObject: o

Precondition:

PreservationObject
  preservationObjectIdentifier: i
  class-of (i): "File"
  hasCharacteristic: x

Characteristic
  characteristicIdentifier: x
  associatedWith: (i)
  hasProperty: p9067
  hasValue: "text"

Property
  propertyIdentifier: p9067
  propertyName: "formatType"
  appliesTo: (Bytestream)
  range
    hasDataConstraint: formatType vocabulary
  hasValueOrigin:
    hasValueOriginID: vo12756
      (e.g. this might specify the software that characterizes the formatType)

Postcondition:

PreservationObject
  preservationObjectIdentifier: o
  class-of (i): "File"
  HasCharacteristic: y

Characteristic
  characteristicIdentifier: y
  hasObject: o
  hasProperty: p782
  hasValue: "fmt/53"
    (this is the unique identifier (PUID) for RTF 1.8 in the PRONOM registry)

Property
  propertyIdentifier: p782
  propertyName: "formatDesignation"
  range
    hasDataConstraint: PUID
  hasValueOrigin:
    hasValueOriginID: vo908
      (e.g. this might specify the PUID look-up in the PRONOM registry)

Figure 18: Example Requirement

5. Conclusion

This report contains a conceptual model for the preservation domain. It consists of a UML model, a data dictionary, roots for useful vocabulary in the domain, and a machine-interpretable model. It presents a simple yet expressive representation of the preservation services domain.

It builds on the idea of preservation as a process to identify and mitigate risks to current and future access to digital objects.

It allows for uniform treatment of preservation processes on all levels of the *PreservationObject* hierarchy, such as *IntellectualEntities, Representations* and *Bitstreams.* Since the preservation planning process naturally encompasses requirements on all these levels it is equally applicable to all preservation processes that express *Characteristics* and *Requirements* on these levels, such as monitoring, characterization, comparison of *Characteristics*, evaluation of candidate *PreservationAction*s, etc..

The vocabulary offers a starting point for creating individualised models for individual stakeholders, even if the stakeholder does not aim for a machine-interpretable document. The goal is to have a deep vocabulary that would be generally acceptable and sharable by different stakeholders. The vocabulary currently covers core applications. Work on growing vocabulary in the domain is taking place in many workpackages within and outside of Planets.

Many *Characteristics* and *Requirement*s can by their nature not be captured in a machine-interpretable form. But this fact does not distract from the value of the model. Even so, the model can be used to guide thinking and communication.

We have now arrived at a stable version of the model which is aligned with the investigated work. Further proof of concept will require use of its features in implemented systems.

Deliverables from this work-package were:

- terminology suitable for describing the domain [this report]
- a discussion of suitable top-down and bottom-up approaches for arriving at a model of this kind, including [PP2-D2 2008]
    - a literature review
    - documented interviews on the state of preservation policy and strategy use at leading institutions
    - a discussion of differences between preservation requirements needed to describe preservation policies and strategies at different institutional types
- a worked example that gives an overview of how the model and vocabulary in this report can be used [PP2-D2 2008]
- a conceptual model [this report, PP2-PresGuid 2008]
- a data dictionary for this model [this report]
- a vocabulary for the model [this report, PP2-D2 2008]
- a machine-interpretable XML representation [this report]
- an analysis on and definition of the role of Significant Characteristics within the preservation domain [PP2-SigChar 2009]
- a survey and definition of the dynamic forms of metadata needed to guide digital preservation services and enable them to interact successfully [PP2-PresServ 2009]

6. Appendices

6.1 Machine-Interpretable Model

This section contains an xsd implementation of the data dictionary..

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.planets-project.eu/pp2" elementFormDefault="qualified"
        attributeFormDefault="unqualified" xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:stratml="http://www.stratml.net" xmlns:pp2="http://www.planets-project.eu/pp2"
        xmlns:premis="http://www.loc.gov/standards/premis/v1">

        <import namespace="http://www.stratml.net"
                schemaLocation="http://xml.gov/stratml/draft/StrategicPlan.xsd"/>

        <import namespace="http://www.loc.gov/standards/premis/v1"
                schemaLocation="http://www.loc.gov/standards/premis/v1/PREMIS-v1-1.xsd" />

        <complexType name="PreservationGuidingRequirementsSetType">
                <sequence>
                        <element name="setIName" minOccurs="0" maxOccurs="unbounded">
                                <complexType>
                                        <attribute name="setVersion" type="string" use="optional"/>
                                </complexType>
                        </element>
                        <element name="setApproval" maxOccurs="unbounded" minOccurs="0">
                                <complexType>
                                        <sequence>
                                                <element name="initiator" maxOccurs="unbounded" minOccurs="0" type="IDREF"/>
                                        <!-- Agent ID -->
                                </sequence>
                                <attribute name="status" type="string" use="required"/>
                                <!-- proposed, approved, superseded -->
                                <attribute name="statusDate" type="date" use="required"/>
                        </complexType>
                </element>
                <element name="setApplicability" maxOccurs="1" minOccurs="1">
                        <complexType>
                                <attribute name="startDate" type="date" use="optional"/>
                                <attribute name="endDate" type="date" use="optional"/>
                        </complexType>
                </element>
                <element ref="stratml:Organization" minOccurs="0" maxOccurs="1"/>
                <element ref="stratml:Source" minOccurs="0" maxOccurs="1"/>
                <element ref="stratml:Vision" minOccurs="0" maxOccurs="unbounded"/>
                <element ref="stratml:Mission" minOccurs="0" maxOccurs="unbounded"/>
                <element ref="stratml:Value" minOccurs="0" maxOccurs="unbounded"/>
                <element name="Goal" minOccurs="1" maxOccurs="unbounded">
                        <!-- Can I use a pp2:Goal but embed in it some stratml elements, except for the pp2:hasRequirement? -->
                        <complexType>
                                <sequence>
                                        <element ref="stratml:SequenceIndicator" minOccurs="0" maxOccurs="1"/>
                                        <element ref="stratml:Name" minOccurs="0" maxOccurs="1"/>
                                        <element ref="stratml:Description" minOccurs="1" maxOccurs="1"/>
                                        <element ref="stratml:Stakeholder" minOccurs="0" maxOccurs="unbounded"/>
                                        <element name="hasRequirement" minOccurs="0" maxOccurs="unbounded">
                                                <complexType>
                                                        <attribute name="idref" type="IDREF" use="required"/>
                                                        <!-- lRequirement ID -->
                                                </complexType>
                        </element>
                        <element ref="stratml:OtherInformation" minOccurs="0" maxOccurs="1"/>
                    </sequence>
                </complexType>
            </element>
            <element name="references" maxOccurs="unbounded" minOccurs="0">
                <complexType>
                    <sequence>
                        <element name="hasCollection" minOccurs="0" maxOccurs="unbounded">
                            <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
                            </complexType>
                            <!-- Type should be ID or specification of a set of PreservationObjects -->
                        </element>
                        <element name="hasRegistryReference" minOccurs="0" maxOccurs="unbounded">
                            <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
                                <!-- Registry ID -->
                            </complexType>
                        </element>
                        <element name="hasPredecessorRequirementsSet" minOccurs="0"
                            maxOccurs="unbounded">
                            <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
                                <!-- RequirementsSet ID -->
                            </complexType>
                        </element>
                        <element name="hasRelatedRequirementsSet" minOccurs="0"
                            maxOccurs="unbounded">
                            <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
```

```
                                                <!-- RequirementsSet ID -->
                                        </complexType>
                                </element>
                        </sequence>
                </complexType>
            </element>
        </sequence>
        <attribute name="id" type="ID" use="required"/>
        <!-- PreservationGuidingRequirementsSet ID -->
</complexType>

<complexType name="PreservationObjectType">
        <sequence>
            <element name="preservationObjectName" type="string" maxOccurs="unbounded" minOccurs="0"/>
            <element name="preservationObjectDescription" type="string" minOccurs="0"
                maxOccurs="unbounded"/>
            <element name="hasParent" minOccurs="0" maxOccurs="unbounded">
                <complexType>
                        <attribute name="idref" type="IDREF" use="required"/>
                        <!-- PreservationObjectID -->
                </complexType>
            </element>
            <element name="hasEnvironment" minOccurs="0" maxOccurs="unbounded">
                <complexType>
                        <attribute name="idref" type="IDREF" use="required"/>
                        <!-- Environment ID -->
                </complexType>
            </element>
            <element name="hasCharacteristic" minOccurs="0" maxOccurs="unbounded">
                <complexType>
                        <attribute name="idref" type="IDREF" use="required"/>
                <!-- Characteristic ID -->
            </complexType>
        </element>
        <element name="hasRisk" minOccurs="0" maxOccurs="unbounded">
            <complexType>
                <attribute name="idref" type="IDREF" use="required"/>
                <!-- PreservationRisk ID -->
            </complexType>
        </element>
        <element name="hasRequirementsSet" minOccurs="0" maxOccurs="unbounded">
            <complexType>
                <attribute name="idref" type="IDREF" use="required"/>
                <!-- PreservationGuidingRequirementsSet ID -->
            </complexType>
        </element>
        <element name="hasStakeholder" minOccurs="0" maxOccurs="unbounded">
            <complexType>
                <attribute name="idref" type="IDREF" use="required"/>
                <!-- Agent ID -->
            </complexType>
        </element>
        <element name="hasRight" minOccurs="0" maxOccurs="unbounded">
            <complexType>
                <attribute name="idref" type="IDREF" use="required"/>
                <!-- Right ID -->
            </complexType>
        </element>
        <element name="hasEvent" minOccurs="0" maxOccurs="unbounded">
            <complexType>
                <attribute name="idref" type="IDREF" use="required"/>
                <!-- Event ID -->
```

```
                    </complexType>
                </element>
            </sequence>
            <attribute name="id" type="ID" use="required"/>
            <!-- PreservationObject ID -->
            <attribute name="hasParent" type="IDREF" use="optional"/>
        </complexType>

        <complexType name="IntellectualEntityType">
            <complexContent>
                <extension base="pp2:PreservationObjectType"></extension>
            </complexContent>
        </complexType>

        <complexType name="ComponentType">
            <complexContent>
                <extension base="pp2:PreservationObjectType">
                    <!-- Can one restrict the inherited hasParent to IntellectualEntity or Component IDs? -->
                    <sequence>
                        <element name="ComponentType" type="string"/>
                        <element name="hasRepresentation" minOccurs="0" maxOccurs="unbounded">
                            <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
                                <!-- Representation ID -->
                            </complexType>
                        </element>
                    </sequence>
                </extension>

            </complexContent>
        </complexType>
```

```
<complexType name="RepresentationType">
        <complexContent>
                <extension base="pp2:PreservationObjectType">
                        <sequence>
                                <element name="rendersComponent" minOccurs="1" maxOccurs="unbounded">
                                        <complexType>
                                                <attribute name="idref" type="IDREF" use="required"/>
                                                <!-- Component ID -->
                                        </complexType>
                                </element>
                                <element name="hasRepresentationBitstream" minOccurs="1" maxOccurs="unbounded">
                                        <complexType>
                                                <attribute name="idref" type="IDREF" use="required"/>
                                                <!-- RepresentationBitstream ID -->
                                        </complexType>
                                </element>
                                <element name="hasRepresentationBitstreamStructmap" minOccurs="1"
                                        maxOccurs="unbounded">
                                        <!-- T his should be something like a METS structmap ??? pop in xml-->
                                </element>
                        </sequence>
                </extension>
        </complexContent>
</complexType>

<complexType name="RepresentationBitstreamType">
        <sequence>
                <element name="hasRepresentation" maxOccurs="unbounded" minOccurs="1">
                        <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
                        <!-- Representation ID -->
                    </complexType>
                </element>
                <element name="implementedBy" minOccurs="0" maxOccurs="1">
                    <complexType>
                        <attribute name="idref" type="IDREF" use="required"/>
                        <!-- Bitstream ID -->
                        <!-- A link between Bitstream and RepresentationBitstream is mandatory in at least one direction -->
                    </complexType>
                </element>
            </sequence>
        </complexType>

        <complexType name="BitstreamType">
            <complexContent>
                <extension base="pp2:PreservationObjectType">
                    <sequence>
                        <element name="implements" minOccurs="0" maxOccurs="unbounded">
                            <!-- A link between Bitstream and RepresentationBitstream is mandatory in at least one direction -->
                            <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
                                <!-- RepresentationBitstream ID -->
                            </complexType>
                        </element>
                    </sequence>
                </extension>
            </complexContent>

        </complexType>

        <complexType name="BytestreamType">
```

```
		<complexContent>
			<extension base="pp2:BitstreamType"> </extension>
		</complexContent>
	</complexType>

	<complexType name="FileType">
		<complexContent>
			<extension base="pp2:BytestreamType"> </extension>
		</complexContent>
	</complexType>


	<complexType name="EnvironmentType">
		<sequence>
			<element name="environmentName" maxOccurs="unbounded" minOccurs="0" type="string"/>
			<element name="environmentDescription" maxOccurs="unbounded" minOccurs="0" type="string"/>
			<element name="environmentPurpose" maxOccurs="unbounded" minOccurs="0" type="string"/>
			<!-- creation, ingest, preservation, remote access, local access, migration, etc.. Can this be made restricted to an extensible
controlled vocabulary??? -->
			<element name="environmentFunction" maxOccurs="unbounded" minOccurs="0" type="string"/>
			<!-- rendering, editing, executing, printing, etc.. Can this be made restricted to an extensible controlled vocabulary??? -->
			<element name="environmentIntention" maxOccurs="unbounded" minOccurs="0" type="string"/>
			<!-- necessary, recommended, acceptable, etc.. Can this be made restricted to an extensible controlled vocabulary??? -->
			<element name="hasParentEnvironment" maxOccurs="unbounded" minOccurs="0">
				<complexType>
					<attribute name="idref" type="IDREF" use="required"/>
					<!-- Environment ID -->
				</complexType>
			</element>
			<element name="hasPreservationObject" maxOccurs="unbounded" minOccurs="0">
				<complexType>
```

```
                                <attribute name="idref" type="IDREF" use="required"/>
                                <!-- PreservationObject ID -->
                        </complexType>
                </element>
                <element name="hasCharacteristic" maxOccurs="unbounded" minOccurs="0">
                        <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
                                <!-- Characteristic ID -->
                        </complexType>
                </element>
                <element name="hasRisk" maxOccurs="unbounded" minOccurs="0">
                        <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
                                <!-- PreservationRisk ID -->
                        </complexType>
                </element>
                <element name="Event" maxOccurs="unbounded" minOccurs="0">
                        <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
                                <!-- Event ID -->
                        </complexType>
                </element>
        </sequence>
        <attribute name="id" type="ID" use="required"/>
        <!-- Environment ID -->
</complexType>

<complexType name="PreservationRiskType">
        <sequence>
                <element name="riskName" maxOccurs="unbounded" minOccurs="0" type="string"/>
                <element name="riskDescription" maxOccurs="unbounded" minOccurs="0" type="string"/>
                <element name="associatedWith" maxOccurs="unbounded" minOccurs="0">
                    <complexType>
                        <attribute name="idref" type="IDREF" use="required"/>
                        <!-- PreservationObject or Environment ID -->
                    </complexType>
                </element>
                <element name="hassEvent" maxOccurs="unbounded" minOccurs="0">
                    <complexType>
                        <attribute name="idref" type="IDREF" use="required"/>
                        <!-- Event ID -->
                    </complexType>
                </element>
            </sequence>
            <attribute name="id" type="ID" use="required"/>
            <!-- PreservationRisk ID -->
            <attribute name="hasRequirement" type="IDREF" use="required"/>
            <!-- RiskSpecifyiingRequirementID -->
        </complexType>

        <complexType name="NewVersionRiskType">
            <complexContent>
                <extension base="pp2:PreservationRiskType"> </extension>
            </complexContent>
        </complexType>

        <complexType name="DeteriorationOrLossRiskType">
            <complexContent>
                <extension base="pp2:PreservationRiskType"> </extension>
            </complexContent>
        </complexType>
```

```
<complexType name="LackingSupportRiskType">
        <complexContent>
                <extension base="pp2:PreservationRiskType"> </extension>
        </complexContent>
</complexType>

<complexType name="ProprietaryRiskType">
        <complexContent>
                <extension base="pp2:PreservationRiskType"> </extension>
        </complexContent>
</complexType>

<complexType name="UnmanagedGrowthRiskType">
        <complexContent>
                <extension base="pp2:PreservationRiskType"> </extension>
        </complexContent>
</complexType>


<complexType name="PreservationActionType">
        <sequence>
                <element name="actionName" maxOccurs="unbounded" minOccurs="0" type="string"/>
                <element name="actionDescription" maxOccurs="unbounded" minOccurs="0" type="string"/>
                <element name="hasParentPreservationAction" maxOccurs="unbounded" minOccurs="0">
                        <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
                                <!-- PreservationAction ID -->
                        </complexType>
                </element>
                <element name="Type" minOccurs="0" maxOccurs="unbounded" type="string"/>
                <element name="hasRisk" minOccurs="0" maxOccurs="unbounded">
```

```
          <complexType>
                    <attribute name="idref" type="IDREF" use="required"/>
                    <!-- PreservationRisk ID -->
          </complexType>
</element>
<!-- At least one input or output PreservationObject or Environment needs to exist -->
<element name="InputPreservationObject" maxOccurs="1" minOccurs="0">
          <complexType>
                    <attribute name="idref" type="IDREF" use="required"/>
                    <!-- PreservationObject ID -->
          </complexType>
</element>
<element name="OutputPreservationObject" maxOccurs="1" minOccurs="0">
          <complexType>
                    <attribute name="idref" type="IDREF" use="required"/>
                    <!-- PreservationObject ID -->
          </complexType>
</element>
<element name="InputEnvironment" maxOccurs="1" minOccurs="0">
          <complexType>
                    <attribute name="idref" type="IDREF" use="required"/>
                    <!-- Environment ID -->
          </complexType>
</element>
<element name="OutputEnvironment" maxOccurs="1" minOccurs="0">
          <complexType>
                    <attribute name="idref" type="IDREF" use="required"/>
                    <!--  Environment ID -->
          </complexType>
</element>
<element name="Environment" maxOccurs="1" minOccurs="1">
```

```
				<complexType>
					<attribute name="idref" type="IDREF" use="required"/>
					<!--  Environment ID -->
				</complexType>
			</element>
			<element name="hasCharacteristic" minOccurs="0" maxOccurs="unbounded">
				<complexType>
					<attribute name="idref" type="IDREF" use="required"/>
					<!-- Characteristic ID -->
				</complexType>
			</element>
			<element name="hasEventOutcome" minOccurs="0" maxOccurs="1">
				<complexType>
					<sequence>
						<element name="hasRequirementSet" minOccurs="0" maxOccurs="unbounded">
							<complexType>
								<attribute name="idref" type="IDREF" use="required"/>
								<!-- PreservationGuidingRequirementsSet -->
							</complexType>
						</element>
						<element name="DegreeOfCompliance" minOccurs="0" maxOccurs="unbounded">
							<complexType>
								<sequence>
									<element name="associatedWith" minOccurs="1" maxOccurs="unbounded"
type="IDREF"/>

									<!-- ???  a vector of parameters which are either PreservationObject or
Environment IDs -->
								</sequence>
								<attribute name="hasRequirement" type="IDREF" use="required"/>
								<attribute name="hasMeasure" type="IDREF" use="required"/>
								<!-- The definition of measure is out-of-scope for tis model -->
```

```
                                        <attribute name="hasOutcome" type="string" use="required"/>
                                    </complexType>
                                </element>
                            </sequence>
                        </complexType>
                    </element>
                </sequence>
                <attribute name="id" type="ID" use="required"/>
                <!-- PreservationAction ID -->
            </complexType>

            <complexType name="PropertyType">
                <sequence>
                    <element name="propertyName" maxOccurs="unbounded" minOccurs="0" type="string"/>
                    <element name="propertyDescription" maxOccurs="unbounded" minOccurs="0" type="string"/>
                    <element name="appliesTo" maxOccurs="unbounded" minOccurs="1">
                        <complexType>
                            <attribute name="idref" type="IDREF" use="required"/>
                            <!-- PreservationObject or Environment or PreservationAction ID -->
                        </complexType>
                    </element>
                    <element name="hasRange" maxOccurs="unbounded" minOccurs="0">
                        <complexType>
                            <attribute name="hasUnit" type="IDREF" use="optional"/>
                            <!-- Unit ID -->
                            <attribute name="hasDataConstraint" type="string" use="required"/>
                            <attribute name="isDefault" type="string" use="required"/>
                            <!-- yes, no -->
                            <attribute name="hasDefaultValue" type="string" use="required"/>
                            <!-- This could be any value, number, string, etc.-->
                        </complexType>
                </element>
                <element name="hasValueOrigin" maxOccurs="unbounded" minOccurs="0">
                        <complexType>
                                <attribute name="hasValueOriginID" type="string" use="required"/>
                                <!-- ValueOrigin ID -->
                                <attribute name="isDefault" type="IDREF" use="optional"/>
                                <!-- yes, no -->
                        </complexType>
                </element>
                <element name="hasRelationship" minOccurs="0" maxOccurs="unbounded">
                        <complexType>
                                <attribute name="hasRelatedProperty" type="IDREF" use="required"/>
                                <!-- Property ID -->
                                <attribute name="hasRelationshipType" type="string" use="required"/>
                        </complexType>
                </element>
                <element name="hasEvent" minOccurs="0" maxOccurs="unbounded">
                        <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
                                <!-- Event ID -->
                        </complexType>
                </element>
        </sequence>
        <attribute name="id" type="ID" use="required"/>
        <!-- Property ID -->
</complexType>

<complexType name="ValueOriginType">
        <sequence>
                <element name="valueOriginName" minOccurs="0" maxOccurs="unbounded" type="string"/>
                <element name="valueOriginDescription" minOccurs="0" maxOccurs="unbounded" type="string"/>
                <element name="hasSource" minOccurs="0" maxOccurs="unbounded" type="string"/>
                <element name="hasTargetUnit" minOccurs="0" maxOccurs="unbounded" type="IDREF"/>
                <!-- Unit ID -->
                <element name="hasTechnique" minOccurs="0" maxOccurs="unbounded" type="string"/>
                <element name="hasAgent" minOccurs="0" maxOccurs="unbounded" type="IDREF"/>
                <!-- Agent ID -->
                <element name="hasTrigger" minOccurs="0" maxOccurs="unbounded" type="string"/>
            </sequence>
            <attribute name="id" type="ID" use="required"/>
            <!-- ValueOrigin ID -->
        </complexType>

        <complexType name="UnitType">
            <sequence>
                <element name="unitName" minOccurs="0" maxOccurs="unbounded" type="string"/>
                <element name="unitDescription" minOccurs="0" maxOccurs="unbounded" type="string"/>
                <element name="hasDataConstraint" minOccurs="1" maxOccurs="1" type="string"/>
                <element name="hasConversion" minOccurs="0" maxOccurs="unbounded">
                    <complexType>
                        <sequence>
                            <element name="hasTechnique" minOccurs="0" maxOccurs="unbounded"
                                type="string"/>
                            <element name="hasAgent" minOccurs="0" maxOccurs="unbounded" type="IDREF"/>
                            <!-- Agent ID -->
                        </sequence>
                        <attribute name="hasSource" type="IDREF" use="required"/>
                        <!-- Unit ID -->
                    </complexType>
                </element>
            </sequence>
            <attribute name="id" type="ID" use="required"/>
        <!-- Unit ID -->
    </complexType>

    <complexType name="CharacteristicType">
        <sequence>
            <element name="associatedWith" maxOccurs="unbounded" minOccurs="1">
                <complexType>
                    <attribute name="idref" type="IDREF" use="required"/>
                    <!-- PreservationObject or Environment  or PreservationAction ID -->
                </complexType>
            </element>
            <!-- a vector of parameters which are either PreservationObject or Environment or PreservationAction IDs -->
            <element name="Annotation" minOccurs="0" maxOccurs="1">
                <complexType>
                    <attribute name="hasUnit" type="IDREF" use="optional"/>
                    <!-- Unit ID -->
                    <attribute name="hasValueOrigin" type="IDREF" use="optional"/>
                    <!-- ValueOrigin ID -->
                    <attribute name="hasTechnique" type="string" use="optional"/>
                    <attribute name="hasSource" type="string" use="optional"/>
                    <attribute name="hasAgent" type="IDREF" use="optional"/>
                    <!-- Agent ID -->
                </complexType>
            </element>
        </sequence>
        <attribute name="id" type="ID" use="required"/>
        <!-- Characteristic ID -->
        <attribute name="hasProperty" type="IDREF" use="required"/>
        <!-- Property ID -->
        <attribute name="onDemandP" type="string" use="optional"/>
        <!-- yes, no -->
```

```
          <attribute name="hasValue" type="string" use="optional"/>
          <attribute name="hasCreationEvent" type="IDREF" use="optional"/>
          <!-- Event ID -->
     </complexType>

     <complexType name="RequirementType">
          <sequence>
               <element name="requirementName" type="string" maxOccurs="unbounded" minOccurs="0"/>
               <element name="requirementDescription" type="string" maxOccurs="unbounded" minOccurs="0"/>
               <element name="hasRequirementsSet" type="IDREF" maxOccurs="unbounded" minOccurs="0"/>
               <!-- PreservationGuidingRequirementsSet ID -->
               <element name="hasStakeholder" type="IDREF" maxOccurs="unbounded" minOccurs="0"/>
               <!-- Agent ID -->
               <element name="requirementApplicability" maxOccurs="1" minOccurs="0">
                    <complexType>
                         <attribute name="startDate" type="date" use="optional"/>
                         <attribute name="endDate" type="date" use="optional"/>
                    </complexType>
               </element>
               <element name="Specification" maxOccurs="1" minOccurs="1">
                    <complexType>
                    <sequence>
                         <element name="context" minOccurs="0" maxOccurs="unbounded" type="string"/>
                         <element name="precondition" minOccurs="0" maxOccurs="unbounded" type="string"/>
                         <element name="postcondition" minOccurs="0" maxOccurs="unbounded" type="string"/>
                         <!-- The postcondition includes the tolerance factor -->
                         <!-- No necessity to invent an XML constraint language at this juncture. Type "string" allows for versatile use.
For an example UML constraint language see OCL (the Object Constraint Language) -->
                    </sequence>
                    </complexType>
               </element>
```

```
            <element name="requirementImportanceFactor" type="string" maxOccurs="1" minOccurs="0"/>
            <element name="hasEvent" type="IDREF" maxOccurs="unbounded" minOccurs="0"/>
            <!-- Event ID -->
        </sequence>
        <attribute name="id" type="ID" use="required"/>
        <!-- Requirement ID -->
    </complexType>

    <complexType name="NonPreservationRequirementType">
        <complexContent>
            <extension base="pp2:RequirementType"> </extension>
        </complexContent>
    </complexType>

    <complexType name="PreservationRequirementType">
        <complexContent>
            <extension base="pp2:RequirementType"> </extension>
        </complexContent>
    </complexType>

    <complexType name="RiskSpecifyingRequirementType">
        <complexContent>
            <extension base="pp2:PreservationRequirementType"> </extension>
        </complexContent>
    </complexType>

    <complexType name="PreservationObjectSelectingRequirementType">
        <complexContent>
            <extension base="pp2:RiskSpecifyingRequirementType"> </extension>
        </complexContent>
    </complexType>
```

```
<complexType name="PreservationProcessGuidingRequirementType">
        <complexContent>
                <extension base="pp2:PreservationRequirementType"> </extension>
        </complexContent>
</complexType>

<complexType name="PreservationInfrastructureRequirementType">
        <complexContent>
                <extension base="pp2:PreservationProcessGuidingRequirementType"> </extension>
        </complexContent>
</complexType>

<complexType name="PreservationGuidingRequirementType">
        <complexContent>
                <extension base="pp2:PreservationRequirementType"> </extension>
        </complexContent>
</complexType>

<complexType name="ActionDefiningRequirementType">
        <complexContent>
                <extension base="pp2:PreservationGuidingRequirementType"> </extension>
        </complexContent>
</complexType>

<complexType name="SignificantCharacteristicRequirementType">
        <complexContent>
                <extension base="pp2:PreservationGuidingRequirementType"> </extension>
        </complexContent>
</complexType>
```

```
<complexType name="RiskActionMatchingRequirementType">
        <complexContent>
                <extension base="pp2:PreservationGuidingRequirementType"> </extension>
        </complexContent>
</complexType>

<element name="ActionDefiningRequirement" type="pp2:ActionDefiningRequirementType"/>
<element name="Bitstream" type="pp2:BitstreamType"/>
<element name="Bytestream" type="pp2:BytestreamType"/>
<element name="Characteristic" type="pp2:CharacteristicType"/>
<element name="Component" type="pp2:ComponentType"/>
<element name="DeteriorationOrLossRisk" type="pp2:DeteriorationOrLossRiskType"/>
<element name="Environment" type="pp2:EnvironmentType"/>
<element name="File" type="pp2:FileType"/>
<element name="IntellectualEntity" type="pp2:IntellectualEntityType"/>
<element name="LackingSupportRisk" type="pp2:LackingSupportRiskType"/>
<element name="NewVersionRisk" type="pp2:NewVersionRiskType"/>
<element name="NonPreservationRequirement" type="pp2:NonPreservationRequirementType"/>
<element name="PreservationAction" type="pp2:PreservationActionType"/>
<element name="PreservationGuidingRequirement" type="pp2:PreservationGuidingRequirementType"/>
<element name="PreservationGuidingRequirementsSet"
        type="pp2:PreservationGuidingRequirementsSetType"/>
<element name="PreservationInfrastructureRequirement"
        type="pp2:PreservationInfrastructureRequirementType"/>
<element name="PreservationObject" type="pp2:PreservationObjectType"/>
<element name="PreservationObjectSelectingRequirement"
        type="pp2:PreservationObjectSelectingRequirementType"/>
<element name="PreservationProcessGuidingRequirement"
        type="pp2:PreservationProcessGuidingRequirementType"/>
<element name="PreservationRequirement" type="pp2:PreservationRequirementType"/>
<element name="PreservationRisk" type="pp2:PreservationRiskType"/>
```

```
        <element name="Property" type="pp2:PropertyType"/>
        <element name="ProprietaryRisk" type="pp2:ProprietaryRiskType"/>
        <element name="RepresentationBitstream" type="pp2:RepresentationBitstreamType"/>
        <element name="Representation" type="pp2:RepresentationType"/>
        <element name="Requirement" type="pp2:RequirementType"/>
        <element name="RiskActionMatchingRequirement" type="pp2:RiskActionMatchingRequirementType"/>
        <element name="RiskSpecifyingRequirement" type="pp2:RiskSpecifyingRequirementType"/>
        <element name="SignificantCharacteristicRequirement"
                type="pp2:SignificantCharacteristicRequirementType"/>
        <element name="Unit" type="pp2:UnitType"/>
        <element name="UnmanagedGrowthRisk" type="pp2:UnmanagedGrowthRiskType"/>
        <element name="ValueOrigin" type="pp2:ValueOriginType"/>

</schema>
```

6.2 Changes to the Model

6.2.1 Changes to the Conceptual Model since Report PP2-D2

Figure 19 compares the conceptual model from the draft in report [PP2-D2 2008] and in this report.

- We removed the *EnvironmentComponent* concept from the model and replaced it with a recursive *Environment* concept. This is a semantically equivalent but simpler representation.
- We removed the *PreservationWorkflow* concept from the model and replaced it with a recursive *PreservationAction* concept. This is a semantically equivalent but simpler representation. There is an aggregate, but not a BPEL containment, as was initially suggested. This latter change is due to a Planets-wide decision to not express workflows through BPEL constraints.
- We renamed the concept *PreservationGuidingDocument* to *PreservationGuidingRequirementsSet* to capture the notion that *Requirements* can be captured by means other than in documents.
- We made the *PreservationObject* and *Environment* concepts separate concepts, rather than having the *PreservationObject* be part of its own *Environment*. The latter representation was confusing. It would have been useful if a *PreservationObject* could have different *Characteristics* in its different *Environments*. But we have not observed this to be the case. As a consequence, now, all three concepts *PreservationObject, PreservationAction, and Environment* can have *Characteristics*, rather than just *PreservationAction, and Environment.*

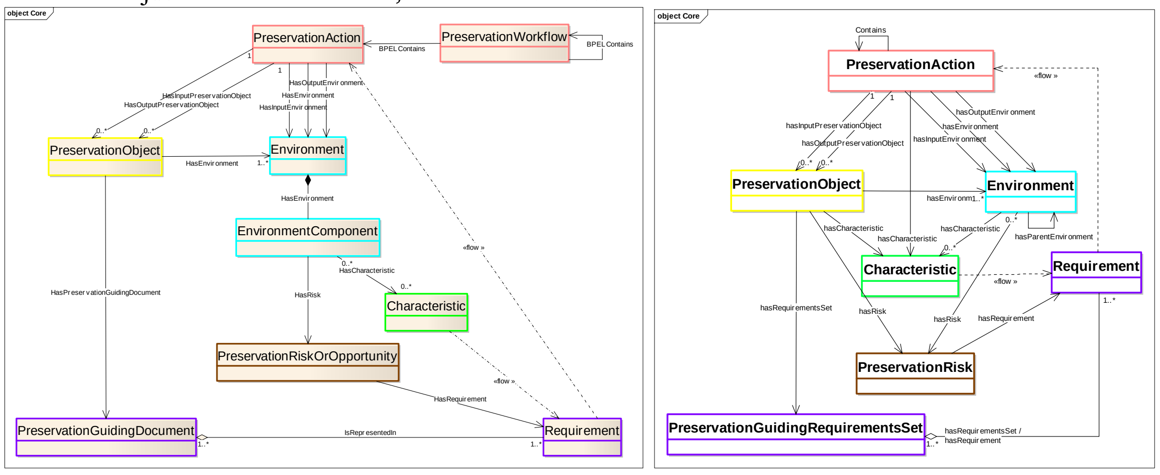

Figure 19: PP2-D2 and PP2-D3 Conceptual Models Compared

Figure 20 compares the PreservationObject subclasses from the draft in report [PP2-D2 2008] and in this report.

- For the subclasses of *PreservationObject* we made the following changes:
  - We renamed the concept *Manifestation* to *Representation*, which is the accepted terminology in PREMIS [PREMIS 2008] and avoids conflicts with the FRBR terminology [FRBR 1998].
  - We renamed the concept *ManifestationFile* to *RepresentationBitstream* since the *Representations* of *Components* are not necessarily captured in files or even bytestreams but might be a set of bitstreams which can be extracted from a set of files. Similarly, we generalized *Bytestreams* to *Bitstreams* in the model in order to be able to capture the most general case.
  - We removed the subclasses *Collection*, *DeliverableUnit* and *Expression* and replaced them with one concept, *IntellectualEntity*. All of these concepts had captured logical *PreservationObjects* but they had expressed hard-wired local institutional preferences. The concept *IntellectualEntity* can be instantiated to any local framework for logical *PreservationObjects* (such as *Fonds* and *Series* for archives, *Work* and *Expression* to capture useful FRBR [FRBR 1998] distinctions, or *Collection* and *SubCollection* to capture organisational structures). Additionally this change better aligns our model with PREMIS [PREMIS 2008] which uses the same terminology.

- We removed the *hasRepresentation* link from *IntellectualEntity* to *Representation*. This might initially be unintuitive but is justified as follows. A *Representation* is a collection of all *Bitstreams* that are needed to create one rendition of an *IntellectualEntity* or of a part of it, a *Component*. Since the *IntellectualEntity* can be seen as the largest possible *Component* of itself it is sufficient to havse a *hasRepresentation* link from *Component* to *Representation*. This unifies the treatment of *Representations* across implementations.

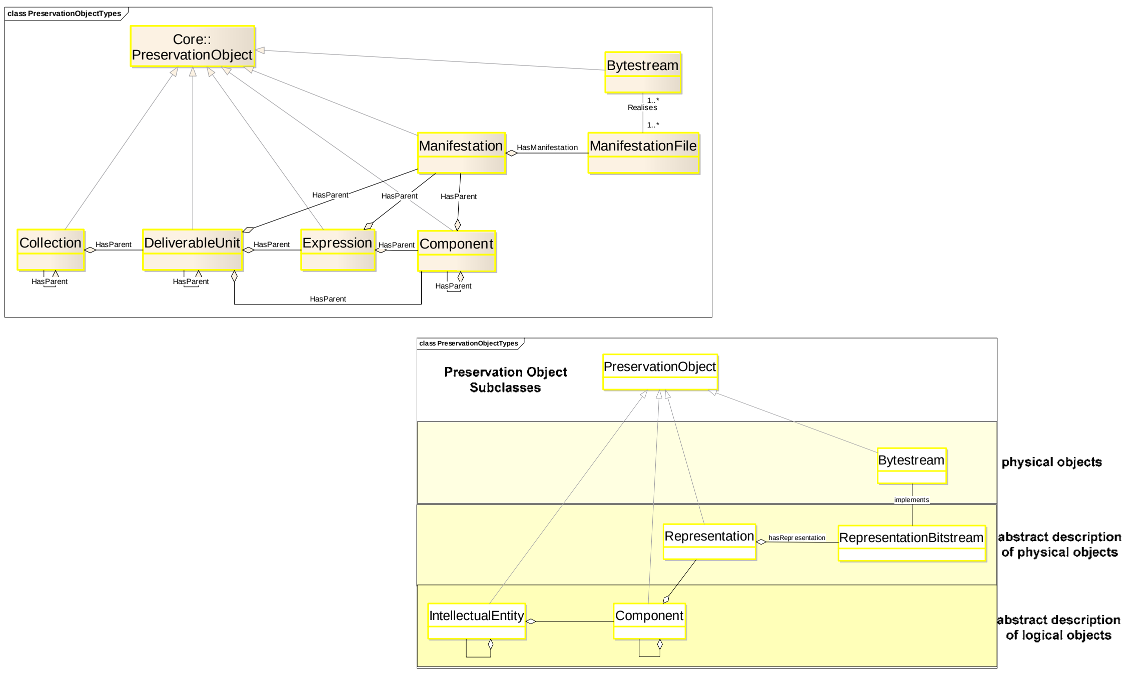

Figure 20: PP2-D2 and PP2-D3 PreservationObject Subclasses Compared

Figure 17 shows the new tree of the *PreservationRequirement* subclasses.

- In order to make explicit that a *PreservationAction* has to match the *PreservationRisk* which it is mitigating we added a *RiskActionMatchingRequirement* as a subclass of *PreservationGuidingRequirement*. This type of *Requirement* can capture which *PreservationServices* are considered acceptable when a certain *RiskSpecifyingRequirement* has been violated. The *PreservationAction* is an execution of an acceptable *PreservationService* to mitigating the *PreservationRisk* that results from the violation of the *RiskSpecifyingRequirement.*

6.2.2 Changes to the Data Dictionary since Report PP2-D2

There has been a large number of changes that resulted from our alignment effort. Not all of them can be explicitly listed here. Some of them are listed here to exemplify the changes.

- All concepts now have uniform treatment for identifiers, names, and descriptions. All *name* elements are made repeatable to allow for synonyms.
- We removed all elements that capture type information (the *Class* of the *Instance* under consideration). This is unnecessary implementation detail.
- We eliminated information about institutions, except for the stratml:Organization information. Any more institution specific information is out of scope.
- For *Representations*, we added some missing relationships: *rendersComponent* as the unique identifier of the *Component* for which the *Representation* serves as physical embodiment*; hasRepresentationBitstream* as the unique identifier of the *Bitstreams* that make up the *Representation*; and *hasRepresentationBitstreamStructmap* to enable us to capture logical and physical structural relationships between the *RepresentationBitstreams* that make up a *Representation* - similar to a METS [METS] structmap.
- In addition to turning *RepresentationFiles* into *RepresentationBitstreams*, as discussed above, we added a relationship from *RepresentationBitstream* to *Bitstream* which allows the specification of the location of the *RepresentationBitstream* within any larger *Bitstream* that contains it.

- For *Environments* we split *Role* into *Functions* and *Purpose*. We consolidated concepts that had previously been distributed over *EnvironmentComponents* and *Environments*.
- Since *PreservationAction* is an *Event* we added general *Event* information, such as *Date* and *hasEventOutcome*. A *PreservationAction* can record under what set of *Requirements* the *PreservationAction* was executed. It can store to what degree the *Requirements* have been satisfied for different *Components*. Furthermore we added a link to record the *PreservationAction's* own *Characteristics*.
- We renamed the element *hasObjectType* in the *Property* concept to *appliesTo* and *hasObjectType* in *Characeristics*, *PreservationRisk* and *PreservationActions* to *associatedWith* in order to align with word usage concerning types, subclasses, and applicability.
- We explicitly allow for *Properties* that relate multiple *Entities*. *appliesTo* has, therefore, been redefined as vector datastructure. Consequently, the *associatedWith* element of *Characteristics*, *PreservationRisk* and *PreservationAction* also has to be a vector.
- We renamed *ValueOptions* to *ValueOrigin* since this terminology captures the intended meaning better.
- We added the element *hasDefaultValue* and *isDefault* to the *Property* concept.
- We took *Unit* definitions and *ValueOrigin* out of *Properties* to show that they are independent re-usable concepts.
- We added the element *hasAgent* to the *ValueOrigin* concept. It expresses which type of agent can determine the *Value* for this *Property*. For automatically derived *Values* this is the software tool and version; for manually assigned *Values* it is the person role. There may be multiple possible agent types.
- We changed the data constraint for the element *hasProperty* of the *Characteristic* concept to be an identifier rather than extensible vocabulary.

6.3 Application in and Alignment against Other Work

In order to align the model with other work

- we presented our model in discussion and through publication [PP2-PresGuid 2008]. On this basis we discussed where it might break down in other applications, or where it exceeds the local usage.
- we shared a workshop with all other PP work-packages to discuss the fit of our approaches. Results from all work-packages went into creating a consistent preservation planning process model presented in PP7.
- we represented our requirements knowledgebase in the PP4 mind-map trees that are in use in work-package PP4.

The changes presented in the previous Appendix resulted from these activities.

In the following we list a brief assessment of the relationship of the PP2 model to other work. It will be necessary to refer to the actual conceptual model to appreciate this comparison. This description was accurate at the time the validation exercises took place. Models in use in other work-packages may have changed in the meantime.

Planets Work-Packages

Overall observations from the alignment and validation effort are:

- Other work-packages use subsets of the PP2 conceptual model.
    - They use only a subset of the concepts: This is to be expected, since every one of them addresses only a part of the whole preservation process.
    - For the concepts they do use, they use only a subset of the elements and attributes available in the data dictionary: That again is to be expected, since some are basic initial implementations that will grow in scope over time, while for others there is simply no need for all the features for their defined goals.

- o In either case, there was, however, no conflict with the PP2 data model and every element or attribute of the model was motivated by at least one requirement resulting from our analysis.

- The TB work-packages mostly avoid semantic interpretation of the objects they are processing. They simply receive input objects from other work-packages, process the information, and pass output for evaluation to other work-packages. The focus is on tool implementation. This means that they can stay semantically largely agnostic. Their simple property / value realisation is trivially consistent with the PP2 conceptual model.
- Some work-packages (PP4, PP6 recommender system) use a propositional reasoning system which is less expressive than the parameterized object-property-value model used in PP2. This means that concepts, elements and attributes of the PP4 systems can be mapped onto the PP2 model, but not necessarily the other way round. The PP2 analysis of existing requirements in policy and strategy documents was, however, useful for creating PP4 models.
- Work-package PP3 represents a qualitative analysis which is not based on explicit conceptual modelling. A match between concepts is therefore not generally possible. However, all PP/3 properties that are within the scope of digital preservation, are fully compatible with the PP/2 model and can be represented in the model.
- Work-package PP7 created a preservation planning process model which conceptually fits seamlessly with the PP2 model.
- Even though the IF8 Core Conceptual Model and the PP2 model have been partially co-developed there are some differences remaining. The Core Model originated in a system implementation. It therefore contains implementation details that are too specific for our general purpose model. On the other hand the Core Model has a smaller scope: Concepts such as *PreservationRisk*, *Requirement*, and *PreservationService* are missing. *Environments* are secondary concepts. Some of the differences evolved over the last year as the lessons-learnt in the alignment exercises prompted the need for change in the PP2 model. One difference, for example is that the Core Model maps Bytestreams and Manifestations (PP2: *Representations*) to OAIS data objects, and it maps the Core Model's equivalents of *IntellectualEntities* and *Components* to OAIS information objects. Per OAIS definition information objects are data objects together with their representation information. In the PP2 model, however, *IntellectualEntities* and *Components* may have representation information (e.g. *Environment* information) of their own. The PP2 model makes it possible for *PreservationObjects* on all levels to have *Environments* of their own. This is sensible, since, for example, *Files* have different *Environments* from the *Representations* they belong to.
- Work-packages which are developing registries that capture definitions of *Properties* that apply to given file formats (PC, PP5, TB) are currently being developed and not yet stable enough to definitively determine how well they align with the PP2 model. Currently their attributes are subsets of those described in the PP2 model. Most of the design of the PP2 *Property* elements and attributes were inspired through discussions with PC work-packages.

PREMIS

As in the PP2 model the PREMIS data model has representations, files (PP2:*Bitstreams*) and bitstreams (PP2:*Components*) as subclasses of objects (PP2:*PreservationObjects*). The PREMIS bitstream concept can be used like the *Component* concept in the PP2 model, except that it is restricted to a bitstream within one file. Planets *Components* can consist of sets of *Bitstreams* which can span several files (See Section 4.5.4 for a discussion.)

The PREMIS data model does not consider *IntellectualEntities* a subclass of *PreservationObjects*. *IntellectualEntities* in PREMIS are currently not yet fleshed out and the PREMIS Editorial Committee is currently considering how this should be done.

Significant properties (PP2: *SignificantCharacteristics*) in the PREMIS model exist but are not as fleshed out, as they are in the PP2 model. There are, for example, no tolerance or importance factors. At the moment they can only be attached to one *PreservationObject* at a time. There is at the moment no capability to express *SignificantCharacteristics* or other *Requirements* which express constraints on *Environments* or combinations of *Environments* and *PreservationObjects*.

Environments in PREMIS are subordinate to objects, rather than taking on a role equivalent to objects, as is the case in PP2. The PP2 approach makes it more natural to model *PreservationActions* that directly impact *Environments*, such as data carrier refresh or emulation as easily as *PreservationActions* that directly impact

PreservationObjects, say migration. While describing those actions is possible with the PREMIS model, it comes less natural.

While specific *Properties* are modelled in some depth within PREMIS, PREMIS does not have a generic, rich specification of *Properties* that takes account of *ValueOrigins* and does not offer a meta-level on which to describe the properties of *Properties* and their relationships to other *Properties*. This is not in scope for PREMIS.

PreservationActions and *PreservationRisks* are outside the scope of the PREMIS data model.

The *Event*, *Agent* and *Rights* concepts of PREMIS are assumed to exist in PP2 but are not modelled in detail in the PP2 data dictionary.

OAIS

The PP2 model has been aligned carefully with the PP7 work-package which describes the relationship between the Planets and the OAIS preservation planning processes. In developing this comparison PP7 has also indirectly described the relationship of PP2 and OAIS.

In addition, it is worth noting that OAIS does not have a concept of expressing *Risks* or *Requirements* which guide digital preservation processes. While there is a notion of significant properties, they are, as in PREMIS, restricted to one object and do not specify characteristics of future objects derived from current objects.

OAIS does not explicitly model the relationships between subsequent *Representations* of an *IntellectualEntity* and does not model *Requirements* that would guide the process of deriving them.

OAIS also does not distinguish between physical and logical objects. OAIS data objects are probably closest to PP2 *Bitstreams* and *Representations*. Together with their representation information they create an information object. This is, however not the same thing as an *IntellectualEntity* or *Component* in PP2. *IntellectualEntities* and *Components* can have representation information (e.g. their *Environments*) of their own and can have no direct data object associated with them (such as a journal title or a collection).

OAIS has no notion of different *Representations* of the same information object other than the ability to model information objects' context.

6.4 StratML

Strategy Markup Language (StratML) is a basic conceptual model for describing the essential contents of a strategy document. It is envisioned as an ISO standardized XML schema and vocabulary for US Federal agency strategic plans that is aligned with the Federal Enterprise Architecture, government policy, and leverages existing standards (based on http://www.xml.gov/presentations/gpo/stratml20060118.ppt). We are borrowing most of StratML's basic elements to describe the non-*Requirement* parts of *PreservationGuidingRequirementsSet*s.

Some top-level elements in StratML are as follows:

- *Submitter*: The person submitting the plan.
- *Source*: The Web address (URL) for the authoritative source of this document
- *Organization*: The legal or logical entity to which the report applies.
- *Vision*: Vision statements are distinguished from goals in that they are the focus of constant pursuit but can never be satisfied in the sense of being met or completed.
  A concise and inspirational description of a state the organization will strive to approach over a relatively long span of years but which can ultimately never be fully achieved.
- *Mission*: Mission Statement. A brief description of the basic purpose of the organization. An agency's goals should flow from the mission statement.
- *Value*: A principle that is important and helps to define the essential character of the organization.
- *Goal*: General Goal.
  A relatively broad statement of intended results to be achieved over more than one resource allocation and performance measurement cycle.
  Goals define a purpose and direction and take all stakeholders and perceived present and future needs into account. Goals must be capable of being effectively pursued with measurable results over more than one budgetary execution cycle but within the reasonably foreseeable future. Goals should be objective, quantifiable, measurable, and defined at the level to be achieved by a program activity.
  Supports Mission
- *Objective*: Performance Goal.
  A target level of results expressed in units against which achievement is to be measured within a single resource allocation and performance execution cycle.
  Supports Goal.
  Objectives are measurable subsets of goals to be achieved within a given time period with available resources. Objectives provide the day-to-day support for achieving goals.

Submitter, source, organization, vision, mission and value are to be used as in StratML. They can be directly used for automatic preservation planning.

The schema definition can be found in http://www.xml.gov/stratml/StrategicPlan.xsd.

Within our model, these concepts are used in the following way:

- *StratML:Value*, which expresses an (ethical) value of a stakeholder, is different from the "*Planets:Value*", which expresses the *Value* of a *Characteristic* (= assigned or derived *Value*).
- A *StratML:objective* is roughly equivalent to a *Requirement* in Planets. In StratML, an objective is represented as a string. In order to support automated preservation planning, however, a machine-interpretable definition of the objective / *Requirement* is needed. This was developed above.

The other StratML elements provide values that can be simply looked up and used by preservation services.

Error! Reference source not found. shows an example snippet of a StratML document for the Boy Scouts of America.

```
<?xml version="1.0" encoding="UTF-8" ?>
<StrategicPlanCore StartDate="1/1/2006" EndDate="12/31/2010" Date="2007-11-27">
   <Submitter FirstName="Owen" LastName="Ambur" PhoneNumber="" EmailAddress="Owen.Ambur@verizon.net"/>
   <Source>http://www.scouting.org/media/strategy/45-016.pdf</Source>
   <Organization>
      <Name>Boy Scouts of America</Name>
      <Acronym>BSA</Acronym>
   </Organization>
   <Vision>The Boy Scouts of America will prepare every eligible youth in America to become a responsible, participating citizen and
leader who is guided by the Scout Oath and Law.</Vision>
   <Mission>The mission of the Boy Scouts of America is to prepare young people to make ethical and moral choices over their
lifetimes by instilling in them the values of the Scout Oath and Law.</Mission>
   <Goal>
      <SequenceIndicator>1</SequenceIndicator>
      <Name>Opportunity for Involvement</Name>
      <Description>Every Eligible Youth Has an Opportunity to Be Involved in a Quality Scouting
         Experience</Description>
      <Stakeholder />
      <Objective>
         <SequenceIndicator>1.1</SequenceIndicator>
         <Name>Market Share</Name>
         <Description>Increase market share and/or growth.</Description>
         <Stakeholder />
      </Objective>
      <Objective>
         <SequenceIndicator>1.2</SequenceIndicator>
         <Name>New Members</Name>
         <Description>Increase the number of new members.</Description>
         <Stakeholder />
      </Objective>
   </Goal>
</StrategicPlanCore>
```

Figure 21: An Example Snippet from http://xml.gov/stratml/BSAStratPlan.xml

7. Bibliography

[Chaudhri 1998]
Chaudhri, V., Farquhar, A., Fikes, R., Karp, P., Rice, J.:. OKBC: A programmatic foundation for knowledge base interoperability. In Proceedings of the 1998 National Conference on Artificial Intelligence . (1998)

[Core 2008]
Sharpe, R. (2009) PLANETS Data Model Overview, Planets Report IF8-D1
http://www.planets-project.eu/private/planets-ftp/WP_IF/if8/Data_Model_1._PLANETS_Data_Models_overview.doc

[DC]
Dublin Core Metadata Initiative (DCMI). Retrieved on May 13, 2009 from www.dublincore.org/

[FRBR 1998]
IFLA Study Group: Functional Requirements for Bibliographic Records.: (1998). Functional requirements for bibliographic records : final report.— München: K.G. Saur, 1998. — (UBCIM publications ; new series, vol. 19). — ISBN 3-598-11382-X. Retrieved on December 1, 2007 from
http://www.ifla.org/VII/s13/frbr/frbr.pdf

[Hockx 2008]
Hockx-Yu, H., Knight, G. (2008): *What to Preserve?: Significant Properties of Digital Objects. Report on the JISC/BL/DPC Workshop of April 7, 2008, British Library Conference Centre*. The International Journal of Digital Curation, Issue 1, Volume 3 | 2008 http://www.ijdc.net/./ijdc/article/view/70/70

[MARCXML]
MARC 21 XML Schema. Retrieved on May 13, 2009 from www.loc.gov/standards/marcxml/

[MODS]
Metadata Object Description Schema: MODS (Library of Congress). Retrieved on May 13, 2009 from www.loc.gov/standards/mods/

[METS]
Metadata Encoding and Transmission Standard (METS) Official Web Site.
http://www.loc.gov/standards/mets/

[NISO_MIX]
Metadata for Images in XML Standard (MIX). Retrieved on May 13, 2009 from www.loc.gov/standards/mix/

[NLM]
National Center for Biotechnology Information (NCBI) of the National Library of Medicine (NLM). Archiving and Interchange Tag Set. Retrieved on May 13, 2009 from http://dtd.nlm.nih.gov/

[OAIS 2002]
CCSDS. Reference Model for an Open Archival Information System (OAIS). CCSDS 650.0-B-1, Blue Book (the full ISO standard). January 2002. Retrieved on May 13, 2009 from
http://public.ccsds.org/publications/archive/650x0b1.pdf

[OCL 2003]
Warmer, A. and Kleppe, A. (2003). The Object Constraint Language. Getting Your Models Ready for MDA. Addison-Wesley Longman Publishing Co., Boston, MA, USA

[Plato 2008]
Becker, C., Kulovits, H., Rauber, A., Hofman, H. 2008. *Plato: A Service Oriented Decision Support System for Preservation Planning.* JCDL'08, Pittsburgh, Pennsylvania, USA.

[PP2-D2 2008]

Angela Dappert, Bart Ballaux, Michaela Mayr, Sara van Bussel. Report on policy and strategy models for libraries, archives and data centres. Planets external report PP2-D2. 24 June 2008. http://www.planets-project.eu/docs/reports/Planets_PP2_D2_ReportOnPolicyAndStrategyModelsM24_Ext.pdf

[PP2-PresGuid 2008]
Angela Dappert, Adam Farquhar (2008). *Modelling Organisational Goals to Guide Preservation*. iPRES 2008: The Fifth International Conference on Preservation of Digital Objects.
http://www.bl.uk/ipres2008/ipres2008-proceedings.pdf , page 5.

[PP2-PresServ 2009]
Angela Dappert, Adam Farquhar (2009). *Implementing Metadata which Guides Digital Preservation Services*. Appended as PP2-D3a

[PP2-SigChar 2009]
Angela Dappert, Adam Farquhar (2009). *Significance is in the Eye of the Stakeholder*. European Conference on Digital Libraries (ECDL) September/October 2009
http://planets-project.eu/docs/papers/Dappert_Significant_Characteristics_ECDL2009.pdf

[PREMIS 2008]
PREMIS Editorial Committee: PREMIS Data Dictionary for Preservation Metadata, Version 2. March 2008
http://www.loc.gov/standards/premis/v2/premis-2-0.pdf (2008)

[TEI]
TEI: Text Encoding Initiative Retrieved on May 13, 2009 from www.tei-c.org/

[TEXTMD]
Technical Metadata for Text (textMD). Retrieved on May 13, 2009 from www.loc.gov/standards/textMD

[Thaller 2008]
Manfred Thaller, Volker Heydegger, et al. (2008). *Significant Characteristics to Abstract Content: Long Term Preservation of Information*. European Conference on Digital Libraries (ECDL) 2008.

[Wilson 2007]
Wilson, A.: Significant Properties Report, InSPECT Work Package 2.2, Draft/Version 2. Retrieved on May 13, 2009 from http://www.significantproperties.org.uk/documents/wp22_significant_properties.pdf (2007)