Automatic Discovery of Preservation Alternatives Supported by Community Maintained Knowledge Bases

Rudolf Mayer, Johannes Binder, Stephan Strodl
Secure Business Austria
Vienna, Austria

Andreas Rauber
Vienna University of Technology
& Secure Business Austria
Vienna, Austria

ABSTRACT

Preservation Planning, which deals with selecting the most appropriate preservation action to be applied to digital objects, is an important step in any digital preservation activity. Comprehensive Preservation Planning depends on the availability of identified alternatives of preservation actions, which are for example file format migrations to migrate data in an outdated format to one that has better support. Also emulation, e.g. of the behaviour of a specific software application (application emulation), can be a viable preservation action. The alternative identification step can either be performed manually by an expert, or (semi-)automatically, if appropriate knowledge bases are available. Building and maintaining such knowledge bases is however a tedious task, as the number of software applications and file formats, and especially their relation to each other, is very large. In this paper, we therefore present an approach to automatically build knowledge bases for Preservation Planing from already existing, open resources. One such source is the community maintained Freebase, which contains linked data on many topics, among them file formats, software applications, and most importantly, their relations, in a structured manner. We demonstrate the applicability of these knowledge bases by automatically identifying possible digital preservative actions on a uses case, an eScience experiment from the domain of data mining. This use case originates from the task of process preservation, where we look beyond single files, but regard complete chains of executions as the objects to be preserved.

1. INTRODUCTION

Preservation planning can be understood as a form of utility analysis, where each different possible preservation action is quantified. The goal is to select the most appropriate preservation action to be applied to digital objects. Preservation Planning is a vital step in any digital preservation activity.

An important phase in Preservation Planning is to identify viable preservation actions, i.e. to identify which actions can be applied to the digital objects that would prepare them to be usable in the future. Such preservation actions are for example file format migrations to migrate data in an outdated format to one that has better support. In most cases, there is a wealth of possible formats to convert into. Also emulation, e.g. the emulation of the behaviour of a software application, is an important approach in digital preservation.

Business processes are a more complex form of digital objects, where the domain of interest moves beyond single files, but to complete chains of process executions, including a number of files generated and consumed, and the software needed to manipulate them. To provide a faithful preservation of the execution of the process, preserving the behaviour of the software stack required for the process steps becomes necessary. In the setting of process preservation, we thus look beyond single files, but also regard the complete chain of a process execution, and the environment a process is executed in. Therefore, we move from regarding only the view path of single object, towards the more complex interaction of multiple view-paths that might be realised in the same system.

Alternative identification for preservation planning can either be performed manually by an expert, or automatically, if appropriate knowledge bases are available. Building and maintaining such knowledge bases is however a tedious task. In this paper, we therefore present an approach to automatically harvest such knowledge bases for Preservation Planing from already existing resources. Specifically, we utilise the community-maintained Freebase, as well as the domain of Linux software packages. On top of these knowledge bases, we develop a service that can automatically identify preservation action alternatives for a given system. These systems need to be described in a formal way according to a model recently proposed in [2], which introduces a model to describe the context of business processes. As a part of this model, the technical environment of a system can be described.

It has to be noted that the service presented in this paper is meant for the discovery and identification of alternatives. The suitability of these alternatives for actually solving the digital preservation problem at hand still have to be assessed and verified by digital preservation experts.

The remainder of this paper is structured as follows. In Section 2 we give an overview on related work. Section 3 reviews the Context Model, which can be utilised to formally represent the context of a process, of which we are specifically interested in modelling computing systems. In Section 4 we then describe the data sources and harvesting processes to obtain our knowledge bases. Section 5 will then detail on how these knowledge bases can be utilised, in conjunction with the formal mode of a system, to identify preservation alternatives. In Section 6, we then show the applicability of the approach on a use case example. Specifically, we take an example of a process to be preserved, and analyse the different alternatives identifiable. Finally, we provide conclusions and an outlook on future work in Section 7.

2. RELATED WORK

The term *Digital Preservation* as defined in the UNESCO Guidelines for the Preservation of the Digital Heritage [18] is the process of preserving data of digital origin. The two main strategies for the preservation of digital heritage listed are migration [10] and emulation [14, 16, 7]

Emulation refers to the capability of a device or software to replicate the behaviour of a different device or software. Emulation can happen on different levels in a system:

- *Application* An application is usually utilised to render a digital object (if the digital object to be preserved is not itself an application, e.g., computer games, digital art, self-running documents, process management software). By replacing the original application interpreting the digital object the functionality of this application is emulated.
- *Operating System* On a modern computer system an operating system provides access to the underlying hardware for an application running on top of it. By providing a layer that redirects the operating system calls of the application to the same calls of a different operating system, it is possible to emulate the operating system with this additional layer on top of a new operating system.
- *Computer Architecture* The most common use of emulation is to emulate the functionality of a computer architecture by using software, thus introducing an additional layer in the software stack of a rendering environment. Physical hardware can be emulated using either full hardware emulation where all hardware components of the computer architecture are recreated in software on a new host-system or by virtualisation where the CPU is not completely emulated (like in virtualisation software such as VirtualBox[1]).

Regarding emulation, in this work, we are primarily interested in identifying emulation opportunities for applications. However, the model described in Section 5 could also be utilised to identify strategies e.g. for Computer Architecture emulation.

File format migration is a strategy of refreshing digital files over time, to keep the content stored in formats that can be interpreted by current technology. Migration might also be done anticipatory and transform contents to formats that are expected to be readable in the future. Such a migration is usually easier done today, as more tools that can read the presumably outdated format are still available. Identification of suitable format migration paths that are supported by currently available software tools is a primary concern for our approach regarding format migration.

Also for this approach, it is important to have a knowledge base on file formats, and the software that can manipulate it. Several possible sources were investigated, foremost well established registries such as PRONOM , and tools developed in the SCAPE project to facilitate preservation planning. However, these approaches did not provide a comprehensive and up-to-date data base of software that can handle the various formats.

Several attempts to build comprehensive digital preservation related knowledge bases or registries exist. The PRONOM registry[2][5], developed by The National Archives of the United Kingdom, primarily contains information on file formats, along with a classification, description, publication dates, and vendors. Further, the registry provides information on software applications, such versions, release dates, and default file formats for that software. In addition, also vendors are registered. Each entry in the registry's database is assigned a PRONOM Unique identifier. Currently, the registry holds around 1,100 file formats, as well as around 280 entries on software. It also contains basic support for identifying migration pathways, i.e. conversion chains from one format to another, along with the software that supports this. However, the database currently contains less than 50 of these pathways. PRONOM is also designed to contain information on whether a format is at risk, however, this information is generally not provided.

The Community Owned digital Preservation Tool Registry (COPTR)[3] is a registry for tools useful for preserving digital information for the long term. It contains a Wiki-style collection of tools along with a short description of their functionality. However, this information is not well structured, and can't be processed automatically. Also, links to file formats these tools are capable of processing are missing. Currently, the registry contains around 360 tool entries.

While PRONOM and COPTR surely have huge impact on digital preservation solutions that need this type of registry information, it seems that the amount of content provided is not enough for identifying a larger set of alternatives. This was also recognised by [6], where the authors try to aggregate information on file formats from several sources. They utilise linked open data repositories for this approach. We will in the subsequent sections investigate also on some of the sources utilised in that approach.

Comparing different options of preservation actions is the challenge of preservation planning. In [3] a preservation planning workflow that allows for repeatable evaluation of preservation alternatives is described.

[1]VirtualBox – `https://www.virtualbox.org/`

[2]`http://www.nationalarchives.gov.uk/PRONOM`

[3]`http://coptr.digipres.org/`

Regarding the long-term availability of software, the Software Sustainability Institute defines, among others, the following strategies [9]:

- Emulation of the execution environment, i.e. utilising emulators that mimic the functionality and behaviour of the hardware and software environment. This strategy requires *Operating System and Computer Architecture Emulation*.
- Migration of the software to a different platform. This can be as simple as just compiling otherwise platform independent software for the different platform or in worst case may require a complete rewrite of the software.
- Technical preservation of the hardware environment.
- Cultivation, by releasing the software into open source and engage the community to maintain and develop it.
- Hibernation, which includes archiving the software and the knowledge needed to use it, for a potential future use.

Most of these strategies can be useful in the preservation of software applications. However, most of them are rather alternatives that try to preserve the status-quo of the current system setup. They do not require a specific identification step of possible alternatives, which would be the case e.g. for migration of a file format, where we need to know which formats are available for a specific process setup.

The view path [17] of a digital object is the combination of a software and hardware that is required to render an object. This can be described with the *Preservation Layer Model* (PLM), which typically consists of the layers of a specific application, and operating system and the hardware supporting that operating system. However, but more complex layering is possible as well. The above mentioned techniques of migration and emulation basically modify elements in this view path. In the domain of preserving complete processes, which can be understood as a digital object itself, we normally encounter a multitude of digital objects that are manipulated in a chain. Often subsequent steps depend on the output of the previous activity. In such a setting, multiple view-paths exist, and they partly share some of the elements from the different layers, e.g. the same operating system might support two different applications used in two different steps.

3. REPRESENTATION OF SYSTEMS TO BE PRESERVED

A formal model to represent the context a process is embedded in was presented in [2]. In the setting of process preservation, all but the simplest processes require to be described by a multitude of information objects, as well as their interconnections and relations. Examples of the details to be preserved are the process model itself, and the actors involved in the process execution. On a more technical level, the infrastructure required to support the process execution is of interest. This includes the hardware and software that provide the execution platform, as well as various artefacts consumed and created during the process. Of interest are furthermore any dependencies to external parties. To enable a semantic description of these objects in a structured manner, the context model, a formal meta-model, was derived. It describes classes of elements and their relations, in the form of OWL ontologies. To be extensible, it is designed with a core (upper) ontology describing the generic concepts, and extension mechanisms to map supplementary ontologies describing more specific aspects. Ontologies are a well-suited method to implement this architecture.

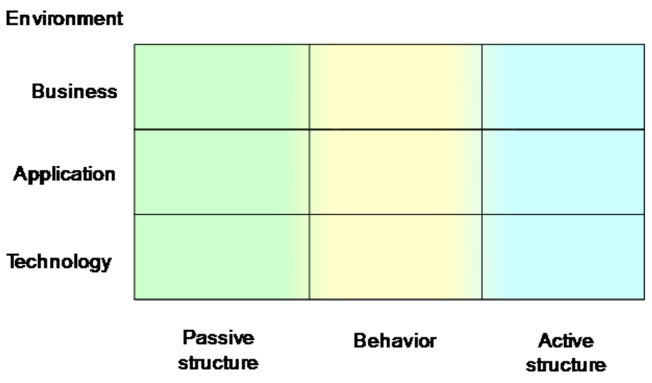


Figure 1: The ArchiMate Framework ([8])

The core ontology is based on the ArchiMate 2.0 language ([8]), an international standard from the Enterprise Architecture domain. The ArchiMate modelling language includes a minimum set of concepts and relationships. The ArchiMate framework organises its language concepts in a 3×3 matrix: the rows capture the different enterprise layers *business*, *application*, and *technology*, and the columns capture the cross layer aspects *active structure*, *behaviour* and *passive structure*. Figure 1 depicts this organisation of the framework, while Figure 2 lists the main concepts provided by ArchiMate, where the colours of the elements corresponding to the categorisation into active structure, behaviour and passive structure. Active structure contains entities capable of performing behaviour. The behaviour itself contains elements defined as units of activity performed by one or more active structure elements, and the passive structure contains objects on which the behaviour is performed.

For the task of identifying preservation alternatives, we can use the concepts of the technological layer of the framework to model our systems.

The core domain-independent ontology of the Context Model is then augmented through a set of specific extension ontologies that are tailored to explicit modelling concerns. Currently, the context model provides extensions to cover aspects such as *Legal*, *License*, *Patents*, *Data & Formats*, *Hardware*. The extension ontologies are, when possible, based on already existing languages, for which then the ontology mapping to the core ontology was provided. On overview on this is given in Figure 3. Most of these extensions map to elements in the technological layer, and are thus also of interest for our modelling concerns.

Specifically, the current implementation of the alternative

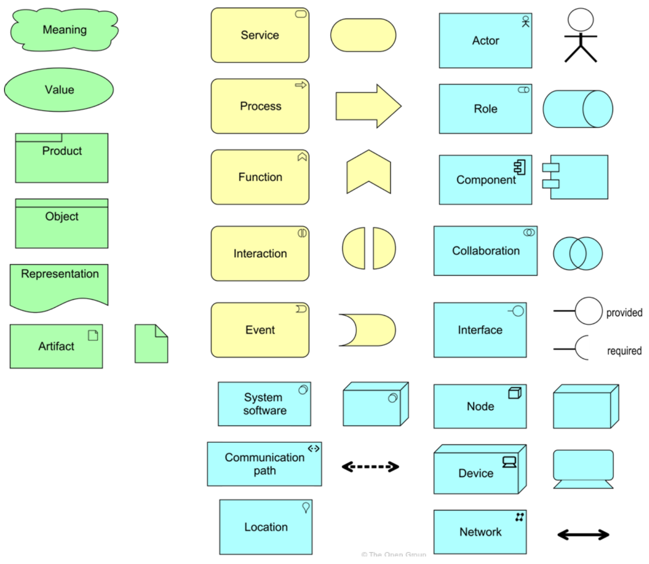


Figure 2: The ArchiMate meta-model

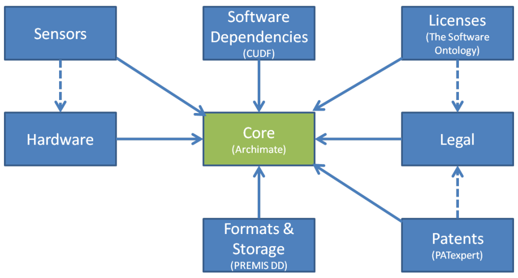

Sensors
Software Dependencies (CUDF)
Licenses (The Software Ontology)
Hardware
Core (Archimate)
Legal
Formats & Storage (PREMIS DD)
Patents (PATexpert)

Figure 3: Overview on available extensions and their relation to the core ontology

identification operates on the following entities: Artifact, SystemSoftware and FileFormat. The two former are part of the core ontology, while the third one is an element defined via the data format extensions, which is realised via the PREMIS data dictionary.

4. KNOWLEDGE BASE GENERATION

In this section, we describe two different approaches to obtain the data needed for the knowledge bases of our alternative identification service. We further discuss technical details of the representation of the knowledge.

4.1 Freebase – Software and File Formats

The online database Freebase[4] [4] provides a community driven and maintained database of semantic linked-data on various topics. Among them, there is information on software applications and file formats. The schema for software tools[5] is described in Table 4.1. Currently, there are more than 9,000 entries in this schema. The schema for file formats[6] is described in Table 4.1. Freebase contains at the moment more than 3,500 entries for file formats.

[4]`http://www.freebase.com/`
[5]`http://www.freebase.com/computer/software?schema`
[6]`http://www.freebase.com/computer/file_format?schema`

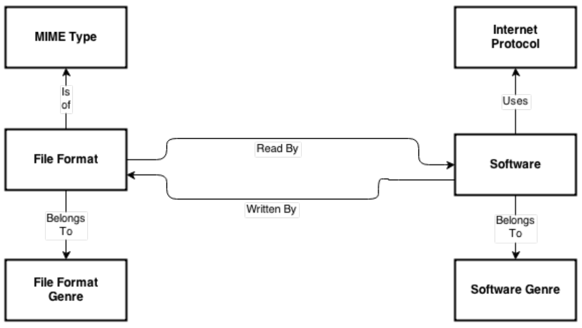


Figure 4: Relations between File Formats and Software in Freebase

An overview of some of the relations in the database is given in Figure 4. The *Written By* and *Read By* properties allow linking software applications to specific file formats. Specifically, this allows on the one hand to identify possible conversion paths from an origin file format to a desired file format, by identifying software tools that can read the origin and write the target format. In more complex cases, if no software is available that can directly do this conversion, chains of format migrations via intermediate formats can be established. On the other hand, the information on which formats can be read by a specific software allows to establish a rudimentary list of software that is compatible to each other. It is possible to deduct which software applications are capable of handling the same types of file formats, and thus, theoretically, exchangeable. Of course this identification of equivalence ignores the functionality provided by each software, and thus might return a list of false-positive equivalents. Also, some of the potential preservation alternatives might not make sense from other points of view. It therefore requires still, as mentioned above, the review and assessment of a digital preservation expert. Another approach of identifying software with similar functionality is via the genre and protocols. The former is a human classification of types, e.g. PDF readers as software that can render PDF files, while the latter can be utilised for software that is no directly manipulating files, such as an FTP client, implementing the File-Transfer Protocol.

While some of the data in Freebase is not as clean as in other registries that are dedicated to digital preservation, it has two rather big advantages. On the one hand, the process of extending the knowledge base is very simple via an online interface, and happens at a frequent rate by the community. Also, due to the linked data scheme, information from Freebase can be easily augmented by other means than directly in the Freebase database, e.g. by augmenting it by a locally available data source. Also the size of the knowledge base is an advantage for the task of alternative identification. At the moment, there are three times as many formats, and 45 times more software applications in Freebase compared to the PRONOM registry.

4.2 Software alternatives for Linux packages

A second approach to build a knowledge base for software application emulation is based on the concept of software

Table 1: Freebase Data Schema for Software

Property	Description
Developer	Manufactures of the software (e.g. organisation or person)
Software Genre	Categorisation of applications, e.g. *Database management system* or *PDF reader*
First Released	Date of the first release of this software
Latest Version	Version number of the latest release
Latest Release Date	Date of the latest release
License	The license the software is released under, e.g. GNU General Public License
Programming languages used	Programming languages used to write the application, e.g. C++, Objective-C, etc.
Compatible Operating Systems	Name and versions of operating systems the software can be run on
Protocols Used	The Internet Protocols used in this application, e.g. Hypertext Transfer Protocol (HTTP)
Protocols Provider	Other software that also use the same protocols

Table 2: Freebase Data Schema for File Formats (excerpt)

Extension	Common extension of this file format
Genre	Categorisation of formats, e.g. *Audio file format* or *Executable*
Creation Date	The date when the format was created / published
Written By	Link to software applications that can **write** this file format
Read By	Link to software applications that can **read** this file format
Used On	A list of operating system platforms the format is commonly used on
Format Creator	The organisation or individual creating the format
Magic	The magic number (identifier) of this file format, e.g. GIF89a for GIF images
MIME Type	The MIME type of the format
Contained By	The container format this format is usually contained in
Container For	Others formats this format is a container for; e.g., CSO is a container format for compressed ISO images
Extended From	Any other format this format is based on / derived from
Extended To	Any other format that extends on this specific format

packages, used in many Linux distributions, e.g. Debian[7]. In these operating systems, software applications (and components) are normally made available in a specific package format, which is in most cases a specific compressed container format. The package contains the actual software application, as well as control information for the installation process of the package. As such, it provides e.g. scripts that should be run after the software application is extracted to the system, e.g. to perform other changes on the system. One example is the creation of a specific user that would execute a package that provides a server program. Furthermore, control information in the packages provides details on the dependencies of that package. It might e.g. define that for a web server package to be installed, also the Java runtime environment is required. The package manager then automatically handles acquiring and installing also these dependencies.

In these package based operating systems, there is generally a universe of packages that can be installed, and that are known to the package manager. Further, there is the concept of a virtual package, which can be seen as a place-holder package for other (real) packages that then provide the functionality. This concept is also reflected e.g. in CUDF (Common Upgradeability Description Format [15]), which is a format used to describe installation and upgrade paths.

[7]`http://www.debian.org/`

Examples of such virtual packages are e.g. "web-browser", or a "java-runtime", and a "c-compiler". These packages then are provided by specific implementations and from the dependency structures defined in the packages, different implementations can be interchanged. For the "java-runtime" package, providers might be OpenJDK[8], Oracle Java[9], or the Cacao Virtual Machine[10]. These packages provide the same functionality according to the Java Virtual Machine specification, but might greatly differ in regards of their implementation and license. One requirement might e.g. be that the used package should have a license that allows obtaining and modifying the source code, to allow modifications in case a changed system environment requires that.

In order to obtain a knowledge base for the software package, we implemented a tool that gathers the virtual packages and their providers for a specific version of distributions of a Linux system. In principle this tool is based on the Debian package system, and thus covers also operating systems based on Debian, such as Ubuntu[11] or Linux Mint[12].

In total, on a current Linux Ubuntu distribution, around

[8]`http://openjdk.java.net/`
[9]`http://www.java.com`
[10]`http://www.cacaojvm.org`
[11]`http://www.ubuntu.com/`
[12]`http://www.linuxmint.com/`

2.000 virtual software packages that have more than one provider can be identified. Not all of these are actual software applications, some are also just components, i.e. virtual packages that are providing libraries that are in turn used in other applications to built end-user applications. Such libraries can e.g. be components for GUI programming, or libraries that allow interfacing with a specific hardware.

4.3 Representation of Knowledge Bases

As a representation format for our knowledge bases, we opted for using ontologies, specifically the Web Ontology Language (OWL) [13], a widely used knowledge representation language. OWL is intended to augment the Resource Description Framework (RDF), and provides formal semantics, as well as RDF/XML-based serialisations. The reason for choosing this representation is that on the one hand, OWL defines several convenient mechanisms to query the knowledge base. Queries can as such be formulated via OWL Description Logic (OWL-DL), or the graph query language SPARQL [1]. Another motivation for choosing OWL ontologies is that the model to represent a system (cf. Section 3), a part of the previously mentioned process context model, itself is authored using the Web Ontology Language. Using OWL for the knowledge bases representation thus simplifies cross-model queries and reasoning.

Freebase provides an API to query the online content. However, we opted to store the data locally for a number of reasons. First of all, the local storage allows for a more efficient querying of the data, as potentially many subsequent queries need to be sent. Furthermore, we also combined the Data from Freebase with information on Formats from PRONOM, by a simple approach of matching along the file extension and MIME Type. Finally, local storage allows us to represent the knowledge base in a form that enables easy automatic reasoning and discovery of migration paths. We therefore developed the ontology that is depicted in Figure 5. The major elements in there are Formats, Tools and Registries. These are further utilised to perform certain actions, such as migration.

For the second knowledge base obtained from the Linux Package manager, we opted to represent this in CUDF (Common Upgradeability Description Format). CUDF is also utilised in the context model presented in Section 3, where it serves as one domain-specific ontology representing package dependencies. A representation of the concepts of CUDF is given Figure 6. The main information entities are a Package and the VirtualPackage; there is a wealth of relations defined, such as depends, conflicts, etc.

In CUDF, virtual packages can be considered to be a kind of categorisation of the concrete packages, similar to the genre provided by Freebase. If we encounter a certain package, we can thus simple query which virtual packages it provides, and then find other packages that provide this virtual package. Alternatively, if the model already uses a virtual package to model explicit what functionality is required, we can query to replace that specific provider.

5. IDENTIFYING PRESERVATION ACTION POSSIBILITIES

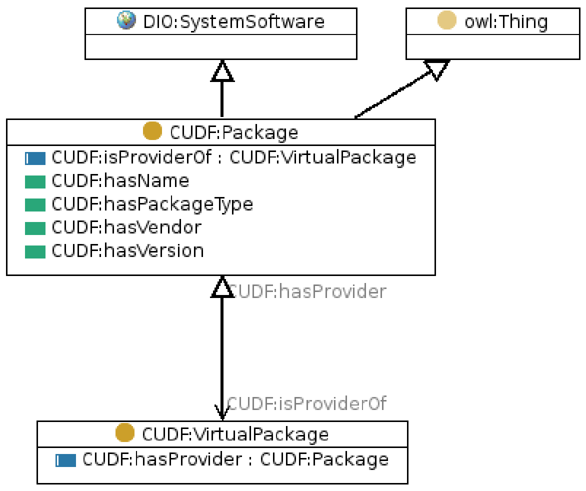


Figure 6: Concepts of CUDF

The alternative identification application currently considers file format migration and software application emulation. We will describe these two in detail below.

5.1 Software Application Emulation

For each software involved in the process (SystemSoftware or Artifact concepts in the Context Model), a software application emulation is proposed, by identifying software that is equivalent to the currently employed applications.

This approach identifies software replacements for a specific software application at risk. Such a risk might be a lack of future support, incompatibility with other components of the process, or that the license the software is published under is prohibitive for the future use or preservation of the system. Using the knowledge base obtained from Freebase, we are able to retrieve migration path information in a structured way. We can e.g. propose the migration of a proprietary word processor file format to a more standard format. Depending on whether we need just read access or also write access to the artefact, different conversions will be available – in general, there will be more support for reading a specific format, thus if this is the only requirement, we will be able to identify more potential alternatives.

The proposed alternative will also take into account which changes in the software stack are needed. To this end, a prototype implementation of a package dependency solver for Linux distributions is being developed, which will be utilised to identify the changes in the software stack. Once a specific file format is identified, consulting the dependency solver will notify us whether the software stack used currently is sufficient to also work with the new file format, or new software needs to be installed. This can in turn mean that a specific software that was previously used to manipulate a digital object is not needed anymore. This may then be removed, and the dependency solver will also be used to determine which other software components that were only needed by the removed application can as well be removed.

5.2 File Format Migration

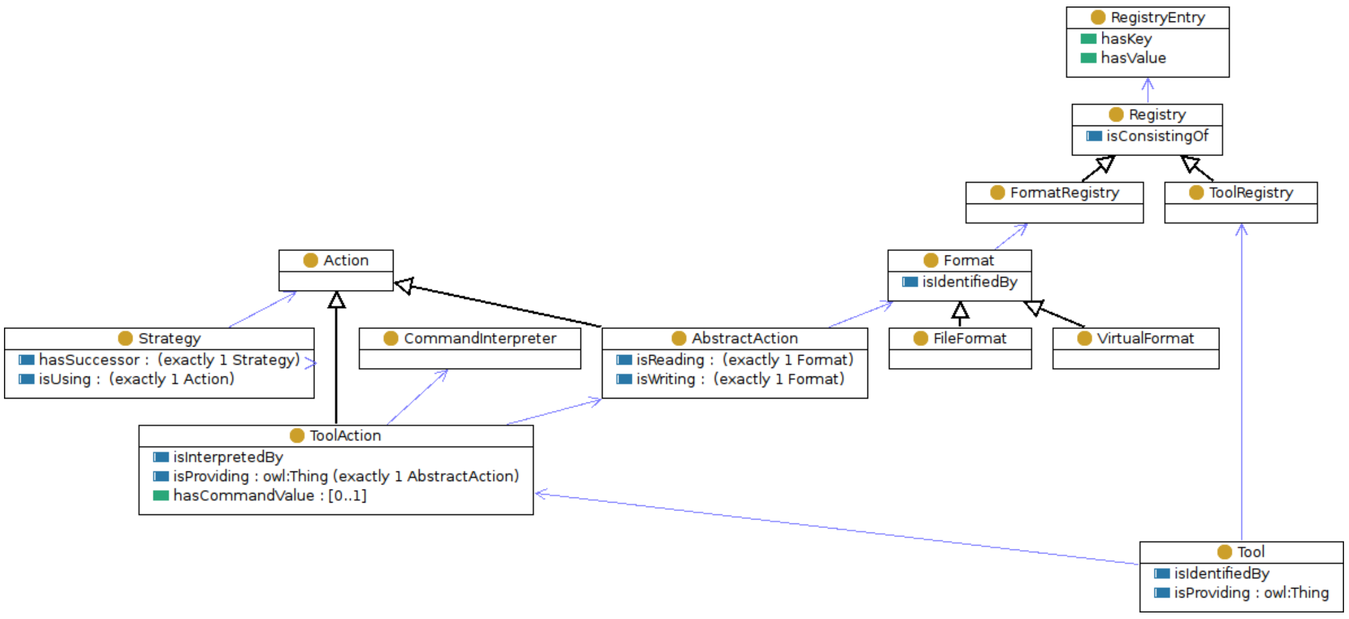


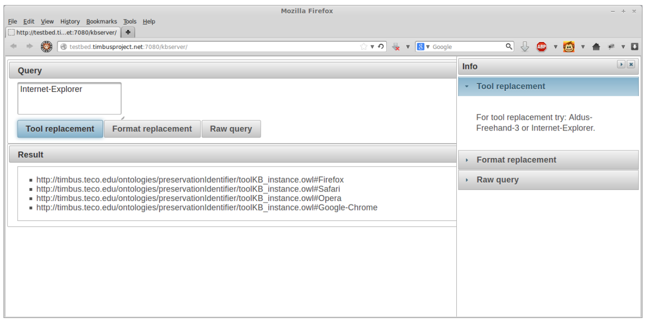
Figure 5: Ontology to store information on migration tools

For each data object (*Artifact* concept in the Context Model) that is either produced or consumed in the process and for which the data format is at risk of becoming obsolete (e.g. a proprietary format for which the vendor support might end), an alternative for ensuring long-term access to this data object has to be produced.

Firstly, once a file format is identified to be at risk and should be replaced, an alternative format providing similar functionality has to be identified. Identifying a similar format can be automated by utilising the *genre* information present in the File Format schema (cf. Section 4), with the straight-forward approach being to identify formats in the same genre, and then select those which are connected via a format migration path using the available software tool migration capabilities.

Secondly, if the new format is not also supported by the software currently available at the system, also the current software setup needs to be modified. In this case, it is need to identify the required changes for the steps in the process that access the files in the old format, and potentially replace the current software applications with different applications that can work with the new format we migrated to. This affects software that reads, writes or renders these files.

Data objects could be both interpreted by humans (in a human processing step in the process, where the human takes decisions based on the content of the data object, or augments/modifies the data object), or by software. In the case of a human task, the exact rendering of the data object e.g. on the screen is important. It is therefore important to select an emulation strategy that preserves this property most faithful. In the case of a machine task, preservation of the data object has to go in hand with ensuring the software can still process the data object, but rendering capabilities are of less importance.



- http://timbus.teco.edu/ontologies/preservationIdentifier/toolKB_instance.owl#Firefox
- http://timbus.teco.edu/ontologies/preservationIdentifier/toolKB_instance.owl#Safari
- http://timbus.teco.edu/ontologies/preservationIdentifier/toolKB_instance.owl#Opera
- http://timbus.teco.edu/ontologies/preservationIdentifier/toolKB_instance.owl#Google-Chrome

For tool replacement try: Aldus-Freehand-3 or Internet-Explorer.

Figure 7: Preservation alternative for replacing "Internet Explorer" by alternative software.

5.3 Online Query interface

The knowledge bases obtained from Freebase and the Linux Package Universe can be queried online for preservation identification alternatives, as seen in Figure 7, which depicts a potential replacement of Internet Explorer by alternative web browser software such as Firefox, Safari, Opera or Google Chrome. In the future, we will also provide an API that could be utilised by other services needing information on file formats and software tools.

5.4 Alternative Identification Output

The output of the alternative identification module is a set of possible preservation action alternatives. These alternatives will then have to be analysed by a digital preservation expert regarding their feasibility and suitability, who will then select those actions that best fit his requirements.

Specifically, each alternative contains a modified version of the technology view of the system, modelled via the process context meta-model, and a list of changes that were done to arrive at this new system, from the original instance. The

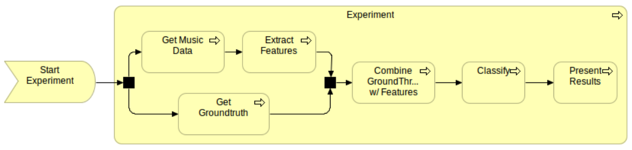


Figure 9: Process model of the eScience experiment

context models, original and modified, are OWL ontologies, as detailed in Section 3. The list of changes is provided by the means of the OWL version of the PREMIS preservation data dictionary. Specifically, the “Event” entity is used to link entities (linkingSourceObject and linkingOutcomeObject), together in a softwareReplacementEvent or formatMigrationEvent. The source and outcome objects are generic PremisEntity elements, of which, via the mapping of the PREMIS extension to the core ontology in the context model, specific software artefacts are instances of. An example of such a change list can in Figure 5.4.

6. USE CASE APPLICATION

The use case we want to investigate in detail is an e-Science experiment in the domain of machine learning. Specifically, it tests the usefulness of a method for automatically classifying items in a music collection into a set of predefined categories corresponding to music genres, by computing the accuracy of the classification (i.e. for how many songs the algorithm can detect the correct genre). It is performed by a researcher which aims to collect performance metrics for classification and make comparisons to the state of the art. The motivation for performing the preservation of such a process is related to any possible challenges to the results that can be made by members of the research community. Thus, by preserving such process, the provenance and authenticity of the results can be proven, and the process can easily be repeated on different data, or with altering the parameters, at a later stage, as well.

A process model is depicted in Figure 9. First, music data and a ground truth (“gold standard“) of the genre assignment are acquired from external providers. Then features (numerical representations) are extracted from the music files. These are combined with the gold standard, and converted to a different format, before a classification model is learned. Finally, the performance of that model is evaluated. This process is described in much more detail in [11] and [12].

Figure 10 depicts a graphical representation of the technological infrastructure of the process. Central to the process is the Taverna Workflow engine, in which the process is modelled, and which orchestrates the execution. The audio feature extraction, as well as the format conversion are implemented in Java, and require a version 6 Java runtime to be executed. The machine learning software is provided by the open source Toolkit ”Weka“ [19], which as well requires Java. Also smaller helper applications to fetch music data and ground-truth are implemented in Java as well. Java is provided in this setup by the Oracle Java 6 implementation, which comes with a restrictive license that disallows redistribution among other things. Important for the pro-

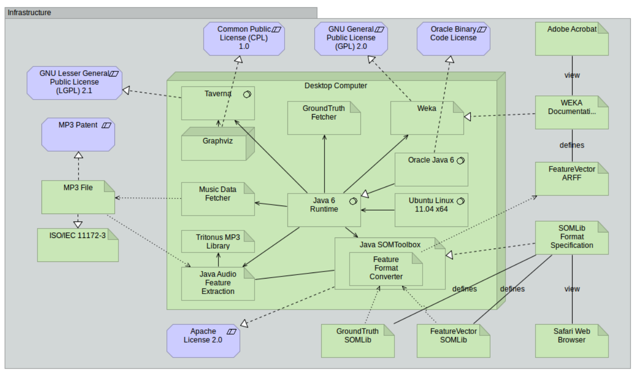


Figure 10: Technical infrastructure model of the eScience experiment

cess are also the File Formats utilised - on the one hand there is MP3 which is used to encode the audio files, and then there are a series of custom text formats, such as the SOMLib and ARFF Formats. These are defined in respective specification documents, which are authored in HTML and Adobe Acrobat respectively. On the current setup, the closed-source software Safari and Adobe Acrobat are used to view them.

To be able to preserve the technical environment of the process, the following automatic alternatives to the current system can be identified

Software Emulation. Oracle Java is restrictive in regards to source code and redistribution, thus it is preservable to replace the Java runtime implementation by other means. Through the knowledge base on Package Alternatives, we can identify OpenJDK as an open-source alternative. Via the Freebase knowledge base, we can identify similar alternatives. The SOMLib documentation is in the current setup displayed via the Safari Browser. This might not be an ideal candidate for long-term preservation, as no source code is available, and it is thus more difficult to adapt to a changed environment. An automatic proposal would yield e.g. Firefox or the Chromium browser as alternatives. A similar issue arises with the documentation of WEKA, which is in PDF format; the alternative proposals yield the open-source tools Okular and Evince as alternatives.

File Format Migration. The feature extraction service currently takes various input formats, such as WAVE, MP3 or FLAC. In the current experimental setup, MP3 is used, which is processed with the help of a third-party library *tritonus*[13]. This library is however not actively developed since 2003, and frequently has errors with MP3 files that have a slightly unusual encoding. Furthermore, MP3 is partially protected by a patent, and that might cause problems for certain preservation actions to be applied in the future. It might thus be beneficial to change to a different file format. Format replacement would suggest e.g. a conversion to WAVE PCM, using e.g. the software *mpg123*. Of course,

[13]http://www.tritonus.org/

```
<ClassAssertion>
  <Class IRI="http://id.loc.gov/ontologies/premis.rdf#Event"/>
  <NamedIndividual IRI="[serviceLocation]/[identifier]/SoftwareReplacement"/>
</ClassAssertion>
<ObjectPropertyAssertion>
  <ObjectProperty IRI="http://id.loc.gov/ontologies/premis.rdf#linkingSourceObject"/>
  <NamedIndividual IRI="[serviceLocation]/[identifier]/SoftwareReplacement"/>
  <NamedIndividual IRI="[originalModelURI]#OracleJava1.6"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
  <ObjectProperty IRI="http://id.loc.gov/ontologies/premis.rdf#linkingOutcomeObject"/>
  <NamedIndividual IRI="[serviceLocation]/[identifier]/SoftwareReplacement"/>
  <NamedIndividual IRI="[serviceLocation]/[modifiedModelURI]#OpenJDK1.6"/>
</ObjectPropertyAssertion>
```

Figure 8: Example of the output describing the changes made to the system by replacing *Oracle Java 1.6* with *OpenJDK 6*

several other tools are available to do this specific migration, such as *lame*, *ffmpeg*, or applications with a graphical user interface such as *mplayer*. In addition, several other target formats are proposed, such as the Free Lossless Audio Codec (FLAC), for which similar conversion tools are applicable.

The current documentation of the SOMLib format in HTML might be risky, as HTML is still an evolving standard (e.g. to currently HTML 5), and it is has shown to be difficult to exactly preserve the behaviour of Documents, especially across different implementations of web browsers. A format migration of HTML would identify PDF as a suitable candidate, using e.g. the tool *wkhtmltopdf*. The software stack also needs to be updated, as we now don't need an HTML viewer anymore.

In Figure 11, we can see one potential candidate modification to the view-paths in the process. We modified specifically viewing the HTML and PDF files, and converted the file format of the music data to WAVE, thus requiring the new application "mpg123". Modifications to the specifications were done manually, the licenses can be determined automatically for some software applications, as this information is provided for most Linux Packages, and for some software applications registered in Freebase.

7. CONCLUSIONS AND FUTURE WORK

Knowledge bases on file formats and software applications are an important aspect in digital preservation, especially in the phase of preservation planing, where alternatives of preservation actions need to be identified and evaluated. Existing knowledge bases often lack in the depth and freshness of information provided, as maintaining them is a tedious task. In this paper, we thus presented an approach to obtain knowledge bases on file formats and software applications from repositories such as the community maintained linked open data source Freebase, as well as Linux package repositories. We on the one hand offer these knowledge bases in a publicly available API that can be used by digital preservation solution providers. Further, we also presented a prototypical implementation of a preservation alternative discovery and identification service that leverages these knowledge bases. We demonstrated the usefulness of this approach on a use case evaluation.

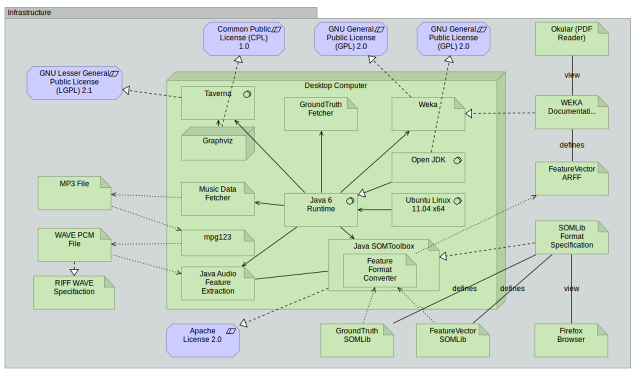


Figure 11: Technical infrastructure model of the eScience experiment, after applying preservation actions to migrate file formats and using alternative software

Future work will focus on fine tuning the software and file format knowledge bases obtained from the online repositories, and improve the alternative identification approach. Potential future extensions to the knowledge base and alternative identification are:

- Emulation of the execution environment, i.e. utilising emulators that mimic the functionality and behaviour of the hardware and software environment (operating system). Information on hardware is available in the context model, thus only information on emulators is mission.
- Migration of the software to a different platform. This can be as simple as just compiling an otherwise platform independent software for the different platform, or in worst case be a complete rewrite of the software. Freebase might offer enough data for this, as information on programming languages utilised for a software, and compilers availability for certain platforms, is available.

Furthermore, we will be applying the alternative identification service to more use cases from the TIMBUS project, among others a use case on monitoring of large civil engineering structures and from the e-Health domain, as well as on other use case from the domain of scientific workflows.

Acknowledgements

This work has been co-funded by COMET K1, FFG - Austrian Research Promotion Agency and by the TIMBUS project, co-funded by the European Union under the 7th Framework Programme (FP7/2007-2013) under grant agreement no. 269940.

8. REFERENCES

[1] SPARQL query language for RDF. Technical report, World Wide Web Consortium, Jan. 2008.

[2] G. Antunes, M. Bakhshandeh, R. Mayer, J. Borbinha, and A. Caetano. Using ontologies for enterprise architecture analysis. In *Proceedings of the 8th Trends in Enterprise Architecture Research Workshop (TEAR 2013), in conjunction with the 17th IEEE International EDOC Conference (EDOC 2013)*, Vancouver, British Columbia, Canada, September 9-13 2013.

[3] C. Becker, H. Kulovits, M. Guttenbrunner, S. Strodl, A. Rauber, and H. Hofman. Systematic planning for digital preservation: Evaluating potential strategies and building preservation plans. *IJDL*, 10(4):133–157, 2009.

[4] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA, 2008. ACM.

[5] T. Brody, L. Carr, J. M. N. Hey, A. Brown, and S. Hitchcock. Pronom-roar: Adding format profiles to a repository registry to inform preservation services. *IJDC*, 2(2):3–19, 2007.

[6] R. Graf and S. Gordea. Aggregating a knowledge base of file formats from linked open data. In *Proceedings of the 9th International Conference on Digital Preservation (iPres 2012)*, pages 293–294, Toronto, Canada, October 1-5 2012.

[7] S. Granger. Emulation as a digital preservation strategy. *D-Lib Magazine*, Vol. 6 (10), 2000. `http://www.dlib.org/dlib/october00/granger/10granger.html`.

[8] T. O. Group. *ArchiMate 2.0 Specification*. Van Haren Publishing, 2012.

[9] T. S. S. Institute. Approaches to software sustainability. Website. `http://www.software.ac.uk/resources/approaches-software-sustainability`.

[10] D. B. Marcum. The preservation of digital information. *The Journal of Academic Librarianship*, 22(6):451 – 454, 1996.

[11] R. Mayer and A. Rauber. Towards Time-resilient MIR Processes. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR 2012)*, pages 337–342, Porto, Portugal, October 8-12 2012.

[12] R. Mayer, A. Rauber, M. A. Neumann, J. Thomson, and G. Antunes. Preserving scientific processes from design to publication. In P. Zaphiris, G. Buchanan, E. Rasmussen, and F. Loizides, editors, *Proceedings of the 16th International Conference on Theory and Practice of Digital Libraries (TPDL 2012)*, volume 7489 of *Lecture Notes in Computer Science*, pages 113–124, Cyprus, September 23–29 2012. Springer.

[13] W. OWL Working Group. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-overview/`.

[14] J. Rothenberg. *Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation*. Council on Library and Information Resources, January 1999.

[15] R. Treinen and S. Zacchiroli. Description of the CUDF Format. Technical report, 2008. http://arxiv.org/abs/0811.3621.

[16] J. van der Hoeven, B. Lohman, and R. Verdegem. Emulation for digital preservation in practice: The results. *IJDC*, Vol. 2 (2):123–132, 2007.

[17] R. van Diessen. Preservation requirements in a deposit system. Technical report, IBM/KB Long-Term Preservation Study Report Series #3, IBM Netherlands, Amsterdam, 2002.

[18] C. Webb. *Guidelines for the Preservation of Digital Heritage*. National Library of Australia, 2005.

[19] I. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.