# Describing Digital Object Environments in PREMIS

Angela Dappert
Digital Preservation Coalition
% The British Library
96 Euston Road
London, NW1 2DB, UK
angela@dpconline.org

Sébastien Peyrard
National Library of France
Bibliographic and Digital Information Department
Quai François-Mauriac
75706 Paris Cedex 13
sebastien.peyrard@bnf.fr

Janet Delve
The School of Creative Technologies
The University of Portsmouth
Winston Churchill Avenue
Portsmouth, PO1 2DJ
janet.delve@port.ac.uk

Carol C.H Chou
Florida Digital Archive
Florida Virtual Campus
5830 NW 39th Ave.
Gainesville, FL 32606
U.S.A.
cchou@ufl.edu

## ABSTRACT

"Digital preservation metadata" is the information that is needed in order to preserve digital objects successfully in the long-term so that they can be deployed in some form in the future. A digital object is not usable without a computing environment in which it can be rendered or executed. Because of this, information that describes the sufficient components of the digital object's computing environment has to be part of its preservation metadata. Although there are semantic units for recording environment information in PREMIS 2, these have rarely, if ever, been used. Prompted by increasing interest in the description of computing environments, this paper describes on-going efforts within the PREMIS data dictionary's Editorial Committee to define an improved metadata description for them.

## Keywords

H.1.0 [**General Models and Principles**]: PREMIS; preservation metadata; technical environments; software preservation; hardware preservation; representation information; representation information network; conceptual modelling.

## 1. INTRODUCTION

"Metadata" is information about an object that is needed in order to manage that object. "Digital preservation metadata" is the information that is needed in order to preserve digital objects successfully in the long-term so that they can be deployed in some form in the future [1]. A digital object is not usable without a computing environment in which it can be rendered or executed. Digital objects are normally not self-descriptive and require very specific intermediary tools for access by humans and specific knowledge for interpreting them. Neither may be commonly available amongst a repository's Designated Community (as defined in OAIS [2]). Because of this, information that describes the sufficient components of the digital object's environment constitutes essential representation information that is needed in order to be able to use the digital object and to make it understandable in the future.

Core metadata for the digital preservation of any kind of digital object is specified in the PREMIS Data Dictionary [3], a de-facto standard. Core metadata is the metadata that is needed by most preservation repositories, rather than application or content specific metadata defined for niche uses. Metadata about digital objects' computing environments must be preserved together with the digital objects as part of their core metadata.

In addition to describing an Object's representation information, some computing environments, such as software, can themselves be the primary objects of preservation, as may be the case for computer games. They may also take the role of a software Agent in a preservation Event, and may require a thorough metadata description for those reasons.

Although there are semantic units for recording environment information in PREMIS version 2, these have rarely, if ever, been used. In 2011, the PREMIS data dictionary's Editorial Committee commissioned a working group to re-examine what computing environment metadata needs to be captured in order to be able to successfully redeploy digital objects in the long-term. This paper describes these on-going efforts. The result may be implemented in version 3 of the PREMIS Data Dictionary.

## 2. PRESERVING COMPUTING ENVIRONMENTS

### 2.1 The Current State

In version 2 of the PREMIS Data Dictionary [3], there are four key entities that need to be described to ensure successful long-term preservation of digital objects: Object, Event, Agent and RightsStatement. The Object entity provides two places to describe subordinate environments. For one, there is the "environment" semantic unit that permits the description of software, hardware and other dependencies. Rather than being an entity per se, an Environment is modelled as a semantic unit container that belongs to an Object and is, therefore, subordinate to the Object entity. The second environment-related semantic unit is the "creatingApplication" that also is sub-ordinate to the Object entity. Creating applications are outside the scope of an OAIS repository and have therefore been historically treated separately from other Environment descriptions. In a generic digital preservation framework that is not restricted to OAIS use, but supports the end-to-end digital preservation life-cycle, one would describe Environments uniformly, no matter in what context they are used. Our proposal prefers a solution that accommodates this view.

Its subordinate position to Objects means that Environments can only be captured to describe an Object's computational context. This has the following limitations:

- Environments are too complex to be handled in an Object repository.

- Environments are rarely specific to a single Object, resulting in their redundant spread across different Objects. This results in
    - unnecessary verbosity;
    - cumbersome management of Environment descriptions as they evolve.
- They are unable to describe stand-alone Environments and unable to be used for modelling an Environment registry that describes Environment components without the need for creating Objects.
- They are primarily applicable to computing environments and do not include representation information in the broader sense. This restricts the description to a technical level rather than to a level that comprehensively enables redeployment.

Our use case analysis identified the five desirable relationships illustrated in Figure 1. Because Environments are subordinate to Objects, it is impossible to express the latter four of them.

1. An Object specifies its Environment, i.e. its computational context. This is the existing relationship in PREMIS 2.
2. An environment (for example, software source code) is to be preserved as first-class entity in its own right. It is described as Environment and takes on the role of an Object.
3. An environment is described as Environment and takes the role of an Agent (for example, as software Agent involved in a preservation action Event).
4. An environment is described as Environment and is related to another Environment through inclusion, dependency, derivation or other relationships.
5. An environment is described as Environment and has an Event associated with it (for example, a creation or versioning Event).

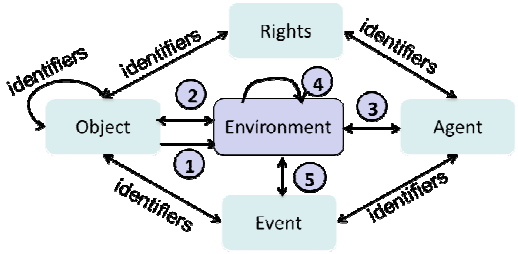


**Figure 1: The basic entities of the PREMIS Data Dictionary (in blue) with the desired Environment entity and their relationships.**

Another limitation is that in PREMIS 2, Environments are unable to refer to external dedicated registries, which would enable the delegation of "up-to-date and complete" information to an external source if needed. The identified shortcomings may be the reason that the Environment semantic container in PREMIS is rarely used.

The goal of the PREMIS Environment Working group is to rethink the metadata specification for environments. Their description must meet the improved understanding of how to ensure their longevity.

## 2.2 Related Work

The comprehensive conceptual model of the digital preservation domain in Dappert and Farquhar [4] includes Environments, Requirements (including significant characteristics) and Risks as first-order entities and justifies why this is beneficial.

There are also several efforts in the digital preservation community to specify the metadata needs for certain aspects of computing environments.

Specialised metadata has been defined to support the preservation of software. For example, "The Significant Properties of Software: A Study" project [5, 6] identified *Functionality*, *Software Composition*, *Provenance and Ownership*, *User Interaction*, *Software Environment*, *Software Architecture and Operating Performance* as the basic metadata categories for software that should be applied on *Package*, *Version*, *Variant* and *Download* level. The Preserving Virtual Worlds project [7], POCOS [8], SWOP [9] and DOAP [10] have made proposals for software preservation metadata. Examples of software repositories, the NSRL National Software Reference Library [11], MobyGames [12] and AMINET [13] illustrate practically used metadata schemas, but do not necessarily support digital preservation functions. JHOVE [14], PRONOM [15], UDFR [16] and the Library of Congress [17] have defined metadata that is needed to technically or qualitatively describe file formats and have built repositories based on their metadata descriptions. This includes some software metadata specifications, which, for PRONOM, are now available in a linked data representation and for UDFR contains software description in the recently released UDFR database [18].

There are metadata initiatives that address more complex dependencies. The Virtual Resource Description Framework (VRDF) [19] captures virtualized infrastructures; the Cloud Data Management Interface (CDMI) [20] "describes the functional interface that applications use to create, retrieve, update and delete data elements from the Cloud"; and the Web Service Definition Language (WSDL) [21] describes network services as a set of endpoints operating on messages.

The KEEP project on emulation [22] designed a prototype schema for the TOTEM database [23]. It is a recent move towards building a repository for describing the technical properties of computing and gaming environments including software and hardware components. The IIPC [24] has developed a technical database based on a computing environment schema as foundation for web archiving, and TOSEC (short for "The Old School Emulation Centre") [25] "is dedicated to the cataloguing and preservation of software, firmware and resources for microcomputers, minicomputers and video game consoles."

The TIMBUS project [26] addresses the challenge of digital preservation of business processes and services to ensure their long-term continued access. TIMBUS analyses and recommends which aspects of a business process should be preserved and how to preserve them. It delivers methodologies and tools to capture and formalise business processes on both technical and organisational levels. This includes preservation of their underlying software infrastructure, virtualization of their hardware infrastructure and capture of dependencies on local and third-party services and information. This means that, in addition to technical preservation metadata, it draws on metadata standards that capture business processes, such as BPMN [26], and identifies forms of supporting business documentation needed to redeploy processes and services.

Environments correspond to the "Representation Information" of the OAIS information model [2]. Representation Information is "the information that maps a Data Object into more meaningful concepts" [2]. Examples for a specific .docx file would be its file format specification that defines how to interpret the bit sequences, a list of software tools that can render it, hardware requirements, the language in which the contained text is written, and context information that states the author, purpose and time of its writing. Environments include documentation, manuals, underlying policy documents, cheat sheets, user behaviour studies, and other soft aids for interpretation.

# 3. MODELLING CHOICES

The following principles guided us through the modelling choices:

- Ensure backward compatibility with the existing PREMIS Data Dictionary,
- Ensure compliance with the OAIS information model,
- Provide straightforward Data Dictionary semantics that are easy to implement and that can be implemented within the existing XML Schema and PREMIS ontology,
- Provide clear mapping of historic Environment features to the newly proposed ones.
- Permit an Environment instance to describe a physical item such as software, hardware, a format, a document, a policy or a process. It may or may not be in digital form. It may be more or less concretely specified.

## 3.1 A Possible Solution

We propose to treat Environments as first class entities that do not have the limitations listed in Section 2.1. Treating Environments as first class entities also makes it more natural to model preservation actions that directly impact Environments, such as data carrier refresh or emulation, as easily as preservation actions that directly impact Objects, say migration. This is particularly important for the preservation of computer games and other kinds of software. While describing those actions is possible with the PREMIS model in version 2, it is not doable in a very natural way.

## 3.2 Supporting Different Verbosity Needs

Having a dedicated Environment entity gives implementers the ability to make precise and complete descriptions that can be shared with others. To ensure that all needed levels of description can be realised using the PREMIS 3 Data Dictionary, we considered 3 description levels that were designed to match 3 different verbosity levels.

- The most concise: Full outsourcing to an external description. Here the implementer merely wants to point an Object, or an Agent, to a description of its supporting Environment available elsewhere, most likely in some technical registry. This could be achieved by adding a linkingEnvironmentIdentifier from the Objects and the Agents without maintaining the resource that is being referred to.
- The intermediate one: A link is made between an Object or Agent, and its supporting Environment. The Environment instance is described and maintained in the repository, but its components are summarised within its description, rather than elaborated as individual Environments with precise descriptions of all their semantic units that are then linked to each other. This Environment description can be shared across Agents and Objects, but its component descriptions are not usable individually.
- The most verbose, and precise one: the Environment instance is fully described as a network of modular components, where each Environment is a separate instance. This can be achieved by adding relationships between Environments.

New PREMIS semantic units for Environments should support these description needs, and each more concise verbosity level is built on the basis of the semantic units of the more verbose levels. This way we can maintain a single consistent data dictionary while allowing different levels of description.

## 3.3 Modelling a Catch-All Term Precisely

Depending on the context, "Environment" can refer to different things. Here are some examples:

- "This operating system only runs on a 64-bit environment". The environment is hardware, but it is a category consisting of several hardware architectures.
- "This data object can be read on a European NES Games Console environment". Here the Environment is defined precisely and integrates hardware (including cartridge and controllers) and software (notably the BIOS) at the same time.
- "This ePUBReader plugin requires Firefox 3.0 or later as an execution environment". Here the Environment merely references software, without pointing to a precise version (all Firefox versions above 3.0 are supposed to work).

These examples demonstrate the following characteristics:

- Environments can connect to other Environments and can consist of related Environment components at lower levels of granularity.
- Depending on the context, as determined by business requirements, different environment subsets are relevant. An Environment can be atomic, freely usable within other Environments; but it can also be a set of running services that achieve a defined purpose (e.g. render an object).
- Environments have a purpose. They allow objects to be rendered, edited, visualised, or executed.
- Some Environments are generic; only the critical aspects of the Environment are specified. Several versions of the Environment or Environments with the same relevant behaviour can be used in its stead.
- Others are specific, real-world instances that are being used or have been used in the lifecycle of preserved Objects.

For capturing the connected nature of Environments, we decided not to introduce a separate concept for "components". Instead, we treat Environments as entities that can be recursively defined by logical or structural relationships of sets of other Environments. As with other kinds of aggregation, experience proves that, in an implementation-dependent context, what is the top-level entity and what constitutes components varies and results in the choice of different subsets of Environments. Using a recursively-defined Environment entity means that Environments can be flexibly reused in order to create new Environments as dictated by changing business needs. As we had stated that Environments correspond to the "Representation Information" of the OAIS information model, the recursively defined Environment entity forms a Representation Information Network.

## 3.4 Referring to External Registries

PREMIS evolved from an OAIS tradition. Its goal is to define all preservation metadata that is needed to safeguard Objects stored in an OAIS repository. This excludes events before the Object is ingested into the repository and focuses on the preservation of bitstreams, files and structurally related sets of themfiles, captured as representations. It was not intended that it would take the role of a registry, where descriptions and definitions are stored for reuse. Technical registries share with PREMIS the aims of supporting "the renderability, understandability of digital objects in a preservation context" and of representing "the information most preservation repositories need to know to preserve digital materials over the long-term". Technical registries do NOT describe content "held by a preservation repository".

As the above examples show, for preservation purposes, an Environment can be a generic description of technical or other characteristics that intend to make the preservation task easier for preservation repositories, but can be increasingly concrete to the point where it would describe a concrete custom-tailored environment for a specific repository. The two domains of registry and repository touch. In a Linked Data implementation there is an almost seamless continuum from the repository preserving digital objects to the external environment descriptions in external registries.

Adding the Environment entity broadens the scope of PREMIS. It focuses no longer only on the Objects preserved in a repository, but also on the representation information needed to render or execute the Object. It captures its reticular nature and core semantics with a new dedicated entity and its semantic units. In the extreme, one could even imagine technical registries using "premis:Environment" natively to describe standalone Environments without relating them to any Object or Agent.

## 3.5 Matching Environments to the Existing Data Model

We propose to make Environment a new first-class entity so that it can be described with its own semantic units. Therefore, we need to match it to the existing data model, so that backwards compatibility is maintained and so that it is clear when something should be described as an Object, an Agent or an Environment.

In order to achieve reusability and varying levels of specificity an Environment instance should **describe** its characteristics but it should **not state how it is used** in an OAIS repository.

Within an OAIS repository an Environment can take three roles:

- It can take the role of representation information for an Object so that the Object can be redeployed successfully in the future (relationship 1 depicted in Figure 1).
- It can be preserved in the repository for example, to preserve software or a computer game (relationship 2 depicted in Figure 1).
- It can act as an Agent involved in an Event (or, less likely, in a RightsStatement) (relationship 3 depicted in Figure 1).

The fact that an Environment takes on any of these roles is specified in the Object and Agent that captures this information. That is to say that, for example, if an Environment component describes an Agent that is involved in a preservation action Event then a corresponding Agent instance should be created and related to the Environment description. If an Environment component is to be preserved, then a corresponding Object instance should be created, the Environment's content has to be captured as an Information Package so that it can be considered an Object, and the instance should be related to the Environment description. If one wishes to merely specify the Environment as representation information for an Object, then again, the Object instance should be created and related to the Environment description.

## 3.6 Identifying Environments

As indicated in Figure 1, the solution for capturing Environments needs to specify how Environments are to be identified and how other entity instances should link to them. PREMIS 2 offers several different ways of identifying and linking to entity instances. The proposed solution should mirror them for consistency's sake. The existing approaches include:

- Linking to an entity instance through the **identifier type and value** of the target instance:

linking[Entity]Identifier, to unambiguously link an instance of one entity to an instance of another kind of entity, e.g. an Object to an Event; these links can be particularised with a linking[Entity]Role that allows one to specify the role of the referred entity.

relationship, to unambiguously relate different instances of the same entity, i.e. an Object to another Object. This relationship must be particularised with a type and a subtype. Currently the type values "structural" and "derivation" are suggested values in the Data Dictionary.

dependencyIdentifier, to relate an Object to a file that is needed to support its delivery, e.g. a DTD or an XML Schema.

- Linking to an entity instance through a **registry key:** formatRegistryKey, to relate a file or bitstream Object to a description of its format in an external registry.
- Linking to an entity instance through a **designation**: formatDesignation, to identify a format by name and version.

An Environment as a PREMIS entity must define its identifierType and identifierValue as all other PREMIS entities do. PREMIS Environments are instances that can be linked to from other entities using the premis:identifier mechanism through a linkingEnvironmentIdentifier recorded in the linking Object, Agent or Event (the linking relationships 1, 2, 3 and 5 depicted in Figure 1 pointing towards Environment). For the bi-directional relationships 2, 3, and 5 in Figure 1 one may use the linking[Entity]Identifier from within the Environment entity to identify related Objects, Agents or Events.

The question of whether Environment descriptions are stored as separate Information Packages in the repository or whether they must be stored together with the Objects or Agents whose role they take should not be specified within the PREMIS Data Dictionary since PREMIS is implementation independent. As with all implementations, however, if the PREMIS identifier mechanism is used, it must be guaranteed that it persistently and uniquely identifies the entity.

We are proposing a variety of mechanisms for implementing the relationship 4 depicted in Figure 1, which relates one Environment instance to another.

From within an Environment instance, one can refer to other Environments, such as from the description of a software application as Environment A to its operating system as Environment B. This would take the form of a relatedEnvironmentIdentifier

link using the PREMIS identifier mechanism to capture structural, derivative and dependency relationships.

Additionally, from within a local Environment instance in a repository one can refer to the corresponding (possibly, more complete or more up-to-date) descriptions in other registries (e.g. TOTEM or PRONOM). Here a premis:registryKey could be used to refer to information about the description in an external registry. Note that such a description does not imply identity between the Environment descriptions in the repository and the registry. Because of the sliding specificity of Environment descriptions (see Section 3.3) it is almost impossible to assert that two descriptions are identical. We assume that the referenced Environment description in the registry has to be more generic, and, therefore, can be inherited.

A further form of linking to an external Environment description could be an Environment designation, consisting of name and version. Additional specifications, such as the country of release of the version can be used to identify the Environment precisely.

In order to allow referring to different, internal or external descriptions of the same Environment at the same time, any form of linking should be repeatable and combinable. Each use of a linking mechanism should declare its role by some mechanism, such as premis:registryRole or linking[Entity]Role.

## 3.7 Expressing Dependencies between Environments

How Environments depend on each other so that they can be run, is key preservation information, which has to be expressed in the most satisfactory way possible. In PREMIS 2 dependencies can be expressed in two places:

1. DependencyIdentifier is used to document a non-software dependency between an Object and another Object, and uses an identifier mechanism to link to the required object.
2. swDependency expresses the fact that a piece of software, part of an Environment supporting an Object, relies on other software to be executed. This swDependency semantic unit is a "full text description" with no linking capability.

A gap analysis uncovered some areas for improvement. For example, low-level software Environments, like operating systems, rely on hardware to run. There is no explicit possibility in PREMIS 2 to document the nature of the dependencies. One can loosely record a hardware and software description in the same Environment container but not express the fact or the nature of their dependence. Links to repository descriptions are currently possible for file formats but not for other environment types. Specification of versions are possible for software, but not for hardware.

With the proposed PREMIS 3 change of Environment becoming a first-class PREMIS entity rather than a semantic container in the Object description, explicit linking mechanisms for describing dependencies can be used.

The existing ways of achieving the goal of expressing dependencies have to be simplified and re-factored so that they are as easy to use (for implementers) and to maintain (for the PREMIS Editorial Committee) as possible, while maintaining expressiveness.

PREMIS has a generic and powerful mechanism that allows linking two descriptions and assigning a type to the link. The two most generic semantic units are the linking[Entity]Identifier and the relationship ones. They can both be used for linking Environments, maintaining the existing pattern that the former links two instances of different entities, and the latter links two instances of the same entity. Thus:

- Whenever there is the need to express the fact that a preserved Object or an Agent relies on an Environment to run, you use a linkingEnvironmentIdentifier mechanism;
- Whenever there is a dependency between two Environment instances, a premis:relationship with a new relationshipType of "dependency" can be used; this achieves the goal of the previous swDependency, and allows other dependencies, such as hardware dependencies, to be expressed as well. This is in addition to the structural and derivative relationships between Environments mentioned above. This implements the linking relationship 4 depicted in Figure 1.
- Whenever the dependency occurs between two Objects, the premis:relationship mechanism with the new relationshipType of "dependency" can be used between their Environments. This achieves the same purpose as the "dependencyIdentifier" PREMIS 2 feature described above.

The other advantage of this mechanism is its extensibility: the relationshipType and relationshipSubType semantic units' recommended values in the Data Dictionary can be augmented. This is important as we cannot foresee all the relationships that can occur between Environments, which is a complex and evolving area. An example of a large variety of dependency relationships can be found in the Debian policy manual [28]. Using the relationship mechanism is a way to leave the door open to other relationships that could be needed in the future. Because of Environments' highly interconnected, networked nature, the Data Dictionary solution should enable all of these linking and identification options.

## 3.8 Environments or Proxy Descriptions

When modelling Environments there is a decision to be made what form and content this Environment should take. If it will be preserved in an OAIS repository it will necessarily take the form of a digital bitstream, file or representation. Software and supporting documents, such as policy representations or manuals, can be captured directly in digital form as an Information Package. Hardware, business processes or non-digital documents are inherently not (necessarily) represented digitally and thus not directly subject to digital preservation as preservation Objects.

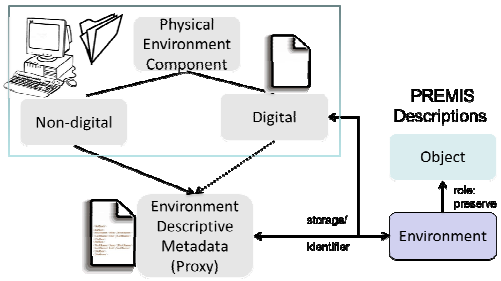


**Figure 2: Environment components as preservation Objects**

In either case, the object can be reduced to a proxy digital description that can be preserved as an Object. This descriptive environment metadata captures the physical object's relevant characteristics and contains all the information needed to rede-

ploy a corresponding environment component with these same relevant characteristics in the future. This kind of environment preservation through proxy descriptions is used in, for example, business process preservation, as illustrated in the TIMBUS project [26]. See Figure 2 for an illustration.

A functional software description or the specification "Adobe Reader 5.0" can be considered instances of proxy descriptive Environment metadata. It is not as concrete as the Adobe Reader software composed of 0s and 1s, in the form of digital files that are the actual physical Environment component. Either or both could take the role of a premis:Object.

It is a business decision of the repository whether it preserves the actual digital representation of the Environment and/or Environment descriptive metadata as a proxy. This is a semantic issue. As with other curatorial decisions, this cannot be prescribed by the PREMIS data dictionary. But the eventual solution for PREMIS Environments must accommodate either use and allow for the nature of the Environment description to be specified.

## 3.9 Existing PREMIS Environment Descriptions

Keeping the existing solutions for describing Environments in PREMIS 2, the "environment" semantic unit and the "creatingApplication" semantic unit, enables backwards compatibility and, pragmatically speaking, offers convenient shortcuts and reduced verbosity for the situations in which they suffice. The PREMIS Environment working group does, however, feel that we would recommend the new Environment entity above those legacy semantic units.

# 4. USE CASE BASED DESIGN

The proposed solution is based on concrete examples rather than abstract considerations. It was driven by and validated with use case analysis. The working group validated that the modelling decisions, which were taken in extending the expressive capacities of PREMIS beyond the sheer description of preserved Objects to representation networks, were applicable to real-world examples.

Use cases should address all scenarios that implementers would expect to implement using PREMIS 3 Environments. The following examples were chosen:

- Describing the environment that is used to render web archives in a particular institution, with all the pieces of software that it bundles together to achieve this purpose;
- Describing the environment used in a normalization event;
- Describing the environment, including testbeds and documentation, used during TIFF to JPEG2000 migration;
- Describing an emulation environment for a Commodore 64 game preserved as an Object;
- Documenting the business processes in a multinational enterprise that operates in the cloud, and all the software and hardware dependencies that allow them to be re-deployed in the future.

The first two have been implemented in detail with a draft Data Dictionary proposal. With their help, it is possible to illustrate some of the features of the proposed Environment extension.

## 4.1 Use Case: Rendering Environments for Web Archives

In the first use case, harvested web pages from the web archives are rendered in the National Library of France's reading room Environment. A web page harvested in 2010 can not necessarily be rendered on the reading room Environment of 2010. For example, for a web page harvested in 2010 that contains an EPUB file, this 2010 environment works for the HTML page. But the Firefox 2.0.0.15 browser it includes does not support EPUB files. The reading room Environment is upgraded in 2012 to an Environment that contains a newer version of Firefox that supports the EPUBReader plugin that allows one to render the EPUB file. In other terms, there was a need to describe these two Environments, the fact that one Environment is superseded by another, the different software components that they include, and the dependency relationships between them.

The preserved Object and its history are described with the PREMIS 2 standard features (Object, Event and Agent) as can be seen in Figure 3. The Environments are described separately and linked to from the Objects they support.

A new relationship type had to be introduced to state that the old Environment was superseded by the newer one. This information can, for example be used if the most current environment becomes obsolete. A preservation professional may choose to track superseded environments, which achieved the same purpose, in the hope of detecting a by-now readily available emulator of the older environment. This is an important feature for hardware and software preservation. This was achieved by a new relationshipType called "replacement", with relationshipsubTypes of "supersedes" or "is superseded by".

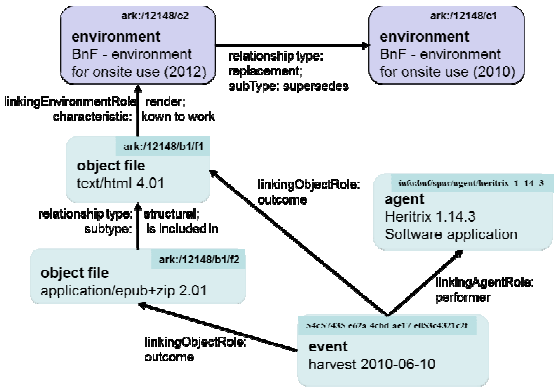


**Figure 3: Web archive use case**

This use case highlights how the environmentPurpose and environmentCharacteristics, familiar from the PREMIS 2 "environment" semantic unit, should be treated. The former was about the purpose an Environment wants to achieve towards a particular object (e.g. create, render, edit) and the latter, about the requirement that the Environment is intended to fulfil for a particular object (e.g. minimum service required, known to work). This should not be part of the Environment itself but part of the relationship between an Environment and the entity (Object or Agent) that it supports. This also increases the ability to share descriptions since the same Environment described above could potentially be used to achieve different purposes with different requirements.

Figure 4 shows the components of those two Environments. Each component is an individual Environment, and bundled into "aggregator" Environments. The aggregate mechanism, allows components to be shared across different Environments. For example, the Windows XP Service Pack 2 description is shared by both Environments since they use the same operating system.

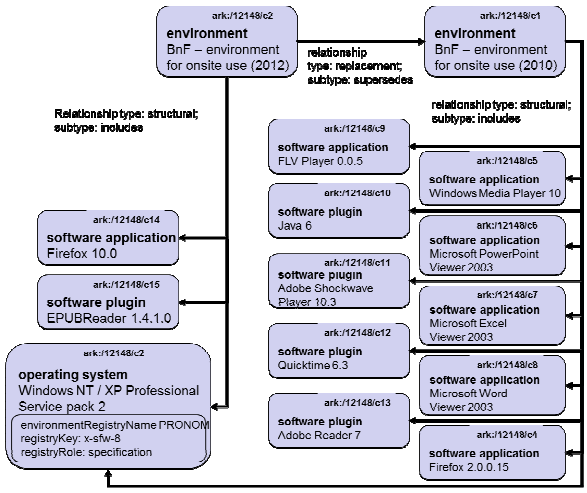


**Figure 4: Inclusion links between Environment platforms and their component Environments**

It also illustrates how one can link to a registry for additional descriptive information. Here, the Environment instance "ark:/12148/c2" describes Windows XP with a particular service pack; on the other hand, there is a description in PRONOM about Windows XP "in general", with no particular service pack. In spite of this difference, adding this entry as a reference can be useful since the PRONOM description is likely to evolve and be enriched over time. A pointer to a repository should only be used if the description found there is an exact match or more generic and abstract than the Environment instance that links to it, so that the link does not cause conflicts in the Environment description.

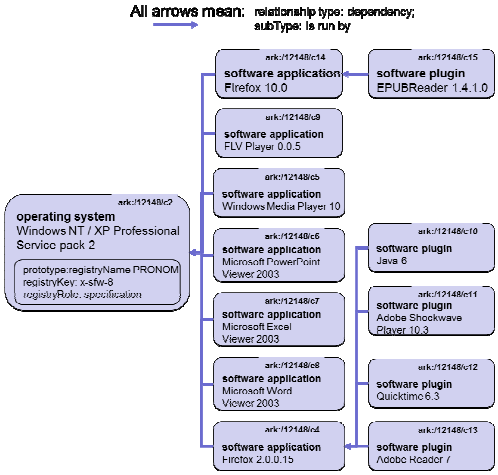


**Figure 5: A dependency network between Environments**

However, Figure 4 does not express all the required information. There is also the need to express the dependency relationships between the different components. Windows XP Professional SP2, Firefox 10.0, and EPUBReader 1.4.1.0 are all part of the same aggregator Environment, but they do not act on the same level. EPUBReader, as an add-on, runs on Firefox 10.0, which in turn runs on Windows XP Professional SP2. These dependencies were documented by using another PREMIS relationship between the environments, as can be seen in Figure 5.

These two different relationships have to be distinguished because they do not act on the same level and do not achieve the same purpose. On the one hand, the whole/part structural links between Environments and their components are about picking Environment components to set up and bundle an Environment platform for a particular purpose, and are thus specific to a particular repository and implementation. On the other hand, the dependency relationships between the components are true whatever the context is.

## 4.2 Use Case: Documenting an Environment Used by a Normalization Service

In this use case, a QuickTime file with dv50 video and mp3 audio streams is submitted to a repository. Upon ingesting the QuickTime file, the archiving institution normalizes the file into a QuickTime file with mjpeg video and lpcm audio streams. A normalization event is recorded, along with the web service and software that performed the format conversion. The derivation links between Objects, and their provenances are described by standard PREMIS entities and semantic units. The new feature is about the Agent description, which is a normalization service with no further description. So the Agent is linked to an Environment which describes what components are actually used by the service, e.g. libquicktime 1.1.5 with dependent plug-ins. The whole description can be summarized in Figure 6 below.

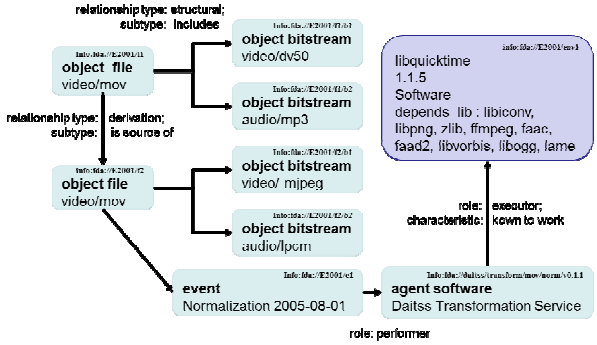


**Figure 6: Normalisation use case**

The distinction between the Agent and the Environment that executes it is important, if one wants to preserve an Agent so that it could be re-enacted in different Environments, or if one wants to track errors that have been discovered or link to an external registry. To this end, one may need to document the software components of which the Agent is built, along with the different Events that have been performed by this Agent in a repository. All this can be done by following the links between those different entities. This example also shows that different verbosity levels can be achieved depending on the implementer's needs. While the web archives use case above used a very thorough Environment network description, this normalization example describes the execution Environment of an Agent more concisely. All the dependent libraries are listed in a single envi-

ronmentNote semantic unit. However it shall be noted that a more precise description could have been made if needed. In such a case, there would have been a distinct Environment description for each component (the software application and all its libraries), an inclusion link to an aggregator Environment executing the Agent, and, finally, dependency relationships between the libraries and the application. All depends on how far a PREMIS implementer needs, or wants, to describe Environments supporting the Objects s/he preserves or the Agents s/he uses. This ability to fit different needs is one of the key principles that guided this study.

## 5. CONCLUSION

The PREMIS Environment working group has been tasked with rethinking how a computing Environment should be modelled so that it meets the digital preservation community's requirements. Several open issues are still being investigated. The analysis and proposed solutions discussed in this paper will be brought to the PREMIS Editorial Committee and will be validated on community-provided use cases. Working within our stated modelling principles, we hope that our proposed approach not only meets contemporary registry preservation needs, but also improves the interoperability between Environment registries that are being developed within the community. The working group has included representatives from the PREMIS [3] Editorial committee, the TOTEM [21] technical registry, the IIPC [24], DAITSS [29] and the TIMBUS [26] project, and has received user requirements from New York University.

## 6. ACKNOWLEDGEMENTS

The authors wish to thank Michael Nolan, Martin Alexander Neumann, Priscilla Caplan and Joseph Pawletko for their contributions. Part of this work has been funded by the TIMBUS project, co-funded by the European Union under the 7th Framework Programme for research and technological development and demonstration activities (FP7/2007-2013) under grant agreement no. 269940. The authors are solely responsible for the content of this paper.

## 7. REFERENCES

Websites were accessed on 8 May 2012

[1] Dappert, A., Enders, M. 2010. Digital Preservation Metadata Standards, NISO Information Standards Quarterly. June 2010. http://www.loc.gov/standards/premis/FE_Dappert_Enders_MetadataStds_isqv22no2.pdf

[2] CCSDS, June 2012. Reference Model for an Open Archival Information System (OAIS): version 2. CCSDS 650.0-B-1, Blue Book (the full ISO standard). http://public.ccsds.org/publications/archive/650x0m2.pdf

[3] PREMIS Editorial Committee, 2012. PREMIS Data Dictionary for Preservation Metadata, Version 2.2. http://www.loc.gov/standards/premis/v2/premis-2-2.pdf

[4] Dappert, A., Farquhar, A. 2009. Modelling Organizational Preservation Goals to Guide Digital Preservation, Vol.4(2)(2009) of International Journal of Digital Curation. pp. 119-134 http://www.ijdc.net/index.php/ijdc/article/viewFile/123/103

[5] Matthews, B., McIlwrath, B., Giaretta, D., Conway, E. 2008. The Significant Properties of Software: A Study. STFC, December 2008. http://bit.ly/eF7yNv

[6] Software Sustainability Institute, Curtis+Cartwright. 2010. Preserving software resources. http://software.ac.uk/resources/preserving-software-resources

[7] McDonough, J., et alii. 2010. Preserving Virtual Worlds. https://www.ideals.illinois.edu/bitstream/handle/2142/17097/PVW.FinalReport.pdf?sequence=2

[8] POCOS. Preservation of Complex Objects Symposia. http://www.pocos.org/

[9] SWOP. SWOP: The Software Ontology Project. http://www.jisc.ac.uk/whatwedo/programmes/inf11/digpres/swop.aspx and http://sourceforge.net/projects/theswo/files/

[10] Dumbill, E. 2012. DOAP- Description of a Project. http://trac.usefulinc.com/doap

[11] National Software Reference Library (NSRL). National Software Reference Library www.nsrl.nist.gov/,

[12] MobyGames. http://www.mobygames.com/

[13] AMINET. http://aminet.net/

[14] JSTOR and the Harvard University Library. JHOVE - JSTOR/Harvard Object Validation Environment. http://hul.harvard.edu/jhove/

[15] The National Archives: PRONOM. http://www.nationalarchives.gov.uk/pronom/

[16] Unified Digital Format Registry (UDFR) http://www.udfr.org

[17] Library of Congress. www.digitalpreservation.gov/formats/

[18] UDFR. UDFR ontology. http://udfr.org/onto/onto.rdf

[19] Kadobayashi, Y. 2010. Toward Measurement and Analysis of Virtualized Infrastructure: Scaffolding from an Ontological Perspective. http://www.caida.org/workshops/wide-casfi/1004/slides/wide-casfi1004_ykadobayashi.pdf,

[20] SNIA. Cloud Data Management Interface (CDMI) http://www.snia.org/cdmi

[21] W3C. 2001. Web Services Description Language (WSDL) 1.1. www.w3.org/TR/wsdl

[22] KEEP (Keeping Emulation Environments Portable). http://www.keep-project.eu/ezpub2/index.php

[23] TOTEM database. Welcome to TOTEM - the Trustworthy Online Technical Environment Metadata Database. http://keep-totem.co.uk

[24] IIPC Preservation Working Group. http://netpreserve.org/about/pwg.php

[25] TOSEC (The Old School Emulation Centre). What is TOSEC. http://www.tosecdev.org/index.php/the-project

[26] TIMBUS project. http://timbusproject.net

[27] Object Management Group. 2011. Business Process Model and Notation (BPMN). Version 2.0. Release date: January 2011. http://www.omg.org/spec/BPMN/2.0/PDF

[28] Debian. Debian Policy Manual. Chapter 7 - Declaring relationships between packages. Version 3.9.3.1, 2012-03-04. http://www.debian.org/doc/debian-policy/ch-relationships.html

[29] DAITSS. http://daitss.fcla.edu/