PERICLES - Promoting and Enhancing Reuse of Information throughout the Content Lifecycle taking account of Evolving Semantics

[Digital Preservation]

DELIVERABLE 5.3

Complete Tool Suite for Ecosystem Management and Appraisal Processes

GRANT AGREEMENT: 601138

SCHEME FP7 ICT 2011.4.3

Start date of project: 1 February 2013

Duration: 48 months

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
	Dissemination level	
PU	PUBLIC	X
PP	Restricted to other PROGRAMME PARTICIPANTS (including the Commission Services)	
RE	RESTRICTED to a group specified by the consortium (including the Commission Services)	
CO	CONFIDENTIAL only for members of the consortium (including the Commission Services)	

Revision History

V #	Date	Description / Reason of change	Author
V0.1	28.09.16	Initial version of final draft, as compiled from Google Doc contributions.	AC
V0.2	28.09.16	Further content added	AC
V0.3	28.09.16	Further content added	AC
V0.4	29.09.16	Finalising the document	AC, JB
V1.0	30.09.16	Document to KCL for submission	JB
V1.1	04.10.16	Small changes in response to internal review.	FC, AC

Authors and Contributors

Authors

Partner	Name
KCL	Simon Waddington
KCL	Alastair Gill
UEDIN	Adam Carter
UGOE	Johannes Biermann
UGOE	Anna Eggers
ULIV	Fabio Corubolo
ULIV	Jérôme Fuselier
ULIV	Paul Watry

Contributors

Partner	Name
UEDIN	Rob Baxter
CERTH	Efstratios Kontopoulos
ULIV	Maureen Watry

Table of Contents

List of Tables

List of Figures

Glossary

Abbreviation / Acronym	Meaning
API	Application Programming Interface
BPMN	Business Process Model and Notation. A graphical language for describing processes.
Cassandra	A distributed database system which is part of the Apache foundation.[1]
CDMI	Cloud Data Management Interface is a protocol for accessing cloud storage.
Digital Ecosystem (DE)	Network of technical systems, communities, digital objects, processes, policies, and the relations and interactions between them. This is the object of interest that is modelled with the Digital Ecosystem Model ontology.
Digital ecosystem management	Control layer to provide support and manage change in the digital ecosystem and its entities. In the scope of this task, the QA methods are supporting the validation of changes in the digital ecosystem with respect to policies and high value digital media.
Digital Ecosystem Model (DEM)	Ontology developed by the PERICLES project that allows to model Digital Ecosystems: technical systems, processes, digital objects, policies and users to answer and simulate change related questions.
Digital Object (DO)	"Digital objects (or digital materials) refer to any item that is available digitally." (JISC, “Definition of Digital Object”)
ERMR	Entity Repository Model Repository this refers to the T5.1 component.
FA	Functional Architecture
GUI	Graphical User Interface
Content-based (or intellectual) appraisal	Acquisition and retention decisions or assignment of value based on the content of the digital entities themselves.
iRODS	The Integrated Rule-Oriented Data System (iRODS) is an open source data management software that virtualizes data storage resources. The application can be used for data management infrastructure building.
LDAP	Lightweight Directory Access Protocol - standard protocol for distributed directories

[1] http://cassandra.apache.org/

MICE	Model Impact Change Explorer. Software component from PERICLES for analysing change.
MQTT	Message Queue Telemetry Transport is a protocol for machine to machine communication.
PE	Policy Editor component from PERICLES.
PET	PERICLES Extraction tool (PET) for extracting significant environment information.
PoC	Proof of Concept
Policy	Used in very diverse situations both in English, and in IT. A policy is a plan that defines the desired state inside an ecosystem. A policy describes the 'what' (guidelines) and not the 'how' (implementation). Policies can be described in varying degrees in natural language or in a formal language. Policies can also be used to represent the legal requirements and aspects of an ecosystem.
PC	Process Compiler. Transform a set of single processes into a combined process.
Quality Assurance (QA)	*"Program for the systematic monitoring and evaluation of the various aspects of a project, service, or facility to ensure that standards of quality are being met"* (Webster)
RDBMS	Relational database management system.
RDF	Resource Description Framework. A versatile data model in which assertions are expressed as *subject-predicate-object* triples.
ReAL	The Resource action language describes transformative actions on RDF based models. Enables rule functionality on ontology.
REST	Representational State Transfer. A design style for networked applications, usually implemented with HTTP.
SPARQL	SPARQL Protocol and RDF Query Language. An RDF query language
Technical appraisal	Decisions based on the feasibility of preserving the digital objects. This involves determining whether digital objects can be maintained in a reusable form and in particular takes into account obsolescence of software, formats and policies.
Unit Test	Technique that originates from software engineering for modular testing of source code

1. Executive Summary

This document provides a summary of all tools and approaches developed in PERICLES by September 2016, and describes the context in which these results can be used. It focuses primarily on tools and approaches developed in support of digital ecosystem management (WP5), but links these to tools, models and approaches developed elsewhere in the project, and the project's functional architecture described in the deliverable D3.5 Full Report on Digital Ecosystem Management.

This present deliverable covers 4 models, 15 software tools and 5 approaches classifying them according to the relevant phase of the functional architecture such as Model Development, Registration and Operation.

The first part of this document describes the set of tools developed in WP5, including the Entity Registry Model Repository, the EcoBuilder, the Policy Editor and the Technical Appraisal Tool, while the second part summarises key approaches and models such as the policy and QA models or the approach behind the model-driven change management and appraisal.

The Entity Registry and Model Repository (ERMR) is a piece of software for storing and querying models and digital objects and a central component of the PERICLES WP6 test-bed for registering models and metadata. Policies which operate on the data store help to manage the stored data and those policies can invoke external programs. The notification system issues events onto a message queue, so connections to third party components and workflows are possible. The whole tool is designed to be flexible and scalable.

The Policy Editor makes it easier to create policies using templates. The policy level and integration level of the Policy Editor is flexible. It can be used as a stand-alone tool or integrated with other tools, i.e. the templates can be kept within other systems (such as models in the ERMR) or use simple text files. The output can also be text files or integrated into other systems, e.g. by sending policies or associated processes directly to an execution engine. This is possible by providing custom adapters.

The Appraisal Tool is a web-based tool for appraisal of technical risks. It implements a theoretical approach described in section 8. The tool uses ecosystem models as input, and makes use of data harvested from external sources such as Google, software repositories and Wikipedia. The tool features a user interface that presents the complex risk-impact-proximity information in various different views to assist the user in both determining risks in their collections and analysing risks to specific digital objects.

The EcoBuilder tool can be used to help scenario experts create Digital Ecosystem Models (DEMs). It can output the model as file or offers a direct connection to ERMR to store the models there. Using the tool ensures that a valid model is created by providing templates for the ontology constructs. Details and domain specific extensions can be added afterwards, if required. It provides a GUI and an API. The API enables the possibility to integrate the tool, e.g. for connecting it to a notification system to populate change into the DEM model.

Approaches for Policy, Quality Assurance (QA), and Change Management are described. The final model for policy and QA representation has been implemented in the DEM, together with new guidelines for policy derivation, QA and compliance. Rule-based change management for Digital Ecosystems (DE) is illustrated with use-case examples representing policies and their implementation, supporting automated change management. Experiments with semantic and user community change observation and QA have been presented. Finally, we have done a Proof of Concept rule implementation of the above-mentioned approaches. This contribution has shown how policy, QA and change management can be automated for the different entities of the PERICLES ecosystem.

We have conducted both a study and classification of appraisal criteria in the context of evolving digital ecosystems, as well as developing methods and practical tools for specific criteria. We build on the catalogue of appraisal criteria from D5.2 Basic Tools for Digital Ecosystem Management, and describe an approach based on the principle of evaluating specific criteria and providing this information in an easily digestible form to a curator. In the technical appraisal work, we have adopted a data-driven predictive approach to evaluating sustainability of complex digital objects. The approach is supported by the Appraisal Tool mentioned above.

These approaches support the idea that the various tools developed in this project can be used together to provide novel solutions to digital preservation problems. This remains a challenging area, but the idea behind the PERICLES tools and approaches is that they can be re-used and built upon to complement existing and future preservation processes and infrastructure.

2. Introduction & Rationale

PERICLES is ultimately about promoting and enhancing reuse of digital information. This is a challenging issue and it cannot be solved through one piece of software or a single approach. The project has therefore been about exploring various approaches to this challenge, and building software tools to help address particular aspects of it.

A key risk in being able to keep digital information re-usable over time is the occurrence and impact of *change.* This is also reflected in the project's title, which emphasises content lifecycle and evolving semantics. Change includes both unmanaged and managed change; that is, both external change that we need to respond to, and the controlled changes that can be made in response.

This high-level approach has been labelled as *model-driven preservation*, and this is a common theme which relates the various tools and approaches that the project has created and investigated. Another important concept is that of a *digital ecosystem* analogous to a biological ecosystem with its many interacting systems. The idea here is that in order to preserve digital content for reuse, it must be possible to describe the content itself, the systems being used to preserve it and the wider external environment to which the content relates, including both the environment in which the content was created and that in which it could be reused in future. This combination of digital objects, systems, preservation processes and the wider environment constitute the digital ecosystem.

Relating the ideas of *change* and *digital ecosystem* we note that the ecosystem itself is in a state of constant change, and this is one of the main rationales for adopting a model-driven approach. Describing in concrete models the many relationships between different types of entities allows understanding the impact of change on all related entities. Moreover, we note that with a changing ecosystem, static models will not suffice, and several of the tools and approaches described here are designed to help modify and create models.

PERICLES never sought to build a self-contained preservation system, or propose a monolithic approach that would require the use of all of the tools developed in the project. Instead we have aspired to develop the various tools to be interoperable, both with each other, and with other existing or future preservation infrastructures. The project test-bed (developed in WP6) provides both a platform for demonstrating the interoperability of the tools and the validity of the approaches, and a runtime platform which can be used as an execution layer for automated processes involving multiple components. The interoperability is also supported by the ERMR, a component which can act as a central registry and model repository.

Rather than developing a full system, we have created a Functional Architecture that describes how we envisage that the project outputs could be used together.

In this document, we summarise all of the tools and approaches that have been developed in the project and describe the context in which they can be used. The inter-related nature of the project means that many of these tools and approaches have already been described elsewhere, so the primary focus in this document is on tools and approaches that have been developed in WP5.

2.1. The PERICLES Functional Architecture

The PERICLES Functional Architecture (FA) is described in full in the deliverable D3.5 Full Report on Digital Ecosystem Management. The FA is an idealised model on how the different PERICLES components can work together in a model-driven way. The FA is a high-level view on the components, but not a technical architecture. The FA starts from the concepts of *Model* and *Digital Ecosystem* as depicted in Figure 1. This figure also shows many of the outputs from the PERICLES project in the grey boxes.

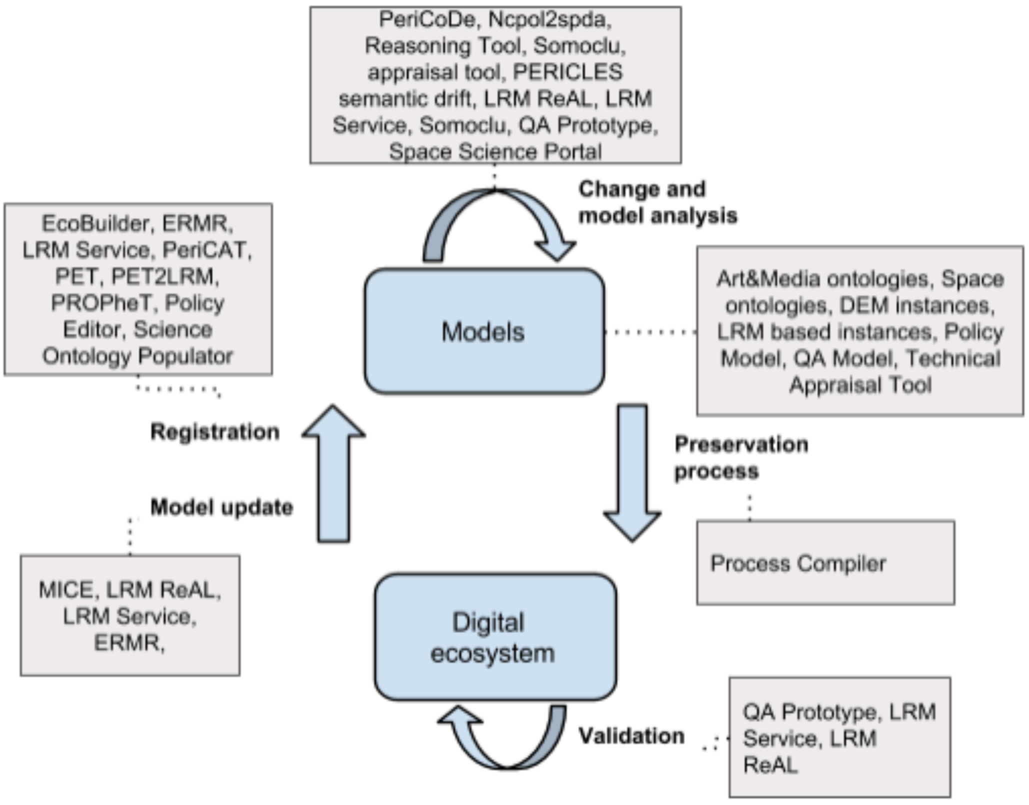

PeriCoDe, Ncpol2spda, Reasoning Tool, Somoclu, appraisal tool, PERICLES semantic drift, LRM ReAL, LRM Service, Somoclu, QA Prototype, Space Science Portal
Change and model analysis
EcoBuilder, ERMR, LRM Service, PeriCAT, PET, PET2LRM, PROPheT, Policy Editor, Science Ontology Populator
Models
Art&Media ontologies, Space ontologies, DEM instances, LRM based instances, Policy Model, QA Model, Technical Appraisal Tool
Registration
Model update
Preservation process
Process Compiler
MICE, LRM ReAL, LRM Service, ERMR,
Digital ecosystem
Validation
QA Prototype, LRM Service, LRM ReAL

Figure 1: A high-level overview of the PERICLES functional architecture (in blue) and related PERICLES components (in grey)

D3.5 goes into more detail in explaining this cycle and the static view it represents, the registration of digital ecosystems, change and model analysis and preservation process and validation. In the present report, we use these descriptions to map them to the components developed in the project.

2.2. PERICLES Components

In this section we provide an overview of the many PERICLES components. We classify them according to the phase in the Functional Architecture in which they are likely to be used and label them as a model, an approach, or as software. In the case of software, we describe whether the interface is graphical or programmatic.

We can identify three main phases from the FA:

- Model development (phase M): In the FA it is part of the static view and registration of ecosystems; here the ecosystem in question is analysed, the entities are identified and information is extracted. This is mostly done with UI-based tools.

- Registration and configuration phase (phase R): This means that models and entities get instantiated and filled with concrete values. UI-based and process or API based tools are involved. On the FA this is the registration of digital ecosystems
- Operation (phase O): Here the models are being actively used for ecosystem management in particular for model analysis and change management. Mostly API based tools are involved here. On the FA this corresponds to the change and model analysis and preservation process and validation views.

Each tools in this tool suite is either:

- a software tool, which has a GUI and/or an API (programming interface),
- a model, which is a structured description of a "thing" or "type of thing", or
- an approach, which is a product of experiments, and evaluations of topics and ideas. Approaches often involve software tools, some of which could have more general use, but others are primarily created to experiment with the approach.

Table 1, below, summarises all developments within PERICLES and gives a reference to the corresponding description. The items which are described in this deliverable are highlighted in bold.

PERICLES component	Short description	Phase	What	Reference
Art & Media Domain Ontologies	Models for the WP2 Art & Media case studies	M	Model	WP2/T2.3.3 D2.3.2
Digital Ecosystem Model	Model for describing digital ecosystems	M, R, O	Model	WP3 T3.5.1, T3.5.2, D5.1.1, D5.2 and D3.5
EcoBuilder	**Tool for creating DEM instances**	**M, R**	**GUI + API**	**This document chapter 6**
Entity Registry Model Repository (ERMR)	**Used for storing and registering entities and models**	**R, O**	**API + GUI**	**This document chapter 3**
Linked Resource Model	Model for describing dependencies	M, R, O	Model	WP3 T3.2 D3.2, D3.3, D3.4
LRM ReAL	Language for change management plus interpreter	**R, O**	API	WP3 D3.3, D3.4
LRM Service	Service for working with LRM	**R, O**	API	WP3 D3.4

	based models			
Model Impact Change Explorer (MICE)	Component responsible for visualising the changes applied to an entity	**O**	GUI	WP6/T6.3.3 D6.4, D6.6
Ncpol2spda	Theoretical research on quantum-like behaviour patterns in evolving data.	**R, O**	API	WP4/T4.5 D4.5
PeriCAT	A framework for Information Encapsulation techniques	**R, O**	GUI + API	WP4 D4.2
PERICLES Semantic Drift	A set of software metrics and associated tools for detecting, measuring and evaluating semantic drifts in ontology sets.	**R, O**	Approach	WP4/T4.3, T4.4, T4.5 D4.4, D4.5
PeriCoDe	Algorithms for detecting high-level visual concepts in images	R, O	Approach/API	WP4/T4.3, D4.3
PET	Tool for extraction of significant information from the environment	**R, O**	GUI	WP4/T4.1 D4.1
PET2LRM	Transforms PET output into LRM compatible descriptions	M, R, O	API	WP4/T4.3-4-5
Policy Editor	**Edit pre-defined policy templates**	**R, O**	**GUI**	**This document chapter 4**
Policy Model	**Policy model and policy derivation**	**R, O**	**Approach**	**This document section 7.1**
Process Compiler (PC)	Takes a representation of a process model and transforms it into another form that can be executed by the workflow engine.	**O**	API	WP6/T6.2 D6.4, D6.6
PROPheT	Tool for populating domain ontologies (here art & media)	**M, R, O**	GUI	WP4 D4.3

QA Model	**Approach for verification of the correct policy application**	**R, O**	**Approach**	**This document chapter 7**
QA Prototype for Change Management	**Methodology for change and conflict management, semantic change of communities**	**R, O**	**Approach**	**This document chapter 7**
Reasoning Tool	Semantic reasoning for contextual content interpretation	O	Approach	WP4/T4.5
Science Ontology Populator	Tool for populating parts of the WP2 space ontology	M, R, O	API	WP6/T6.1, D6.6
Somoclu	a generic tool to study semantic fields, concept drifts, evolving semantics, and contextual data.	**R, O**	API	WP4/T4.3, T4.5 D4.1, D4.4, D4.5
Space Science Domain Ontology	A model from the WP2 space case studies	**M**	Model	WP2/T2.3.3 D2.3.2
Space Science Portal	A web portal which allows visualising, exploring, querying and augmenting the semantic model in use and is greatly influenced by the Topic Maps related standards.	**O**	GUI	WP6 D6.6
Technical Appraisal Tool	**Performs risk analysis based on statistical analysis of external datasets as well as ecosystem models**	**R, O**	**GUI**	**This document chapter 5**

Table 1: A full list of PERICLES components

2.3. Context of this Deliverable Production

2.3.1. What to expect from this document

As the title of the deliverable suggests, this document aims at describing the suite of tools developed in WP5 in support of the digital ecosystem management described in D3.5, how they work, how they can be implemented and where to source them. In addition, the document provides a full description of the different approaches used, their theoretical background and the developed methodology including results from experiments, where applicable.

2.3.2. Relation to other work packages and output

This deliverable is the final deliverable of WP5. It completes the initial report done in deliverable D5.2 and in part D5.1, and updates some of the work described there. It also relates closely to D3.5 which contains the full description of Functional Architecture and the Digital Ecosystem Model. The other relations are listed in table <ref>. Work in WP5 continues to support the wider objectives of the project, and further work related to several of the tools listed above will be reported in deliverables D3.4 and D6.6.

2.4. Document Structure

The document is structured into two main parts. The first part covers all the WP5 tools, which are ERMR, Policy Editor, Technical Appraisal Tool, EcoBuilder. The second part covers all WP5 approaches, namely the final Policy Model and Guidelines, Rule-Based Change Management for Ecosystem and Policy, Quality Assurance for Semantics and User Communities, Prototypes for supporting change in technology, semantics and user communities and the theoretical background of the Technical Appraisal Tool. Appendices can be found at the end of the document, which include API documentation, examples, and other information to complement the main part of the report.

PART ONE
SOFTWARE TOOLS

3. The Entity Registry Model Repository

In this chapter we describe the design and implementation of the Entity Registry Model Repository (ERMR), illustrate how it can be used, and describe where it can be obtained. This work was undertaken as part of Task 5.1.1.

3.1. Background & Motivation

The PERICLES project requires a controlled environment where data integrity, reproducibility, and quality can be secured. In support of this, we have developed and implemented a middleware component responsible for storing models and registering entities called the Entity Registry Model Repository (ERMR). This component provides distributed, robust, and extensible storage capabilities, coupling object stores and triple stores to an easy to use rule system. Such a combination allows for scalable systems that can self-manage, and are not reliant on complex workflow systems or external management frameworks. Relatively simple configurations allow for the distributed ingest, transformation and curation, without the need to have complex scheduling frameworks.

The objective of the ERMR is to build a data management infrastructure to support policy-driven data management and rule-based change management (described in chapter 7), that contributes both the data life cycle and sustainability of data collections and repositories in support of the PERICLES project objectives. Our approach uses modern web foundational technologies to implement a repository that can support data management policies by reporting changes and events to an external rule engine based on, for example, SPIN[2] rules (see D4.4 Modelling Contextualised Semantics) or the LRM service. In this respect, the approach has similarities to the integrated Rule Oriented Data System (iRODS), but with changes in design and implementation to support ontology reasoning on a triple store, and a standardised interface (CDMI) for cloud storage. These are described in greater detail below.

3.2. Use and Functionality

The ERMR provides multiple related pieces of functionality. Its primary role in PERICLES is as a registry and repository for models, but it also offers the functionality of a general repository system designed for the long-term sharing, curation, and preservation of data. It provides secure, distributed object storage that is standards-based, and has been built to support other PERICLES tools and approaches. This software, developed specifically in response to the project's objectives, leverages widely used, web-scale technologies to provide high availability, resilient access to data over geographical distances.

From a user's perspective, the ERMR provides access to both managed (internal) and unmanaged (external) data using a common HTTP standard (Cloud Data Management Interface: "CDMI"), with the ability to organise data into containers (folders or directories), with user specified annotation and

[2] http://spinrdf.org/

labelling of data, whether internal or external.

A user can also use rules, which are executed when anything happens to a piece of data or metadata (such as creating, modifying, deleting), and the rule can be written in any computer language that can access the messaging system. When activated, a rule can trigger any action, such as taking a copy, adding metadata, sending an email alert, etc.

The current implementation is based on the Apache Cassandra database system, with a web-GUI and the CDMI access protocol being provided by Python scripting code in the NGINX framework, which provides web server software. The repository data, metadata, and all control data is contained within the Apache Cassandra database, and is designed to be intrinsically consistent and resilient. In addition to the object store, it is possible to store and retrieve RDF data. RDF data can be manipulated with the SPARQL query language.

It is possible to register external resources and connect them via a URL, providing a way of organising diverse data sources from other services (such as ftp) into containers accessible via a single protocol or the GUI, and enhancing these external resource with user specified metadata.

Access control is provided within the ERMR framework using ACLs. Authentication is provided using either a local username table or via LDAP, and is used to establish a “role”. The access control entries authorise roles (not users), so when a user leaves no change is needed to the data: only to the role table.

A list of essential ERMR capabilities is as follows:

1. Hierarchical object storage
2. Triple store for RDF data
3. Arbitrary user defined attributes on objects and containers (“metadata”)
4. Fine-grained access control
5. Replication across multiple nodes
6. Rule engine
7. Federation

In a PERICLES context, the main uses of this software are as (i) an Entity Registry, that is, a place to register the existence of entities in the digital ecosystem that the PERICLES components can work with, and (ii) a Model Repository, a place to store models (which can act as descriptions of entity classes or describe how entities are related to one another).

In both of these cases (where the models and entity descriptions are based on RDF) the ERMR can store these using its triple store component. This offers the potential for these to be queried using SPARQL (Standard Protocol and RDF Query Language). The ERMR can also be used as a repository for models stored in other formats, such as processes described in BPMN. In this case, the object-store functionality of the ERMR can be used to store and access these models.

In addition to storage and querying functionality, the ERMR supports preservation processes through two important mechanisms: a notification system (which can, for example, provide messages to other components when an entity or model is modified) and a listener daemon that can observe the notification queue and execute scripts in response to events.

The ERMR software actually goes beyond what is required for an Entity Registry or Model Repository and can be used in practice as a digital object repository in its own right, but as mentioned above, we stress that different sets of the PERICLES tools can be used in different contexts to support existing preservation systems.

The ERMR provides both an Application Programming Interface (API) and a Graphical User Interface (GUI). The former has been more widely used in the PERICLES project as it offers a straightforward mechanism for integration with other tools. The existence of the GUI, however, is considered important in terms of usability, particularly in contexts with the use of the ERMR is being explored prior to its integration into a wider preservation system or in contexts where the ERMR will be used as a more general, user-facing repository. Screenshots of the GUI are included in Appendix 1; the rest of this section describes how the ERMR is used through the API.

3.2.1. The ERMR API

From the user perspective, the ERMR exposes two RESTful APIs:

- A data store with nested containers and user object metadata implemented as key-value stores. The API is CDMI[3], a standard protocol for self-provisioning, administering and accessing cloud storage
- A triple store that allows assertion of properties and relationships and integrates with the table and object stores. The triple store exposes a SPARQL interface.

The ERMR issues notifications on significant events via the MQTT Protocol[4], which allows the system to instigate additional processing, curation, insertion of additional metadata, queue tasks into a workflow or any other action. The processing elements can be in any language, and Python is explicitly supported via a built-in listener.

Detailed API documentation is provided in Appendix 2.

3.2.2. Example Use Cases

In this section we briefly describe with two example use cases how the ERMR is used with the Process Compiler and workflow engine from the WP6 test-bed. The PERICLES Process Compiler allows to transform and combine RDF-based descriptions of preservation processes into executable workflows which can be executed by the workflow engine.

3.2.2.1. Process Compiler

The process compiler (Grant et al, 2015) can be used to compile aggregate processes from processes stored in the ERMR. This is done, for example, with the process compiler building an executable process (described using BPMN) which is made up from multiple processes.

The Process Compiler queries ERMR to retrieve process entities using the REST API. This API is used to send a SPARQL query. A typical query might look something like the following:

[3] http://www.snia.org/cdmi
[4] The Message Queue Telemetry Transport protocol. See https://mosquitto.org/ and http://mqtt.org/

```
prefix ecosystem:<http://www.pericles-project.eu/ns/ecosystem#>
select ?predicate ?object where {
      ecosystem:agpIngestAWSW ?predicate ?object .
}
```

3.2.2.2. Workflow Engine

The following example describes a process that uses the test-bed from WP6. When the user adds a new media file, the Workflow Engine extracts significant information from the file, and queries the ERMR to check that the necessary items are already included in the model, so that they can be referenced (e.g. a codec is already modelled, and then the RDF describing the codec can be referenced to the file). After sending the proposed change to MICE for the user to check that they're happy with the change, MICE returns the SPARQL describing the update, to the workflow engine. The workflow engine then makes an insert call to the ERMR to update it. So, the two types of call that the WE makes to the ERMR are to query to check that expected triples exist, and to send new triples to be added to the model.

3.3. Design & Implementation

The ERMR's architecture provides a compact REST interface for interoperating with the various heterogeneous tools developed as part of the project, as well as with other compatible third-party software tools. These tools, can be tracked and managed with the ERMR, which itself is extensible, allowing new tools to be added and deployed as trivially as is possible. A typical workflow might involve calling the metadata extraction workflows to extract metadata for indexing, while scheduling a format converter to convert the files to a format that can be processed. Notifications can be also forwarded to a rule engine, as illustrated in Chapter 7, in support of rule based change management, so that appropriate rules can execute the appropriate action, connecting to the DEM model and model driven preservation approach.

For production deployments, the ERMR offers authentication options for LDAP as well as local user authentication.

3.3.1. Implementation

The repository functions of the ERMR are created using a minimal schema on top of a decentralised, distributed hash table (DHT), providing a lookup service in which key value pairs are stored and any participating node can efficiently retrieve the value associated with a given key. The ERMR is implemented using Apache Cassandra distributed database management system, which is a widely used and hardened framework designed to handle large amounts of data across many commodity services, providing high availability with no single point of failure.

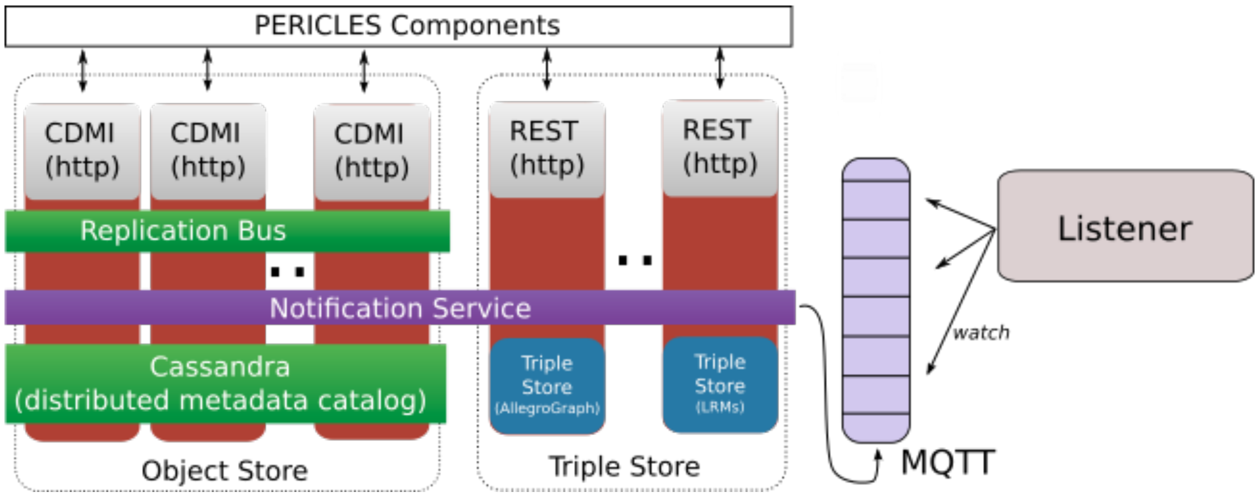


Figure 2: ERMR Technical Implementation

Internally, there is a directory of resources that describe the managed objects, and a block store for internally stored objects. Access to the ERMR objects is via the standard SNIA Cloud Data Management Interface (CDMI), which defines the functional interface used to create, retrieve, update, and delete data elements from the Cloud. The CDMI interface provides a hierarchical object store with metadata at the object and collection levels. This approach is illustrated in Figure 2.

The ERMR responds to internal events by firing a notification to a "listener" that executes scripts in Python or other scripting languages. These scripts access the repository or other actions, such as notifications, emails, logging, metadata updates, etc.

The implementation uses the Python scripting language under the Django web framework, which uses of the modularity and services already provided by the framework. This incorporates an access library that can be used by other programmes, such as scripting or command-line utilities.

Whenever possible, existing and well-supported standards are used to leverage the internet developer community's efforts to provide on-going support and reliability, as follows:

- Cassandra distributed database is an Apache top-level product
- Django and Nginx is a widely used and de-facto standard
- CDMI has been widely adopted by cloud service providers and is core to OpenStack. It requires no special libraries.
- AllegroGraph is a persistent graph database. It provides a triple store with a SPARQL interface for performing queries. It's loosely integrated with ERMR through its REST API.

3.3.1.1. Design of the Data Tables

ERMR reflects the data model of CDMI, i.e. a hierarchical organization of objects and containers, with every object and container also having a unique ID that allows direct access without traversing a directory tree. As such the two primary tables are the resource table and the collection table, describing, respectively, objects and their containers.

- *Collection Table*: The collection table is simple: containers, or collections, are a logical organising mechanism for objects akin to directories in a file system. Collections can be considered to be a label or part of the name of the object but they have some attributes such as timestamps and permissions that are shared across all the other objects in the collection.

- *Resource Table*: The resource table holds descriptions of the actual objects: the object's name, user annotations (metadata), and a reference to the place where it is stored, represented as a URL. Currently two URL schemes are supported: one for local files on the file system of a node in the cluster, and the other for files held in Cassandra as blocks. The URLs are of the form file://<node_address>/<path_to_file> and cassandra://<id_of_blob> respectively. It is straightforward for a developer to add other URL schemes and hence support other storage schemes, such as ftp, http, and other object stores.

3.3.1.2. Identifiers

The CDMI scheme requires that all objects and containers are also accessible directly using a CDMI ID string — a 256 bit unique identifier. The IdIndex table holds these IDs and refers to the primary records in the resource or container tables.

3.3.1.3. Users, Groups and Roles

Authentication and authorisation in the system is built around users, groups and roles. As they operate within the repository authenticated users have a role (or, equivalently, are members of a group), and it is that group membership that gives them authorisation to do certain things. When a user logs in, the system assigns them access rights based on their group membership. This is enforced using a system of access control lists (ACLs), with each ACL having a number of entries, each of which specifies whether particular rights are allowed or prohibited to particular groups of users. The user and group tables are used to specify (a) whether the ERMR system itself defines a user (i.e. local users), and (b) which groups the user belongs to. Users may also be authenticated using the LDAP system, in which case permissions are allocated based on an equivalent name, or are assigned to a generic group.

3.3.1.4. Ingest

Ingesting large collections takes a lot of time, and is prone to failures. The problem is that when failures occur, for whatever reason – disk full, network down, machine crashes – they occur on a large scale, in the middle of a lengthy process, and recovery is going to be painful. This reality implies that an ingest 'tool' is not enough. What is needed is an ingest workflow – one that remembers where it was, what it had done, and can restart, and preferably one that has multiple agents capable of working in parallel without treading on each other's toes.

The approach that we have taken is to use an RDBMS, in this case PostgreSQL to tabulate the objects to be ingested, along with their state, and so rather than having, for instance, a script that walks a file-system tree and injects into ERMR, we have scripts that walk a file-system tree and add any files that are found to a work queue in PostgreSQL. We then have a number of processes working in parallel reading records from the work=queue which are in a 'READY' state, moving them to 'IN-PROGRESS', with a timestamp, and then injecting them into ERMR, before moving them to a 'DONE' state.

3.3.1.5. Notification, triggers, and actions

One aim of the project is to develop automated policies and services that can be used across federated infrastructures. The science and arts datasets examined in the project rely heavily on an extensive set of metadata that allows for the discovery, analysis, and preservation of experimental results. Both space science and arts communities have developed common metadata standards and, as a primary objective for the ERMR is to allow these datasets to be managed using rules that react to change in repositories.

Every create-read-use-delete action on the ERMR generates an activity entry in the Cassandra database. The activities are published as topics to a messaging queue. Any client can subscribe to a relevant set of topics and execute workflows accordingly. The payload of the message can be used to identify the entity being impacted by the action.

The iRODS system has a vast range of triggers that can cause associated rules to be executed when certain events occur. Although very rich and general, the approach has not received widespread use. We determined in the design of the ERMR that we needed a mechanism whereby the rules could be written in any scripting language, that, for security reasons operated outside the core repository – so that it had to re-authorize. This essentially requires event queues distinguishable by originating node, type of event, affected object/container and timestamp. Unfortunately, the architecture of Cassandra means that it is not well suited to this, in particular, because it hashes the partition key and orders by the clustering key, it cannot efficiently handle wildcard matching, making Cassandra tables a poor candidate for generic event listeners. In practice we have implemented a hybrid system, whereby every notification gets written directly to a Cassandra table. A daemon is used to pull elements off the table in timestamp order and inject them into the notification system.

3.3.1.6. Management Policies

The PERICLES project assumes the existence of management policies that will control the properties of data collections as required for long-term digital curation. The research into policy requirements for any given collection involves specific investigation as to the purpose, the legal requirements, and the conditions for re-use of a collection (or sets of federated collections); it may also involve broader issues of consensus across different communities. There is already an international standard for managing Trusted Digital Repositories (ISO 16363) which, for the Space Science data use case, has informed recommendations and guidelines published as long-term digital preservation (LTDP) for observational data. The PERICLES project has taken this as a starting point for establishing the most appropriate methodology given use of the ERMR, to track and maintain an audit trail for the procedures supporting any management policy in state.

The PERICLES project has defined the set of procedures to ensure compliance as a policy pack. A policy pack is used to define when a policy should apply, whether it is met, and what should be done to bring it into compliance. The ERMR supports five flags signalling whether a case is 1) conforming; 2) non-conforming; 3) not-known; 4) not-applicable; and 5) exempted (a specific case of not-applicable). Formally, the system also supports graded levels of actions (“will”, “must”, “should”, "may", and "should not"), which are more or less precisely defined depending on the source of the policy.

Unfortunately, most "policy" documents are a mixture of policy and procedures, mainly in natural language text, with little or no structure. Taking such "policies" and turning them into actionable policy packs in is not a trivial process; and can requires rigorous interpretation. In practice, this means defining an explicit statement for each written policy, the obligations the policy imposes (both positive and negative), and what is permitted or not permitted for each circumstance. For any case, it must be possible to ascertain an exact policy "state" (not-applicable, not-conforming, not-known, confirming, or in progress), and the appropriate "action" to be taken. For any state change, the ERMR system will log what has happened, when and by whom, the policy version, and its justification. Furthermore, the ERMR also has implemented a mechanism for testing that all active policies are complied with before committing changes.

For example, the text of the ISO 16363 standard refers mainly to organisational and administrative requirements, with specific policy requirements that certain documents exist. The assessment criteria are based on the verifiable existence of such documents within the system. Additional criteria (mainly in Section 4 of ISO 16363) relate to the operation of the repository system itself: so it would be necessary for a person to certify that a repository maintains audit trails, applies access controls, etc. Operationally, an ISO 16363 "policy pack" would minimally verify the existence of documents and log that they have been changed and reviewed. The LTDP Observational Science guidelines suggest additions (in Sections 6 and 7) to the ISO 16363 standard, requiring an augmented policy pack.

Different policy requirements, such as those for medical records used in research, demonstrate great differences, which may be combined with the ISO 16363 policies. For example, the management of medical records may require that no personally identifiable information should be revealed, and that consent for the use of the data should have been obtained. In this case the procedures enforcing the policy might be: "No record will be ingested unless an accompanying consent record is present and is retained" and "No record will be shared unless the destination is known".

For the PERICLES project, we reviewed all of the elements of ISO 16363 standard in light of the LTDP Observational Data guidelines (and amended to fit the space science use case), as a first step in identifying the fundamental concepts required for implementing an appropriate data management infrastructure. This consisted of analysing 109 metrics and sub-metrics. The lessons learned in this exercise were used to evaluate and inform the architecture of the ERMR as a means of implementing preservation at scale in a distributed environment. An open question remains about how to interpret non-binary (and non-structured) "policies" such that they can be computer actionable; and to what extent can such policies (where they are computer actionable) be made generic and reapplied across different domains and communities. The experience we have obtained to date is that data management policies must be extensively tuned, but there are some core values that appear to be common to almost all use cases. Recent developments in game theory may open up a promising way to approach to the problem of policy management, to determine an equilibrium between generic and specific requirements, which will set the agenda for our future research.

The following example shows how a procedure that can be executed when allowed by the ISO 16363

Trusted Digital Repository Standard and the OAIS Standard (which provides the context for ISO 16363). Such requirements permit “deletion” operations when they are already identified as part of an approved strategy (i.e. there should be no ad hoc deletions). The execution of the script below will be informed by a broader administrative context: for example, if an organization has a policy for “group deletion” after six months of inactivity, the script will be invoked, referencing appropriate authentication credentials (where appropriate). The larger “organisational” or “strategic” contexts come from an analysis of the use cases, which is separate from this task.

The following is a sample script written in python used “to delete groups of users”. To enable the script to work, an administrator would upload the script in the /scripts directory, adding metadata to define the hook (‘topic’ : ‘/delete/group/#’):

```
import sys
import json
from indigo import DataObject

# Name of the script that got executed
script_name = sys.argv[0]
# Payload of the message
payload = json.loads(sys.argv[2])

# name of the deleted group:
group_name = payload["post"]["name"]

for data_obj in DataObject.find(user=group_name):
      data_obj.update(user="ProjectManager")
```

3.3.1.7. The Listener

The ERMR issues notifications whenever a significant event occurs, which can be caught by a “listener” and an action script invoked.

The listener is a daemon process which is hooked to the MQTT broker. It subscribes to all topics generated by the notification system. Each time a new event is sent to the queue the listener checks in the scripts it is managing and executes the one whose topic is matching. It manages a set of scripts stored in a special collection of the repository. Each script is linked individually to a specific action and executed by listener when the condition is met. Every script is executed in an isolated Docker image, it has only access to a limited environment. By reading the payload of the message it can obtain information on the objects to consider and act accordingly. It may for instance use an external web service call to extract additional metadata. The scripts are executed in a Docker image to provide isolation, for security. The scripts are currently written in Python in the PERICLES implementation, but any other languages can be added. Figure 3 illustrates the procedural steps of the listener.

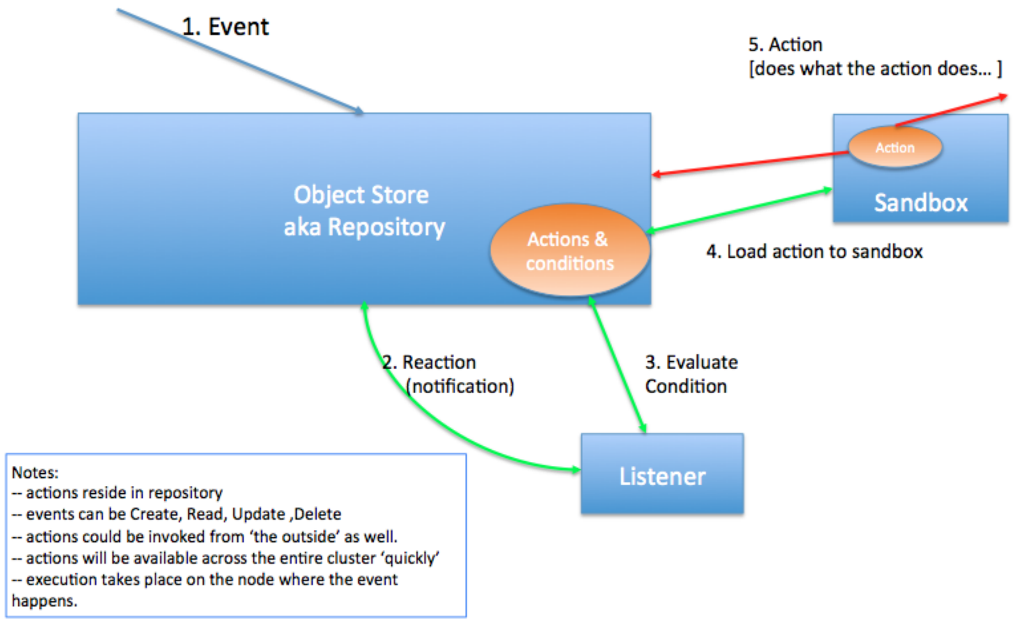

1. Event
5. Action
[does what the action does…]
Action
Sandbox
Object Store
aka Repository
Actions &
conditions
4. Load action to sandbox
2. Reaction
(notification)
3. Evaluate
Condition
Listener
Notes:
-- actions reside in repository
-- events can be Create, Read, Update ,Delete
-- actions could be invoked from 'the outside' as well.
-- actions will be available across the entire cluster 'quickly'
-- execution takes place on the node where the event
happens.

Figure 3: The Listener Architecture Diagram

The scripts managed by the Listener are stored in a special collection in the object store. Specific metadata can be added to the script to define the rule execution condition for this script. Details are included in Appendix 3.

3.4. Obtaining the ERMR

ERMR is an Open Source (Apache v2) tool and will be published on GitHub by the end of January 2017 under the URL https://github.com/pericles-project/ERMR.

3.5. Conclusion

The ERMR is a software for storing and querying models and digital objects and constitutes a central component of the PERICLES WP6 test-bed for registering models and metadata. Policies which operate on the data store help to manage the stored data and those policies can invoke external programmes. The notification system issues events onto a message queue, so connection with third party components and workflows is possible. The whole tool is designed to be flexible and scalable, it also provides an authorisation and authentication mechanism for secure access. It uses standard protocols for storing and querying the data, which eases the integration with other systems.

It is planned to continue the development of ERMR after the project end. There have been already contacts with academic and research organisations. The main strand for further development is a central metadata catalogue with cluster functionality. There have been contacts with the Science and Technology Facilities Council, the Culham Centre for Fusion Technology (CCFE) and Edinburgh Parallel Computing Centre (EPCC). As a result of the project dissemination activities, the technology is already adopted at the University of Maryland Digital Curation Innovation Center.

4. Policy Editor

4.1. Background & Motivation

Policies are important to preservation. Preservation policies typically define the desired state of part of the preservation ecosystem, for example, what should be preserved, what formats should be used to store data, how many copies should be stored, etc. They do not typically define how things should be done (this is defined in a process) but in practice, this boundary is often blurred; it could be that the policy dictates that a certain process be used under certain conditions, and it can be that machine-readable policies include processes that, if followed, can guarantee that a policy is enforced. The policy editor described here uses a model of policies that is described below in section 4.2.1.

Creation of sound policies is considered to be a difficult problem. As part of T5.2.1 we have created a Policy Editor. The primary goal of the Policy Editor (PE) is to enable and assist in the manual creation of a consistent set of policies (see D3.2 Linked Resource Model, page 31). We set out to create a policy editor that allows:

- creating natural language policies;
- creating policies that adhere to a particular policy model;
- using a structure of policies that can contain sub-policies that can optionally contain (executable) process data;
- creating/maintaining a set of policies using a predefined set of policy templates;
- creating policies that use domain specific terms and concepts from the digital ecosystem wherein the policies apply;
- ensuring that the set of policies is fully specified to the level of detail that is required for a particular application;
- exporting to both (printable) human-friendly and computer-friendly formats.

The main secondary objective is to allow policy validation and process execution. This optional capability is available when the used policy model supports executable/validatable policies/ processes and a policy execution/validation component is connected to the PE.

4.2. Use and Functionality

A typical use scenario for the PE is depicted in Figure 4. Here a policy creator defines a set of policies from scratch. In this case the policy creator is an individual that belongs to an organisation or team that is responsible for selecting, defining and tailoring, but not necessarily approving, the set of policies applicable for his domain of interest. As an example, in projects that require a formal Data Management Plan, which is a document that describes how a project will manage, describe, preserve and use generated data in the project, the Policy Editor can be used by a data curator (who then fulfils the role as policy creator) to define the contents of the Data Management Plan. Typically, policies are created at the start of an activity by a manager of these activities. Instead of starting from scratch, it is also possible to load an existing set of policies (also created with the Policy Editor) and build on that.

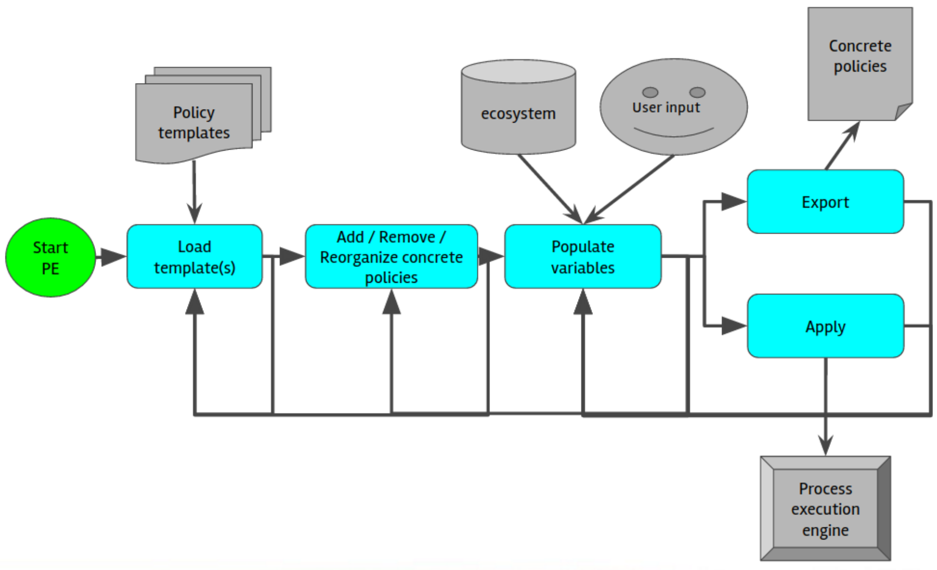


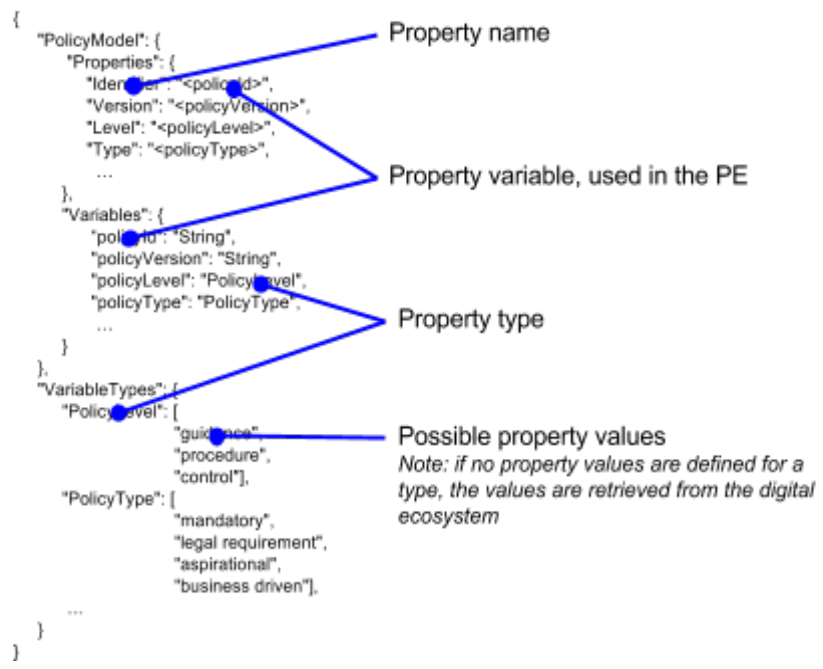
Figure 4: A typical use scenario for the policy editor

The PE can be used as follows:

- The policy creator selects one or more existing policy template files that contain policy templates; these can be thought of as incomplete policies that are to be completed by the policy creator. Policy template files are created by experts and can be, for example, domain-specific, organisation-specific, provided by the authorities, etc.
- From these policy template files, the policy creator makes an initial selection of those top-level policy templates that are applicable to a particular scenario, system, infrastructure or organizational structure. The policy creator can go back and add additional (top-level) policy templates at any time.
- In the next phase the policy creator updates the policy structure of the selected policy templates by adding or removing lower-level policy templates (selected from the template files). This allows the policy creator to fine-tune the policy structure to the specific needs of the application.
- Depending on the complexity of the policy model used, the policy creator has to create concrete policies from the templates by filling in the missing parts. As an example, a policy model dictating that each policy must have a version, will result in the policy creator having to fill in a version for each imported policy template. To improve consistency among policies, variables are introduced that can be shared between policies. Changing the value of a variable in one policy will automatically update all other policies where that variable is used. The variables values are either free text, typed by the policy creator, or selected from a limited list defined either in the policy model, the policy template or the digital ecosystem.
- After creating the set of relevant policies, the policy creator can export the policies to a PDF/text format or a computer-friendly JSON version. If configured with a process execution engine, the PE can also trigger a policy validation and visualise the results.

4.2.1. Templates & Policy Model

For the purposes of the policy editor, policies are modelled as shown in Figure 5[5]. Policies ("instances" of this policy model) can be created from policy templates as described above.



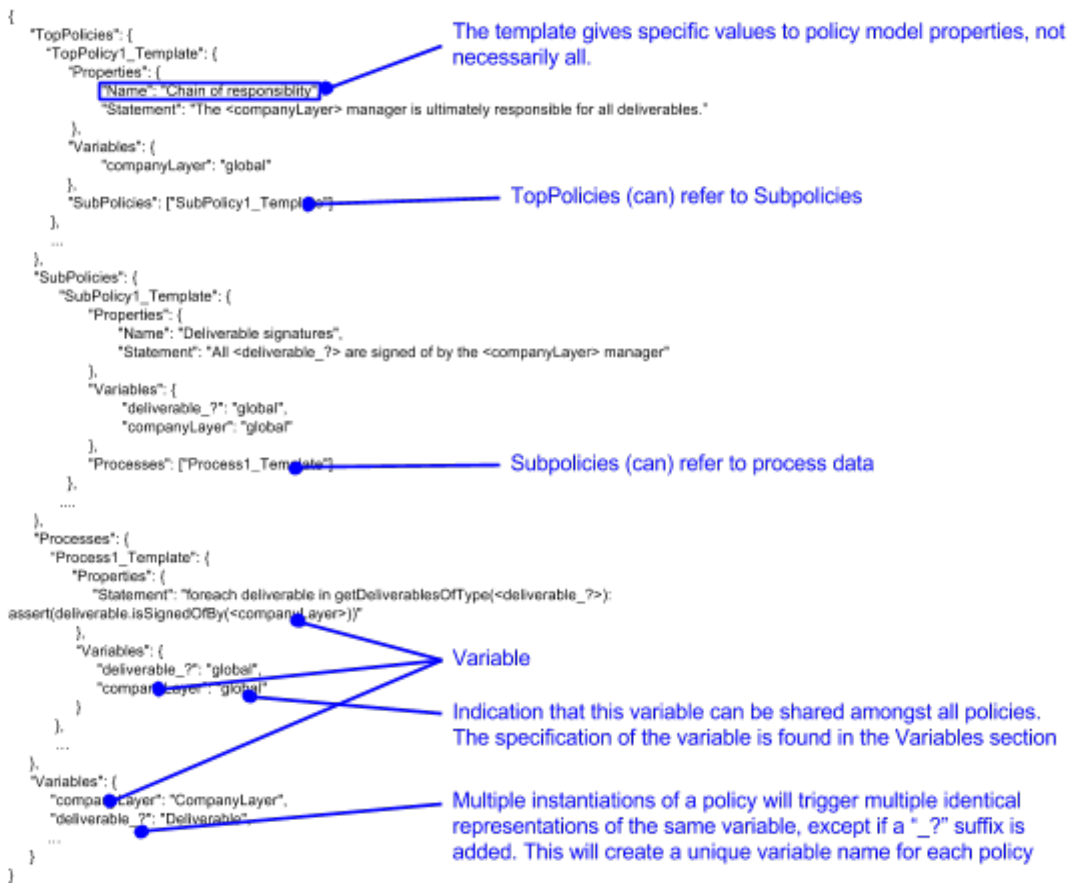
```
{
  "PolicyModel": {
     "Properties": {
        "Identifier": "<policyId>",
        "Version": "<policyVersion>",
        "Level": "<policyLevel>",
        "Type": "<policyType>",
          ...
     },
     "Variables": {
         "policyId": "String",
         "policyVersion": "String",
         "policyLevel": "PolicyLevel",
         "policyType": "PolicyType",
          ...
     }
  },
  "VariableTypes": {
     "PolicyLevel": [
                     "guidance",
                     "procedure",
                     "control"],
     "PolicyType": [
                     "mandatory",
                     "legal requirement",
                     "aspirational",
                     "business driven"],
      ...
  }
}
```

Property name
Property variable, used in the PE
Property type
Possible property values
Note: if no property values are defined for a type, the values are retrieved from the digital ecosystem

Figure 5: Policy Model

Policy templates are described as shown in the example in Figure 6. Templates contain properties with content (which may contain variables), specification of variables and an optional list of lower-level policies. Variables are typed and these types can have a predefined set of values from which the policy creator can choose. These predefined values can originate from either the template, the model or, if an ecosystem adapter is used (as described below), from the digital ecosystem. Further information about the contents of policy files can be found in Appendix 5.

[5] The policy here is described using JSON, but it could equally well be described in RDF using the LRM.



```
{
  "TopPolicies": {
    "TopPolicy1_Template": {
      "Properties": {
        "Name": "Chain of responsiblity"
        "Statement": "The <companyLayer> manager is ultimately responsible for all deliverables."
      },
      "Variables": {
        "companyLayer": "global"
      },
      "SubPolicies": ["SubPolicy1_Template"]
    },
    ....
  },
  "SubPolicies": {
    "SubPolicy1_Template": {
      "Properties": {
        "Name": "Deliverable signatures",
        "Statement": "All <deliverable_?> are signed of by the <companyLayer> manager"
      },
      "Variables": {
        "deliverable_?": "global",
        "companyLayer": "global"
      },
      "Processes": ["Process1_Template"]
    },
    ....
  },
  "Processes": {
    "Process1_Template": {
      "Properties": {
        "Statement": "foreach deliverable in getDeliverablesOfType(<deliverable_?>):
assert(deliverable.isSignedOfBy(<companyLayer>))"
      },
      "Variables": {
        "deliverable_?": "global",
        "companyLayer": "global"
      }
    },
    ...
  },
  "Variables": {
    "companyLayer": "CompanyLayer",
    "deliverable_?": "Deliverable",
    ...
  }
}
```

The template gives specific values to policy model properties, not necessarily all.
TopPolicies (can) refer to Subpolicies
Subpolicies (can) refer to process data
Variable
Indication that this variable can be shared amongst all policies. The specification of the variable is found in the Variables section
Multiple instantiations of a policy will trigger multiple identical representations of the same variable, except if a "_?" suffix is added. This will create a unique variable name for each policy

Figure 6: Example template file

4.2.2. Integration with the Ecosystem and Process Execution

As well as operating as a stand-alone tool, the policy editor can be made aware of the wider preservation ecosystem in which it is being used, by connecting to an external data provider. The interface to the data provider is defined in such a way that there is no 'vendor lock-in' to any particular digital ecosystem infrastructure. This is achieved by using adapter components to form an intermediate layer between the PE and the ecosystem. As an example, an ERMR-specific adapter would translate the PE query `getValues(aType)` to ERMR-specific queries to retrieve all values/entities that are registered in the ERMR that would correspond to the given type. In a more lightweight setting, the digital ecosystem role could be fulfilled by a simple text file containing the possible values for the supported types. In this case the role of the adapter would be to read from this file and return the requested values. As it is not unlikely that policy templates and the digital ecosystem infrastructures originate from different sources and that therefore concepts have different names, the adapters can be configured with a translation mapping between types coming from the PE and types as they are defined in the digital ecosystem.

Although the primary goal of the PE is to allow a policy creator to define policies that adhere to a policy model, the PE optionally allows the policy creator to use and fully define the lowest level policies using "executable" process data. This can be done by having a Policy Statement property that is of *Format* **formal** and of *Language* **SPIN** (for example) and that contains a concrete SPIN rule.

The PE can then be connected to a process execution engine.

When using SPIN rules as the Policy Statement, the PEE role is fulfilled by a respective rule engine. Again, similarly to allowing the PE to connect to a digital ecosystem infrastructure, the PE tries to be generic enough to support multiple types of PEEs. To this end, to be able to use this optional feature, a PEE-specific adapter must be developed that is able to communicate with both the PE and the PEE. When executing a Policy that contains process data, this policy calls the `execute(Policy_ID,Process_data)` function of the adapter that is responsible for translation into a PEE-specific query. Optionally the result is returned to the PE, again through the adapter, so that this result can be visualised in the PE. Note that the PE makes no assumptions whatsoever with regards to the content of the process data. This process data can be queries, a config file, an executable script, or something else. The only offering by the PE in this regard is that in the template the content of this process data can contain variables and that the same rules apply for process data as for other policy data with regards to variable propagation etc.

4.3. Design & Implementation

4.3.1. Architecture

The Policy Editor follows a client-server architecture. The core application is deployed on an application server, and the (GWT-based) GUI is offered through a standard web browser. The PE relies on other (albeit optional) components to offer some of its functionalities. This is illustrated in Figure 7.

More specifically, it offers interfaces to integrate with:

- a policy store (for persistence of policies, processes):
- the Model Repository of the ERMR could be used for storing policies,
- an ecosystem infrastructure (to refer to digital entities from a repository),
- within PERICLES, the ERMR would fulfil this function,
- a Process Execution Engine (to run processes and validate policies).
- within PERICLES, the Process Compiler working together with the Workflow Engine could perform this role.

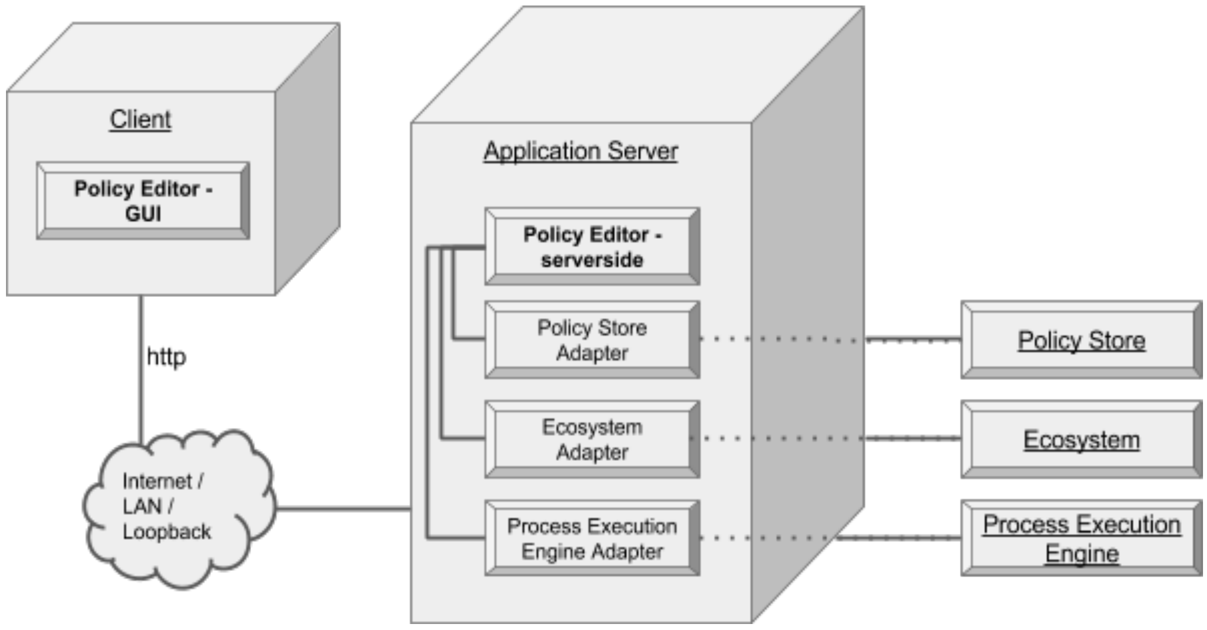


Figure 7: Policy Editor architecture

Where there are existing components, the Policy Editor can be integrated by using adapters. Moreover, thanks to its modular nature, the essential editing functionality can be guaranteed also without the availability of an ecosystem or a Process Execution Engine. A policy store, however, is always required to persist policy and process information. It is treated as a distinct module so as to easily adapt to different integration scenarios. Currently, the policy store is implemented as a JSON export and import component.

The policy editor can be configured in multiple ways, depending on how it must integrate with other components. Various different configurations are described in Appendix 4.

4.4. Obtaining the Policy Editor

The (proprietary) Policy Editor tool is, at the time of writing this present deliverable, not available online. Further information can be obtained by contacting david.deweerdt@spaceapplications.com or rani.pinchuk@spaceapplications.com.

4.5. Conclusion

The Policy Editor assists on creating policies via templates, which means that the policies are predefined and the PE helps in compiling them and fill-in concrete values. The policy level and integration level of the Policy Editor is flexible. The PE can be used as a stand-alone tool or integrated with other tools, so the templates can be kept within other systems or be basic text files. The same applies to the output, it can be plain text files or integrated into other systems, e.g. sending them directly to a policy execution engine. This is possible by providing custom adapters.

5. Technical Appraisal Tool

Sections 5 and 8 constitute the final report on the work carried out on task T5.4 Support functionality for appraisal processes. Section 5 focuses on T5.4.2 Appraisal tools, which describes the implementation of our proposed methods in a software application called the Technical Appraisal Tool. Section 8 describes the work on T5.4.1 Modelling of Appraisal Processes, which defines our overall approach and methodology.

Essentially, this report is an update on the material presented D5.2. There we introduced our methodology, outlined a technical approach and conducted some initial investigations and experiments. The primary focus since then has been on implementing and evaluating the methods and on development of a practical appraisal tool.

5.1. Background and Motivation

In D5.2, we partitioned appraisal criteria into two distinct categories:

- **Technical appraisal** – decisions based on the (on-going) feasibility of preserving the digital objects. This involves determining whether digital objects can be maintained in a reusable form and in particular takes into account obsolescence of software, formats and policies.
- **Content-based (or intellectual) appraisal** – acquisition and retention decisions or assignment of value based on the content of the digital objects themselves.

Roughly speaking, technical appraisal addresses the question “can we preserve?”. Technical appraisal can be extended to cover any aspect of change in digital ecosystem entities and any type of entity, including user communities, policies and processes. Content-based appraisal addresses the question “what to preserve?”. Our main focus in the final development phase of the project has been on technical appraisal.

The main problem being addressed by the technical appraisal tool is to assess the long term sustainability of complex digital objects, which can include science experiments and digital video and software based artworks.

For the purpose of analysing risks to digital ecosystems, we define two types of risks:

- A **primary risk** is a potential change to an entity arising through a stimulus that is external to the ecosystem.
- A **secondary (or higher-order) risk** is a risk to an entity as a result of a potential change to another entity on which it has a dependency.

We use data-driven methods to first determine analytically the primary external risks to the digital ecosystem. We then apply probabilistic methods and the underlying digital ecosystem models to determine the secondary risks, as well as their impact and proximity.

5.2. Use and Functionality

The technical appraisal tool is aimed at conservators of time-based media collections and science data managers who are responsible for maintaining the long-term reusability of complex digital objects. We aimed as far as possible to keep with design of the tool as consistent across the two case studies in order to demonstrate that it could potentially be applied across multiple application domains. The main requirement to extend to other domains is to provide the required ontologies and sources for environment data harvesting. The model instances for each domain are stored in separate folders. The application domain can be selected from the start page of the tool (i.e. digital art or science). See the narratives in section 8.2.

The ecosystem model needs to be provided for each application. PERICLES provides the EcoBuilder tool (see section 6) for constructing ecosystem models. This in turn will import external ontologies or custom domain ontologies built with existing tools such as Protégé. The appraisal tool assumes that the ontologies have been built and are stored in the ERMR.

The appraisal tool was conceived as a web-based application. This provides the most flexibility in running the tool on multiple platforms. For real world deployments in archives and repositories working with large volumes of content, the appraisal tool processing components would be likely to require higher performance servers than an individual PC such as cloud infrastructure.

Further details about our approach to technical appraisal and more detailed description of the functionality of the various components is provided in section 8.

5.3. Design and Implementation

5.3.1. Tool architecture

The technical architecture of the tool is shown in Figure 8. It is based on a web service framework, with individual tools implemented as web services.

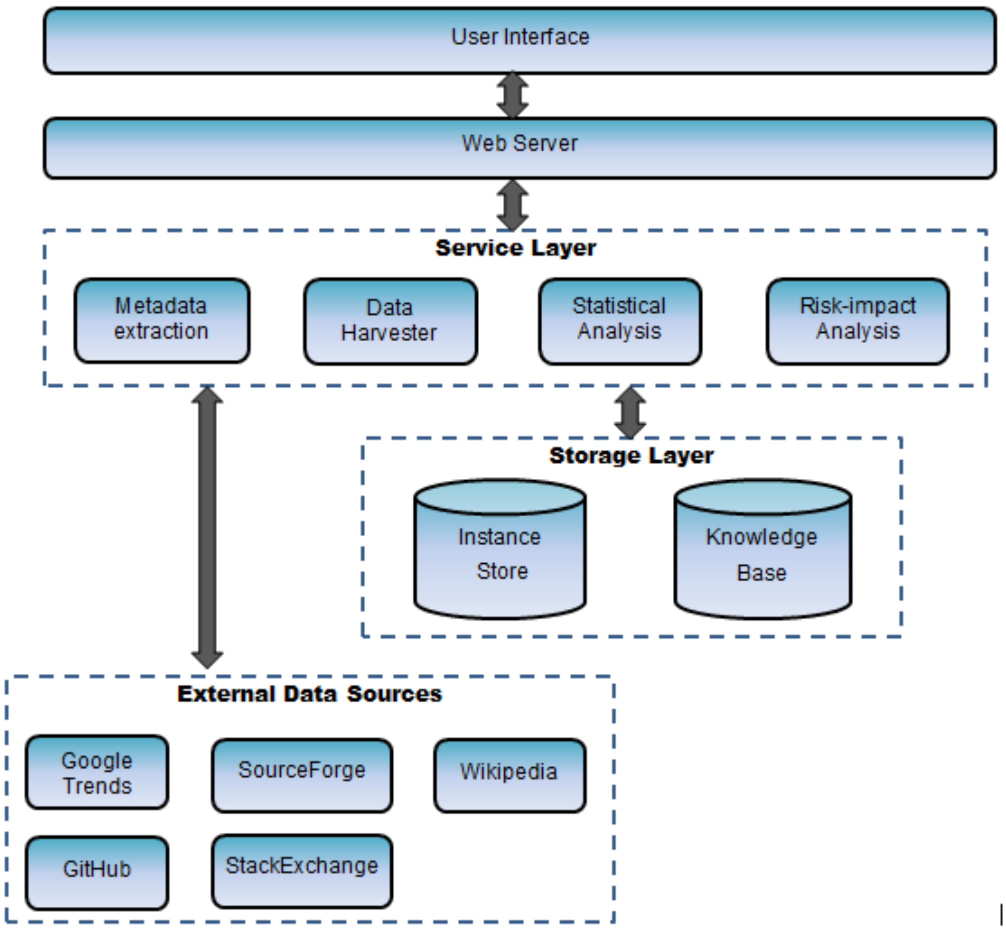


Figure 8: High-level architecture of the technical appraisal tool

The architecture is formed of a number of interacting layers as follows:

- The *user interface* layer contains the components for user interaction with the tool including registration of digital objects, risk-impact analysis and general content management functions.
- The *web server* deals with user requests and serves up pages to the user.
- The *service layer* comprises the main technical functions of the tool. We have a adopted this approach as the various components needed to be written in a number of different languages namely Java, Python and R, to make use of specialised built in libraries.
- The *storage layer* contains both the domain and ecosystem model ontologies in the form of the *knowledge base*, and the ecosystem model instances in the *instance store.* In practice these are stored in the same RDF store.
- *External data sources* are external applications that typically web service interfaces. Custom adaptors for each application are provided in the service layer.

The service layer comprises four main components:

- *The Data Harvester* gathers data from predetermined external data sources for analysis and writes the raw results into the model repository. Harvesting can be configured to run periodically at a frequency set by the user. Currently there exist adaptors for Google Trends, Wikipedia, GitHub, SourceForge and Wikipedia.

- *Statistical Analysis* contains the libraries for modelling the raw data and computing risks and proximities.
- *Risk-impact Analysis* is a component that builds Bayesian models from the ontologies and primary risks to compute the risk impact and proximity on the complex digital object under consideration.
- *Metadata extraction* is a component to analyse digital components and extract relevant metadata fields that can be used to populate an ecosystem model instance.

External data sources are linked to adaptors in the service layer. In general, the more independent sources of data that can be harvested, the more reliable the predictions. We were, however, restricted to data sources freely available on the Internet.

The narratives and workflow for technical appraisal implemented by the tool are described in sections 8.2 and 8.4 respectively.

5.3.2. User Interface: Functional Design

The functional design of the user interface for the technical appraisal tool is described in Figure 9. Each block represents a particular view of the interface with an associated set of functionalities. The arrows represent allowed transitions between the functional areas.

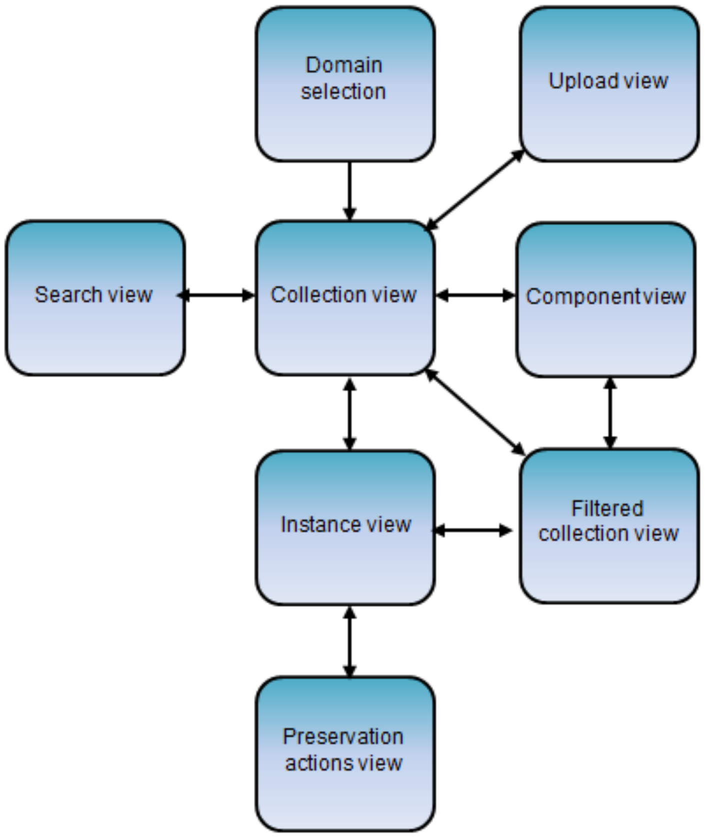


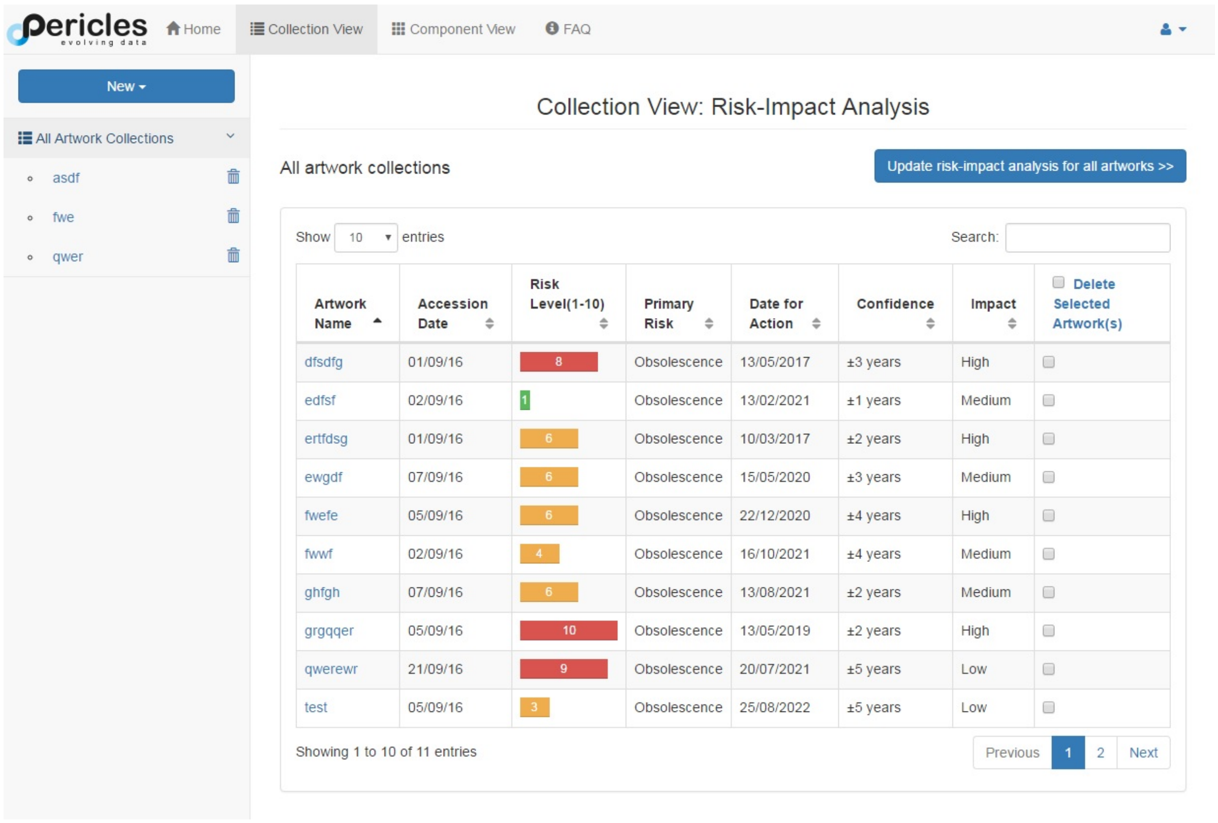
Figure 9: Functional view of the technical appraisal tool user interface

The purposes of the various functional components are as follows:

- D*omain selection* is a single page that enables the tool to be switched between different application domains that use different background ontologies. In the case of the PERICLES demonstrator, this enables us to switch between the digital art and space science domains.
- *Collection view* is the main entry point which displays risk information about ecosystem instances (e.g. digital video artworks or individual instances of science experiments). The collection view also features grouping of ecosystem instances into folders or sub-collections.
- The *component view* provides a view of all the entity types within all collections or specified a specified collection. A component is defined as an ecosystem entity such as a software application (e.g. Windows Media Player). It can also potentially include hardware, user communities or policies. Information is provided on the frequency of occurrence and associated risks.
- The *filtered collection view* provides a view of all ecosystem instances containing a specific component selected in the component view (e.g. all digital video artworks using the H.264 codec).
- The *instance view* provides risk information on all entities or components within a given ecosystem instance (e.g. a digital video artwork). For a given component, the user can select to perform preservation actions that aim to replace that component in order to mitigate the risk. Executing those actions involves the PERICLES test-bed described in Deliverable D6.6 Final Version of Test Bed Implementation.
- The *preservation action view* describes the recoverability options for addressing a risk in a particular component within a given ecosystem instance. It is only possible to perform preservation actions on individual ecosystem instances and not in bulk. Hence the preservation action view is only accessible via the instance view.
- The *upload view* enables a new ecosystem instance to be added to a specified collection, including both manual and automated metadata creation.
- The *search view* enables ecosystem instances to be retrieved by keyword search.

5.3.3. User Interface: Visual Design

Figure 10 shows a view of the user interface for the Collection View.



Collection View: Risk-Impact Analysis

All artwork collections

Update risk-impact analysis for all artworks >>

Show 10 entries Search:

Artwork Name	Accession Date	Risk Level(1-10)	Primary Risk	Date for Action	Confidence	Impact	Delete Selected Artwork(s)
dfsdfg	01/09/16	8	Obsolescence	13/05/2017	±3 years	High	☐
edfsf	02/09/16	1	Obsolescence	13/02/2021	±1 years	Medium	☐
ertfdsg	01/09/16	6	Obsolescence	10/03/2017	±2 years	High	☐
ewgdf	07/09/16	6	Obsolescence	15/05/2020	±3 years	Medium	☐
fwefe	05/09/16	6	Obsolescence	22/12/2020	±4 years	High	☐
fwwf	02/09/16	4	Obsolescence	16/10/2021	±4 years	Medium	☐
ghfgh	07/09/16	6	Obsolescence	13/08/2021	±2 years	Medium	☐
grgqqer	05/09/16	10	Obsolescence	13/05/2019	±2 years	High	☐
qwerewr	21/09/16	9	Obsolescence	20/07/2021	±5 years	Low	☐
test	05/09/16	3	Obsolescence	25/08/2022	±5 years	Low	☐

Showing 1 to 10 of 11 entries

Previous 1 2 Next

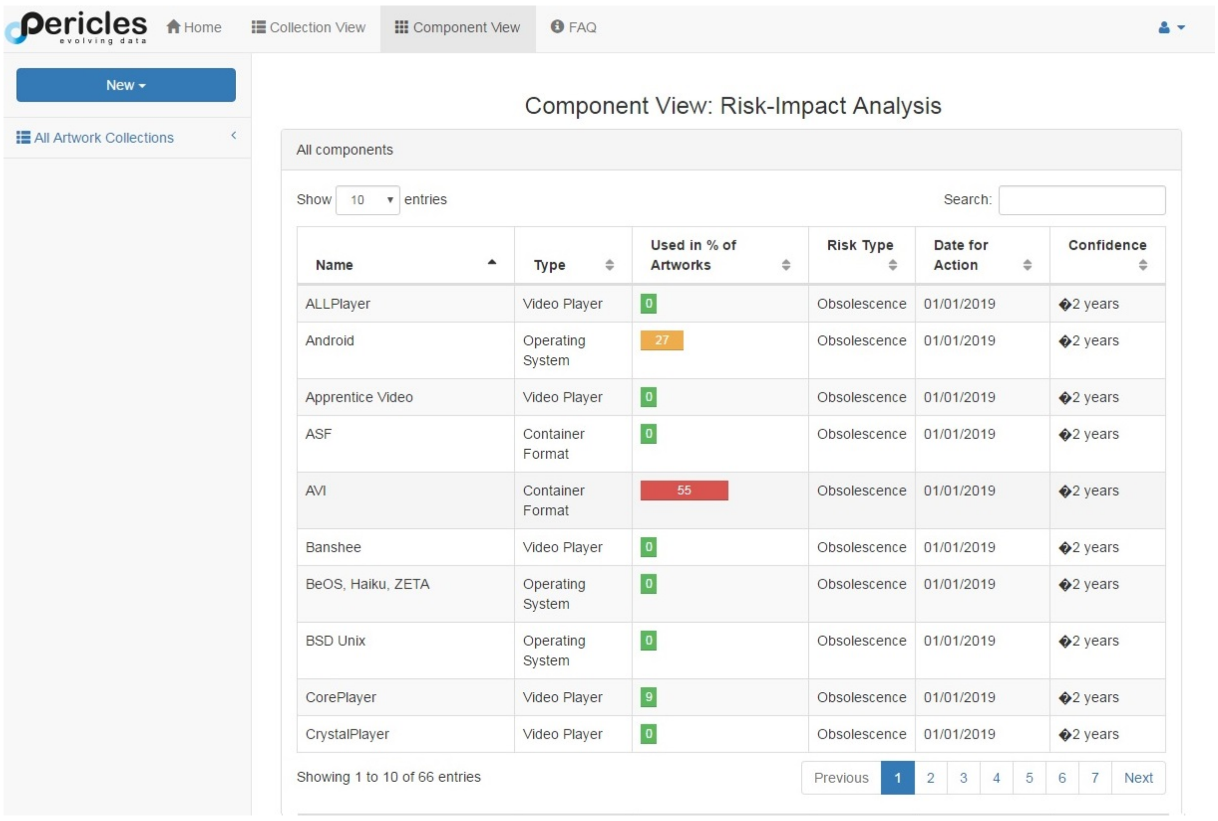
Figure 10 The Collection View page in the technical appraisal tool

The main entry pages, namely collection view and component view are accessible via the top navigation bar. The side pane enables navigation by folder. New folders and ecosystem instances can be added via the new button in the left side pane. The view on the objects (i.e. digital artworks or science experiment instances) themselves contains information about the main preservation risks. Sorting by column is possible, so the user can rank the items according to different criteria.

Since a conservator or data manager is likely to have a large volume of content to manage, we have created a simpler Risk Level measure, which provides a single risk-impact measure to highlight items that require urgent attention.

Since large archive collections may comprise many thousands of items, it may be difficult to gain an overall impression of the state of a collection from viewing items in tables, where the is a limitation of 10-20 items that can be viewed on a single screen. Therefore, we also provide additional graphical views of collections that can be used to highlight the main risk factors.

Figure 11 shows a screenshot of the Component View, which provides an aggregated view of all entities in a collection or sub-collection together with associated risk information.



Component View: Risk-Impact Analysis

All components

Show 10 entries Search:

Name	Type	Used in % of Artworks	Risk Type	Date for Action	Confidence
ALLPlayer	Video Player	0	Obsolescence	01/01/2019	�2 years
Android	Operating System	27	Obsolescence	01/01/2019	�2 years
Apprentice Video	Video Player	0	Obsolescence	01/01/2019	�2 years
ASF	Container Format	0	Obsolescence	01/01/2019	�2 years
AVI	Container Format	55	Obsolescence	01/01/2019	�2 years
Banshee	Video Player	0	Obsolescence	01/01/2019	�2 years
BeOS, Haiku, ZETA	Operating System	0	Obsolescence	01/01/2019	�2 years
BSD Unix	Operating System	0	Obsolescence	01/01/2019	�2 years
CorePlayer	Video Player	9	Obsolescence	01/01/2019	�2 years
CrystalPlayer	Video Player	0	Obsolescence	01/01/2019	�2 years

Showing 1 to 10 of 66 entries

Previous 1 2 3 4 5 6 7 Next

Figure 11: Component view in the technical appraisal tool

5.3.4. User Interface Implementation

The user interface is a web-based user interface which utilises HTML5, JavaScript and CSS3 technologies. SB Admin 2, a Bootstrap 3 based dashboard template is utilised to achieve this goal. Besides this, many JavaScript libraries are used to simplify the client-side scripting of HTML. For example, jQuery is used to handle AJAX calls to exchange data with the backend and update elements of the web pages. DataTables is used to provide advanced interaction controls to HTML tables. Other JavaScript libraries (MetisMenu, Flot, Bootbox, jQuery Validation, File Input, Vis.js, Pace, etc.) are also used to provide extended functionality such as generating dynamic menus, producing pie charts, validating inputs, enabling file upload, drawing graphs and animating page loading progress. In addition, CSS files are used to configure the visual appearance the HTML elements.

5.4. Obtaining the Technical Appraisal Tool

The Technical Appraisal Tool is an Open Source (Apache v2) tool and will be published on GitHub by the end of January 2017 under the URL https://github.com/pericles-project/TechnicalAppraisalTool.

5.5. Conclusion

The appraisal tool is a web-based tool for appraisal of technical risks. It implements the theoretical approach described in section 8. The tool uses PERICLES ecosystem models as input, and makes use of data harvested from external sources such as Google, software repositories and Wikipedia. The tool features a user interface that presents the complex risk-impact-proximity information in a number of different views to assist the user in both determining risks in their collections as well as analysing risks to specific digital objects such as digital artworks or science experiments.

6. EcoBuilder Tool

6.1. Background and Motivation

A key idea behind model-driven preservation is that the software components and associated processes work with models of the digital ecosystem of the objects to be preserved. It is therefore important that ecosystem models can be built easily. The EcoBuilder tool is designed to support the building of ecosystem models, and related policy and QA models. An important target user-base for this tool is scenario experts, who are not necessarily ontology experts or developers and the tool enables them to model their digital ecosystems and scenarios.

The EcoBuilder supports the creation of models and their submission to the ERMR triple store via a provided send function. Two interfaces are provided by the EcoBuilder for model creation: a Java API for developers, and a Graphical User Interface (GUI) for scenario experts. The resulting model contains the entities and relations belonging to a designated scenario and is stored in the ontology formats Turtle and OWL/XML.

The EcoBuilder contains templates for the well-defined creation of all DEM entities and relations. Templating reduces the complexity of modelling which facilitates a simple and well-defined way to create models, but it also restricts the broad possibilities of ontologies. For most scenarios the level of detail for modelling provided by the EcoBuilder is sufficient. Specific demands can be addressed by either making the required changes directly on the outputted model, or if the EcoBuilder's Java API is used then it is recommended to use the Java Jena API[6] which is imported by the EcoBuilder to deal with ontology concepts.

The provided templates encompass all DEM sub-ontologies, and the underlying parts of the LRM which should be directly configurable by the user. The Digital Video Artwork (DVA) domain ontology (see D2.3.2) is also integrated into the EcoBuilder for demonstrating the tools extendibility and for creating exemplary DVA scenarios.

6.2. Use and Functionality

The intended users of EcoBuilder are people who want to apply the DEM for a specific scenario. The intended user groups are persons which want to model their DE, in general persons who manage complex heterogeneous systems, e.g. archive managers, repository managers, system architects and also software developers (for using the programming interface).

The EcoBuilder's GUI is designed to make the modelling of DEMs user friendly by providing a graphical user interface and programming interface, which provides an interface for the connection to other PERICLES tools that deal with models, e.g. the ERMR which provides a triple store for the models.

The user is not bound by the EcoBuilder to follow a designated modelling strategy, but we

[6] https://jena.apache.org/

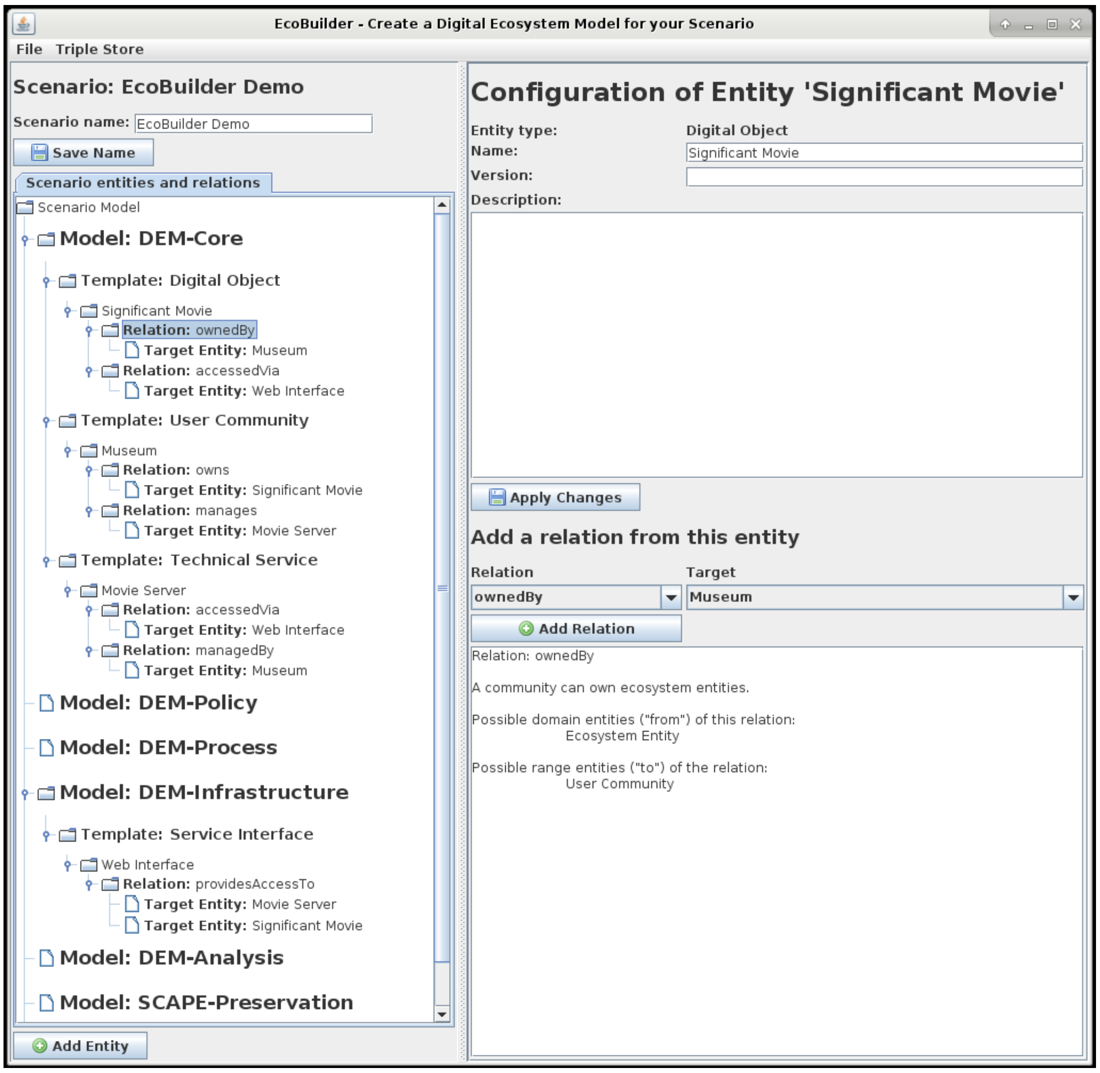
recommend a structured modelling strategy as the introduced policy based modelling in appendix 10, or a digital object centric modelling. This helps to keep an overview of complex scenarios and entity linkage.

Here in PERICLES the tool supports the bootstrapping process of feeding the knowledge base from the functional architecture with a model about a certain DE.

6.3. Design and Implementation

6.3.1. Using the GUI

A screenshot from the graphical interface of the EcoBuilder is shown in Figure 12. On the left side is the scenario model containing all entities and relations belonging to the user's scenario represented in a tree. The tree shows the created scenario entity instances ordered by their entity template types and sub-models to which the template entities belong.


EcoBuilder - Create a Digital Ecosystem Model for your Scenario
File
Triple Store
Save Name
Scenario entities and relations
Apply Changes
Add Relation
Add Entity

Scenario: EcoBuilder Demo

Scenario name: EcoBuilder Demo

```
Scenario Model
  Model: DEM-Core
    Template: Digital Object
      Significant Movie
        Relation: ownedBy
          Target Entity: Museum
        Relation: accessedVia
          Target Entity: Web Interface
    Template: User Community
      Museum
        Relation: owns
          Target Entity: Significant Movie
        Relation: manages
          Target Entity: Movie Server
    Template: Technical Service
      Movie Server
        Relation: accessedVia
          Target Entity: Web Interface
        Relation: managedBy
          Target Entity: Museum
  Model: DEM-Policy
  Model: DEM-Process
  Model: DEM-Infrastructure
    Template: Service Interface
      Web Interface
        Relation: providesAccessTo
          Target Entity: Movie Server
          Target Entity: Significant Movie
  Model: DEM-Analysis
  Model: SCAPE-Preservation
```

Configuration of Entity 'Significant Movie'

Entity type:	Digital Object
Name:	Significant Movie
Version:	
Description:	

Add a relation from this entity

Relation	Target
ownedBy	Museum

Relation: ownedBy

A community can own ecosystem entities.

Possible domain entities ("from") of this relation:
Ecosystem Entity

Possible range entities ("to") of the relation:
User Community

Figure 12: Screenshot of the EcoBuilder

Ontology relations conform to the triple pattern {subject - predicate - object}, where the subject is the entity from which the relation points, the object is the entity to which the relation points, and the predicate is the relation type, e.g. *Movie - owned by - Museum.* Relations can be created between entity instances and are displayed in the scenario tree beneath the subject entities. The object entities of the added relations can be found at the last layer of the scenario tree, beneath the relation entities. Each object entity is equal to an existing subject entity in the tree. It is possible to jump to the corresponding subject entity with a double click on an object entity.

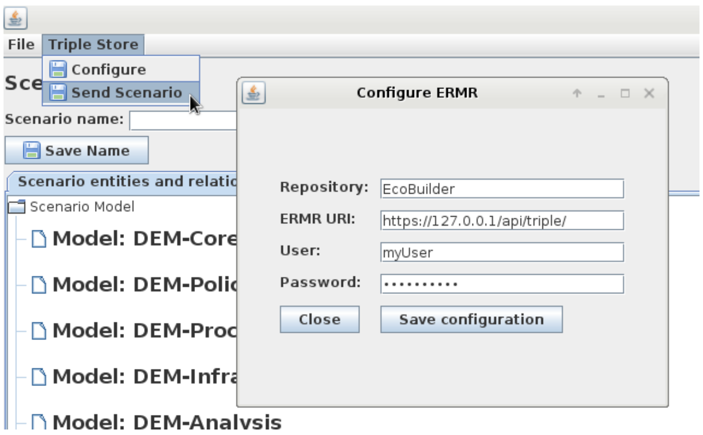
The right side of the GUI shows the entity configuration area. Entities can be configured with a name, a description a version number, and entity template specific configuration options. The templates will ensure that the underlying ontology resources and relations are created correctly, e.g. for each GUI provided description an LRM:Description resource is created and linked with the entity via a LRM:describedBy relation.

Also a list of applicable relations for the tree-selected entity can be found at the right configuration area. The object entity of the relation can be selected from a list, which contains all existing tree entities that are in the range of the relation.

The instantiated scenario model is generated from the scenario tree, once the user saves the scenario. It is stored as both Turtle and OWL/XML file. The final scenario model can be send to the ERMR's triple store via an option available at the Triple Store top-menu.

6.3.1.1. Connecting to the ERMR

The EcoBuilder provides an interface to send the created models directly to the ERMR's triple store. The connection information can be configured via GUI, as shown in Figure 13..



Repository: EcoBuilder
ERMR URI: https://127.0.0.1/api/triple/
User: myUser
Password: ••••••••••

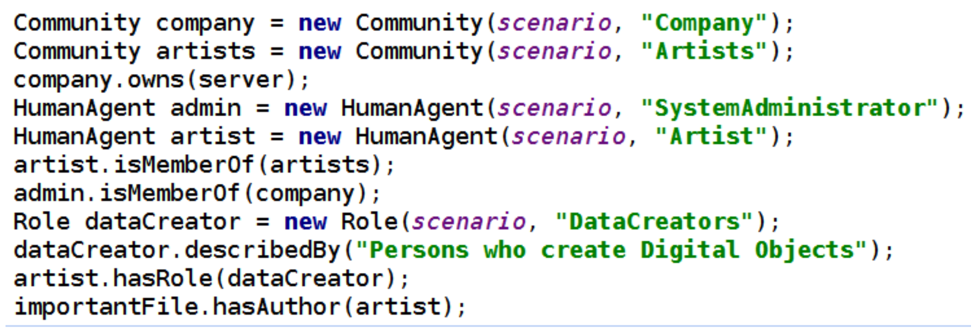
Figure 13: Screenshot of the EcoBuilder's ERMR connection

Models residing in the ERMR can be further processed by other PERICLES components.

6.3.2. Using the Java API

The EcoBuilder provides a Java abstraction layer for the DEM containing Java classes which depict the ontology resources. On Java level the template classes for the entities and relations can be accessed directly to create instances of scenario entities, or for the creation of customised templates. They are ordered by their corresponding sub-ontologies. The EcoBuilder imports the Java Jena API for dealing with the underlying ontology concepts. This API can also be used by developers to integrate ontology concepts into the scenario model that are not provided by the EcoBuilder's API.

The DEM relations are also provided as template classes, that define the domain entities (from which the relation can point) and the range entities (to which the relation can point), e.g. the "owns" relation can only point from entities of the types "Human Agent" and "Community" to other DE entities. Figure 14 shows an example for using the API. The entity templates provide methods for the fast creation of the most common relations in a best practice fashion.



```
Community company = new Community(scenario, "Company");
Community artists = new Community(scenario, "Artists");
company.owns(server);
HumanAgent admin = new HumanAgent(scenario, "SystemAdministrator");
HumanAgent artist = new HumanAgent(scenario, "Artist");
artist.isMemberOf(artists);
admin.isMemberOf(company);
Role dataCreator = new Role(scenario, "DataCreators");
dataCreator.describedBy("Persons who create Digital Objects");
artist.hasRole(dataCreator);
importantFile.hasAuthor(artist);
```

Figure 14: Using the EcoBuilder's Java API to create entities

Modelling with the Java API is on the one hand more difficult than using the GUI, on the other hand it enables to adjust the provided templates for scenario requirements, and to load concepts from external ontologies via Jena. The Java API offers more liberties than the GUI, therefore we got the best resulting models by using the EcoBuilder's Java API.

As the EcoBuilder is open source, it can be extended by developers to support other ontologies required for a designated scenario.

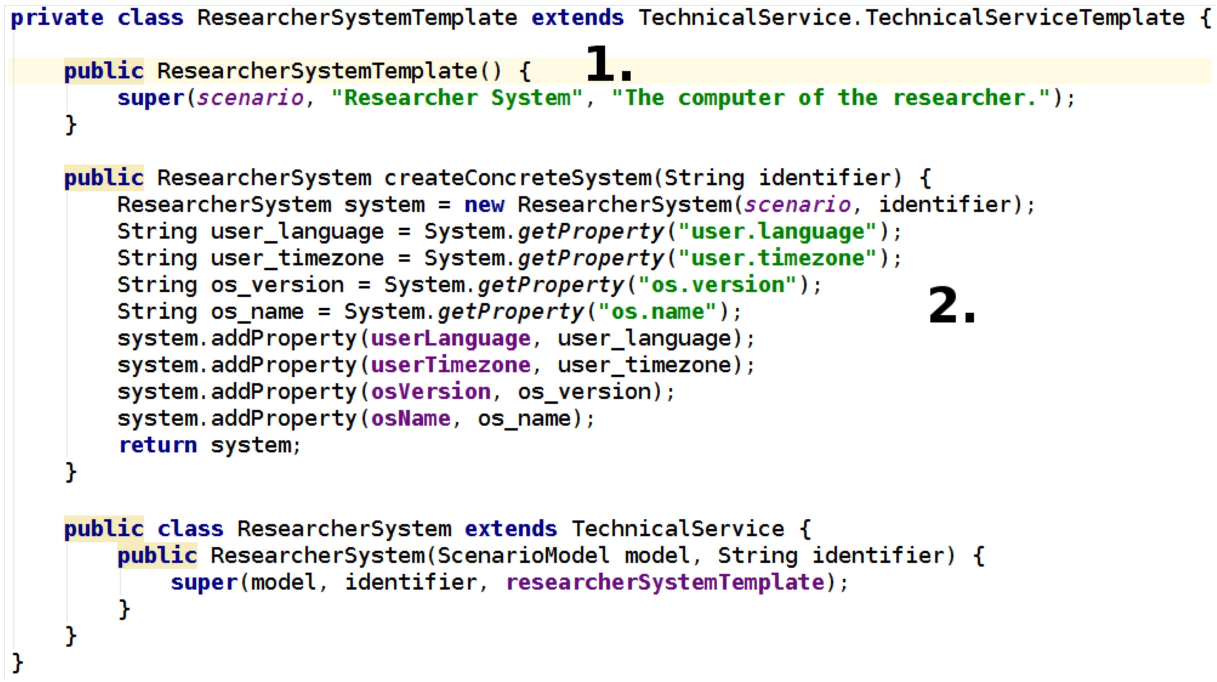
6.3.3. Example use in the PERICLES Project

In this section we describe how the EcoBuilder has been used to automatically update the DEM with environment information extracted by the PERICLES extraction tool. This is one example of how the EcoBuilder can be used in a PERICLES context. Later in this document we describe how the tool can be used in the context of the PERICLES Policy & QA approach.

The PERICLES tools can be used to achieve an almost automated workflow of model change management, in which digital ecosystem changes are propagated into the models automatically.

The PERICLES Extraction Tool (PET) can extract Significant Environment Information from the environments of Digital Objects in a continuous extraction mode, which triggers the extraction on environment changes, as described in D4.1. This extracted information can be used to update the DEM in case of changes of the underlying digital ecosystem, where a mediator script uses the EcoBuilder's Java API to connect PET and the DEM.

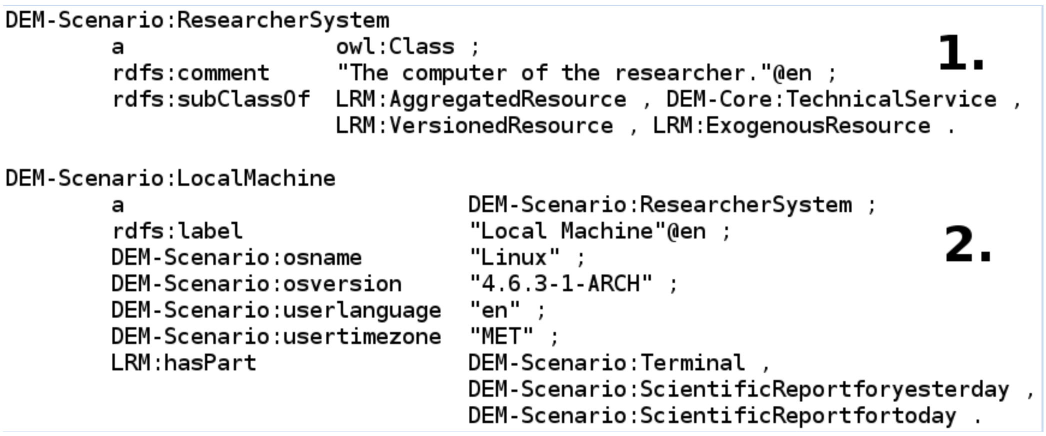
The mediator script defines the custom entity and relation templates for the scenario using constructs from the EcoBuilder API. If for example a *Researcher System* template is needed, then it can inherit the principles from the *Technical Service* template provided by the EcoBuilder to create DEM Technical Service resources. This template can include methods for the creation of specific relations and it can be used to build an arbitrary number of *Researcher System* instances. Those instances can be created automatically and enriched with live extracted information, once a change of the underlying system was detected by PET. The templates define which information is valuable for the enrichment of entities, and the mediator script executes the parsing of the information and enters it into the templates during the instantiation. Furthermore, the mediator script manages the automatic creation of relations and dependencies to the other existing entities.



```
private class ResearcherSystemTemplate extends TechnicalService.TechnicalServiceTemplate {

    public ResearcherSystemTemplate() {      1.
        super(scenario, "Researcher System", "The computer of the researcher.");
    }

    public ResearcherSystem createConcreteSystem(String identifier) {
        ResearcherSystem system = new ResearcherSystem(scenario, identifier);
        String user_language = System.getProperty("user.language");
        String user_timezone = System.getProperty("user.timezone");
        String os_version = System.getProperty("os.version");
        String os_name = System.getProperty("os.name");                    2.
        system.addProperty(userLanguage, user_language);
        system.addProperty(userTimezone, user_timezone);
        system.addProperty(osVersion, os_version);
        system.addProperty(osName, os_name);
        return system;
    }

    public class ResearcherSystem extends TechnicalService {
        public ResearcherSystem(ScenarioModel model, String identifier) {
            super(model, identifier, researcherSystemTemplate);
        }
    }
}
```

Figure 15: (1) Scenario specific template inheriting EcoBuilder templates. (2) Enrichment of entities with extracted information during initialisation.

Figure 15 shows a screenshot of the *Researcher System* template which inherits from the *DEM:Technical Service.* The initialisation method for instances contains a simple way to enrich the entity instances with extracted environment information.



```
DEM-Scenario:ResearcherSystem
        a                owl:Class ;                                          1.
        rdfs:comment     "The computer of the researcher."@en ;
        rdfs:subClassOf  LRM:AggregatedResource , DEM-Core:TechnicalService ,
                         LRM:VersionedResource , LRM:ExogenousResource .

DEM-Scenario:LocalMachine
        a                          DEM-Scenario:ResearcherSystem ;
        rdfs:label                 "Local Machine"@en ;                       2.
        DEM-Scenario:osname        "Linux" ;
        DEM-Scenario:osversion     "4.6.3-1-ARCH" ;
        DEM-Scenario:userlanguage  "en" ;
        DEM-Scenario:usertimezone  "MET" ;
        LRM:hasPart                DEM-Scenario:Terminal ,
                                   DEM-Scenario:ScientificReportforyesterday ,
                                   DEM-Scenario:ScientificReportfortoday .
```

Figure 16: Extract of the resulting model showing (1) the template entity of the Researcher System, and (2) a Local Machine which is an instance of this template

Figure 16 shows an extract of the resulting model created by the EcoBuilder from the script shown in

Figure 15. The *Researcher System* template entity inherits from LRM and DEM resources, and is saved in the resulting scenario model once, while there can be an arbitrary number of instances.

Broader information extracted by PET can be added to models via mediator scripts that include the parsing of the information into the required formats. Mediator scripts can be further responsible for the ingest of the models into the repository, and in combination with triggers for sending notification in case of digital ecosystem or model events.

See also the chapter about modelling strategies and mediator scripts at D3.5.

6.4. Obtaining the EcoBuilder

The EcoBuilder is an Open Source (Apache v2) Java tool built on top of the Digital Ecosystem Model (DEM), which is described in D3.5.The EcoBuilder will be published on GitHub by the end of October 2016 under the URL https://github.com/pericles-project/EcoBuilder.

6.5. Conclusion and future plans

The EcoBuilder tool enables scenario experts, which are not ontology experts, with the creation of DEM models. The tool can output the model as file or offers a direct connection to ERMR to store the models there. Using the tool ensures that a valid model is created. This is made possible by providing templates for the ontology constructs. The downside of the simplification is that it does not offer all options from the ontology. Details and domain specific extensions can be added afterwards, if required. It provides a GUI and an API. The API enables the possibility to integrate the tool, e.g. for connecting it to a notification system to populate change into the DEM model.

Ideas for future work are adding the ability to load domain specific ontologies via the GUI to add specific details about entities if needed and representing the created model as a graph.

PART TWO

APPROACHES

7. Approaches for Policy, QA, Model Driven Preservation and Change Management

7.1. Final Policy Model and Guidelines

This section presents the final version of the policy and QA (Quality Assurance) model, which was introduced in Deliverable 5.2, and is part of task T5.3.1. The final version takes into account the more recent discoveries made while developing the prototypes (T5.3.2), and the helpful feedback and discussion from the recent workshops, introducing also a more detailed model of QA as a new, separate entity. The model is designed to support QA methodologies that can validate and ensure policies are correctly applied and complied with, within a given digital ecosystem. Furthermore, the policy framework can include the definition of change management rules for policy change management, as described later in this deliverable in Section 7.2. While requiring more investment in their creation, these rules react to some types of change and adjust the ecosystem so that policies remain valid and correctly implemented. The policy and QA model described in this section has been implemented in the DEM and can be stored and used in the ERMR.

7.1.1. Modelling Ecosystems for Policy Compliance and QA

When considering policy and QA implementation, we decided to focus on pragmatic approaches that facilitate implementation and reuse of existing infrastructure, saving cost and time of implementation. Following the intuition behind the DE modelling approach, we define policy and QA to have minimum requirements on the technical infrastructure and their specific implementation. Policy modelling does not replace any enterprise architecture, but is thought of as a thin layer on top of the Digital Ecosystem Model, defining clearly and unambiguously policies, their implementation, dependencies, constraints and validation methods. This was recognised as a promising approach during at the Brussels and IDCC 2016[7] workshops.

The issue of implementing policies using formal languages and specific technologies is a high barrier to adoption, because of initial difficulty of learning and migrating to often uncommon technologies. Currently, formal language policies are very domain specific, and QA is usually quite limited and focused on the basic aspects of file-format migration, or left as manual work of developers, system administrators and practitioners. For this reason, our model doesn’t make any strong assumption on underlying technologies and languages.

At the basic level, our model supports the description of the policies, QA methods, and their dependencies in human readable form enable users to communicate and define requirements, to record and share the knowledge and decisions taken when implementing policies. Since policies include, by our broad definition, also aspirational policies, our model can help communicate general objectives of an organisation, and how these map to concrete infrastructure and requirements, and

[7] Brussels second evaluation workshop, Oct. 2015; IDCC workshop, Feb. 2016

is not limited to defining constraints and mandatory practices. This is an important record and communication tool per se.

Furthermore, we propose two different methodologies for policy implementation and QA.

One is described in general terms in section 7.2, and makes use of formal (rule or action) languages to provide automated change management. It can be implemented using PERICLES technologies (LRM, DEM, SPIN[8] rules.) as illustrated in the proof of concept (see section 7.4), or other technologies.

The second approach introduced in D5.2, and extended here in Appendix 10, is an the automated, non-formal approach for policy and QA validation based on the definition of QA methods driven by the Digital Ecosystem Model and automated by triggers reacting to change.

7.1.2. Top Down Policy Implementation Methodology

We propose a simple, three step methodology, starting from a high level view, in order to describe and implement policies:

1) Model the existing architecture, policy and QA methods using the DEM and the policy derivation method
2) When possible and effective: use an automated and formal approach based on the DEM approach and rules (section 7.2)
   a) Define policies and QA methods using rule-driven change management
   b) The DE analysis will describe dependencies and change of different type
3) When more convenient: use a semi-automated approach (based on free-form processes or human intervention), and enrich it with QA methods. These methods can be linked to DE model change by triggers and risk assessment. (See example in Appendix 10.)
   a) Implement the policy freely, in any existing architecture and language; use plain text descriptions for human-driven processes.
   b) Define QA methods for policy and where possible, implement them.
   c) Use automated validation via QA methods, and change management based on the DEM.

We recommend defining policies and QA methods always in natural language. When possible, the model supports the definition of triggers for the different processes based on changes to the digital ecosystem entities.

The triggers can be time-based, or event-based, where the events can represent change in the digital ecosystem. An LRM based implementation is shown in the next section; other trigger implementations are easily defined with a different event and trigger methodology.

7.1.3. Final version of the Policy model

Recent discussion (*IDCC workshop, Feb 2016*) made clear the importance of representing explicitly aspirational policies. These convey what the organisation aims to archive, but currently does not implement, and help to drive developments by explaining the interests and future directions. For this

[8] http://spinrdf.org/

reason, we introduce the type of a policy (represented with *Policy Type*), clearly stating when policies are mandatory, partially implemented, or just aspirational and not implemented. Other polices represent legal requirements, meaning that these cannot be ignored and must have an implementation. Independently of the type, all policies are relevant and important to record.

The policy data model is defined independently of a specific ontology, but it has been implemented in the Digital Ecosystem Model and EcoBuilder.

Policies can serve as communication tools, explaining how the institution sees specific issues. For this reason, policies at different levels should always include a detailed, human readable description, allowing different roles in the organisation to understand them. Since policies can be partially implemented, for a number of reasons (lack of resources, priority, technical infeasibility), we make explicit the real implementation state of policies.

Some policy and QA method implementations can be automated, but still require human validation, while others can be completely automated or completely manual. This is also explicit in the model.

Below is the final definition of the policy model:

- **Identifier**: a unique identifier for the policy
- **Name**: a user-friendly, not necessarily unique, informal name
- **Version:** version number (it can use the LRM versioning mechanism)
- **Policy type: mandatory (e.g. by a funding body), legal requirement (law, such as Freedom of Information Act[9]), aspirational (principles driving the institution), business driven (what we do - our business)**. Not all policies are equal - mandatory ones must be implemented to satisfy law or other requirements, others are aspirational, and most are met with the best possible effort.
- **Level**: what the policy level is. For general policies, we have defined the following levels, based on the SCAPE policy levels, changed in a generic way so that they are not specific to Digital Preservation.
  Guidance: high level principles and general objectives driving an organisation, the most abstract level of policy.
  Procedure: lower level policies that gives detail of how the policy is implemented without strong dependencies on the infrastructure
  Control: low level description of the policy that includes reference to the specific infrastructure. For digital preservation, we make use of the SCAPE[10] policy levels.
- **Policy statements:** detailed definitions of the policy contents as text (formal or non- formal). A natural language, human readable statement must be always provided for any policy, so that the policy can be understood by anyone in the organisation; this way the policy framework can be used as a communication tool across an organisation.

  Format: “formal”; or “non-formal” (free text)

  Language: the language used for the policy definition (natural language, ReAL, SPIN, SWRL,

[9] https://en.wikipedia.org/wiki/Freedom_of_Information_Act_2000
[10] http://wiki.opf-labs.org/display/SP/SCAPE+Policy+Framework

etc.)

- **hasQAcriteria:** reference to the QA criteria implementation described in QA criteria entity (next paragraph).
- **Classification**: defines the category of policy; domain specific. For preservation policies the SCAPE catalogue of policy elements guidance level classification[11] can be used as a reference.
- **Policy authority:** the entity that mandates or generates the policy. The authority could be also a reference to a legal requirement (in case that a policy is mandated by a legal requirement) or a directive. The authority reference is here to trace who had the authority to generate the policy.
- **Responsible**: responsible for the application of the policy (person or role)
- **Sub-policies**: policies that are a more detailed specification of the parent policy as described in the policy derivation process (D5.2)
- **Implementation**: reference to processes implementing the policy.
- **Requirement Level:** what is the desired level of compliance of the policy (must, should, must not); as defined in RFC 2119[12]
- **Implementation state**: how deeply the policy is currently implemented (**implemented, partially implemented, unimplemented, not-implementable**). It is necessary to represent policies that are important as guidance, but can't currently be implemented.
- **Validity information:** any guidance to the policy lifecycle: Valid from; Valid to
- **Conflict detection attributes**: map of attributes for conflicting policies detection (see paragraph 5.7 in D5.2)
- **Target entities**: references to entities that are subject of the policy (depending on the policy level, it could consist of a free text description, a query, or a list of entities)
- **Target user community:** the user community the policy has been designed for
- **Automation status: (manual, automated with human intervention, fully automated).** Specifies if a policy implementation requires human intervention.
- **Replaced policy**: in case a new policy is created in order to replace an old one
- **Policy validation status:** the property serves to indicate the current status of a policy in the ecosystem according to the defined QA criteria: valid (currently respected); non valid (currently not respected); not decidable.
- **Trigger**: what will trigger the policy validation. A trigger can also be a reference to a length of time for recurrent triggering. Triggers can be defined in response of different events, including change events. Triggers can be implemented using the LRM dependency's precondition-impact properties, and SPIN or ReAL.
- **Drift threshold**: defines the drift value that will activate drift evaluation. This is specific for semantic and community drift, so that changes in concepts or community topic of a certain level can be reported and manually validated. An example making use of the Drift threshold is included in Section 7.4.

[11] http://wiki.opf-labs.org/display/SP/Catalogue+of+Preservation+Policy+Elements
[12] https://www.ietf.org/rfc/rfc2119.txt

7.1.4. Quality Assurance Criterion Model

We decided to separate the description to the QA aspects with details that help to describe both automated, and manual QA processes.

- **Format**: formal; or non-formal (free text)
- **Language**: the language used for the criteria definition
- **Implementation:** reference to the processes or rules implementing and enforcing the QA criterion
- **Statement:** human or automated QA process description. The statement describing the QA criteria. A human readable statement MUST be provided to help communication.
- **Trigger**: same definition as the policy model
- **Process**: the reference to a process (can be also a human process) that can validate the criteria.
- **Implementation state:** same definition as the policy model
- **Automation status:** same definition as the policy model
- **Responsible** (**person**): responsible for the application of the QA method
- **Target entities**: The target entities of the QA. A particular QA method may apply only to a subset of the overall policy target; by default, if no target entity is specified, it is assumed to be the one of the policy.

7.1.5. Integration of the Policy Model into the Digital Ecosystem Model

The policy and QA model described here, is integrated in the Digital Ecosystem Model (DEM) (a cross-task effort involving Task 5.3.2 and WP3) and can be generated using the EcoBuilder tool. Details of the DEM policy model implementation are described in D3.5 and are not reported here.

An example that applies the implementation of the policy and QA model using the EcoBuilder in a policy driven modelling approach can be found in Appendix 10. It models the DE of CERN preservation policies including policy derivation QA.

7.2. Rule-Based Change Management for Ecosystem and Policy

The Linked Resource Model (LRM) [D3.2, D3.3] is a common base ontology language for change management. Thanks to the concepts of precondition and impact, as means to handle change in the digital entities through the concept of dependency, it allows management and propagation of change in digital ecosystems. When change happens in digital ecosystems, these LRM features allow defined methods to react and propagate change to the target and dependent entities. The precondition defines the conditions that have to be satisfied in order to activate a dependency, while the impact defines the consequences of the dependency activation. By defining dependencies that make use of these constructs, we propose to implement policies, QA and change management at the model-level, expressed as constraints on entities in the corresponding LRM model.

In order to accomplish this type of policy implementation, it is necessary to have support for rule languages at the model level, such as the ReAL language[13], or the W3C SPIN rule standard[14]. In line with the rest of this task, what we propose here is generic, and can be implemented using different technologies. In section 7.4 we provide exemplar implementation in SPIN.

In line with the ideas and principles we described in the previous section, we here describe a QA and change management methodology for policy and ecosystem entities.

7.2.1. Requirements and Functional Description

In order to implement rule-based change management, a proper architecture for change management must be in place. We are describing such architecture here, in general terms but with a reference to the PERICLES components that implement that functionality.

The components are listed here and their relationship is represented in Figure 17.

- **Repository event listener**: at the repository level, a system needs to be in place in order to register the basic operations and changes that can happen to digital objects. These include

[13] This ontology language will be described in the deliverable D3.4 Language for Change Management (due M46)

[14] https://www.w3.org/Submission/spin-overview/

all changes that can happen in the data repository, such as the typical CRUD operations: Create, Read, Update, Delete. In PERICLES, this is represented by the ERMR component, described in Section 3, which can generate and share events happening in the repository. Events will be reported to the DE model updater component.

- **Digital Ecosystem model updater**: this component will listen to change events in the DE (as reported by the event listeners), and update the ecosystem model accordingly.
- **Digital Ecosystem listener:** A listener that reports change events into local (end user) computers, or other type of changes in other concrete ecosystem entities, such as the concrete services or SW components. Events will be reported to the DE model updater component.
- **Digital Ecosystem Model listener**: changes in the **Digital Ecosystem Model** instances must be observed and reported to the relevant component; this will allow the implementation of precondition-impact and rule-based change management. In PERICLES such functionality will be covered by the LRMS and by a change observer for a SPIN rule engine. The listener receives change events from the model updater. Changes can be expressed using LRM deltas[15].
- **Model consistency checker**: validates the consistency of the model and its dependencies; this can be covered by the LRMS.
- **Rule or precondition-impact engine**: triggers the impact when the preconditions are verified. In concrete terms, this is covered by the LRMS (LRM Services) for ReAL, or by a SPIN rule engine for SPIN.
- **Process execution layer**: executes the processes, which are triggered by the rules. This can be simple, direct processes, or more complex workflows. In PERICLES the process compiler and workflow engine implement such functionality.

[15] See PERICLES D3.3

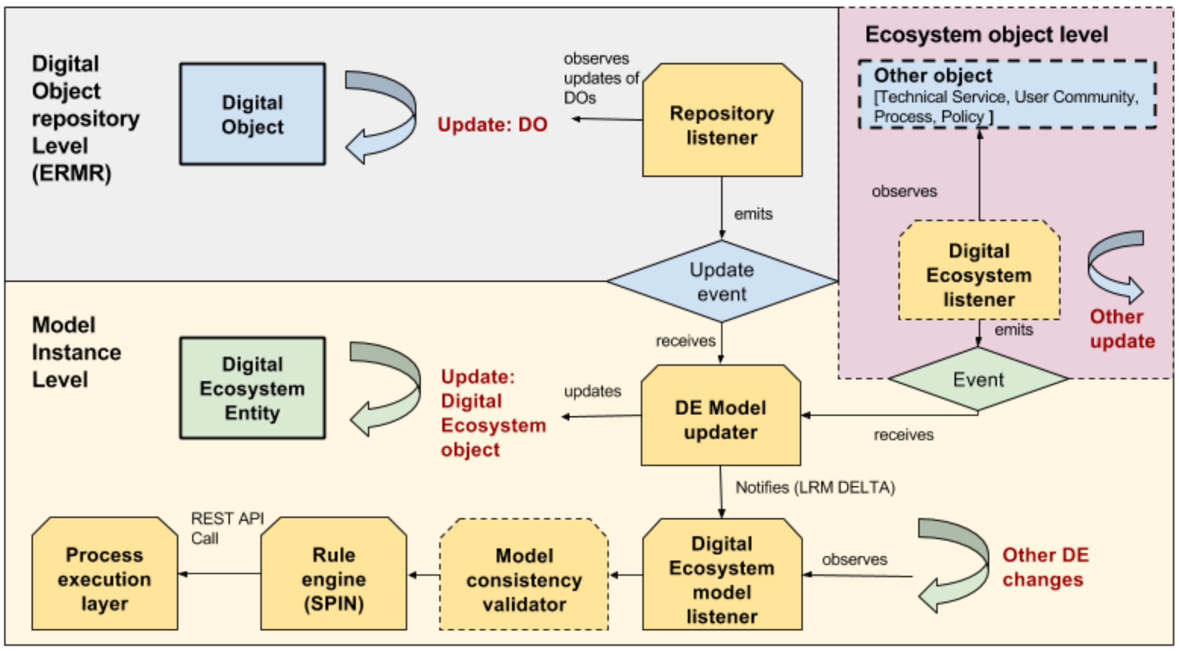


Figure 17: Change management architecture: event propagation and components

7.2.2. Detailed methodology description

Concrete implementation of policies for change management will be highly dependent on the use case. In order to give an understandable, familiar example, we describe in this section the generic approach, exemplified with a simple DP scenario that can be used as a template: the issue of keeping data accessible.

The LRM model defines change to entities in the ontologies using deltas (`lrm:RDF-Delta`) [PERICLES D3.3, 2015], [D.4.4 chapter 4]. Deltas provide meta-information about the modification of a resource, by defining a list of triples that have been deleted and added to the model. In our methodology we assume that such deltas are reported by the DEM updater component and are added to the Digital Ecosystem Model using the LRM Delta notation.

These guidelines[16] drive the construction of an ecosystem model that can manage change through the use of precondition-impact rules.

1) Define the Digital Ecosystem Model and create an ecosystem instance (**DEM**);
2) Define policies in the Digital Ecosystem Model; (**policy**)
3) Connect the policy definition to all the entities that are covered by the policy (targets) using an LRM dependency; (**policy** dependencies)
4) (Optional) dependencies can be automatically created and destructed by respective constructor and destructor rules in the policy. This is useful in cases where the dependencies (linking the policy targets) are dynamic, for example when they are defined on a criterion (e.g. “all video files in a collection”), as opposed to a fixed set of entities. (**appliesTo** dependency in Figure 18)
5) LRM disjunctive dependencies can be used to express alternatives, when multiple components or

[16] The names between brackets indicate the relevant entity in next section’s example.

solution can address the same task; (**players** dependency in Figure 18)

6) The central dependency contains change management rules that will handle change in the relevant entities (**canPlay** dependency in Figure 18)
7) Additional dependencies define and implement QA methods for other entities involved in the process, e.g. unit tests or manual dependency checks to be executed upon change (**uses** dependency)

Notes on the model:

- Thanks to the precondition-impact in the dependencies, the policy will be automatically enforced on all involved resources upon change in the entities, and the QA methods for the policy executed.
- Precondition must not generate any change, as this would generate uncontrolled side effects. This is also a constraint defined in the LRM model.
- The impact can trigger processes, which can in turn change the models or the concrete Digital Ecosystem entities (that will be reflected on the model).
- Dependencies can either be created by testing constraints on the properties of the to and from resources or by running a process to enforce a certain condition to hold.
- Policies can enact processes directly through dependencies to resources, or create new dependencies.

7.2.3. Methodology description for policies dependant on two entities

In order to better illustrate the change management principles, as we have outlined in the previous section, we describe a complete example based on the Digital Video Art ontology and scenario [D2.3.2 and part of D6.6 (pending)]. For readability purposes, this is a simplified example.

7.2.3.1. Scenario

The scenario is based on the high-level policy for preservation of digital media components at Tate, stating "**At least one version of the media components must be playable on a player**". This is a policy that aims to ensure access, taking into account technology evolution and file format obsolescence, which is a common issue in Digital Preservation. We have reformulated the policy, to make the scenario more explicit, into a lower level policy: "**A collection of digital videos has to be kept playable from at least one from a set of video players**."

This specialisation makes clear those two resources involved: the set of *video files*, and the set of *media players*. This type of policy can be adapted to any situation where a data file needs to be processed from a set of processing (rendering, transformation, etc.) software.

The example implements mitigating actions triggered by the precondition-impact part of dependencies when critical conditions arise, as shown in Figure 18. More specifically, the *Change2* destructor in the *canPlay* dependency will check on dependency deletion, if there is no longer any player capable of rendering the video file. In that case format migration (transcode) will be issued in order to keep video playable. In the case of Tate media components, before final transcoding, the video would be submitted to a human process of quality assurance, given the high value and low volume of the resources justifies the high cost of human intervention. In other situations, involving higher volume, lower value entities, this step could be skipped and the process could be completely automated. The selection of the parameter *AFormat* for the transcode process (that defines the target format for transcoding) could be decided by the human involved in the QA process, or specified in the policy itself, or be determined by the list of video players (for example choosing a format supported by the largest number of video players), or again be based on risk analysis.

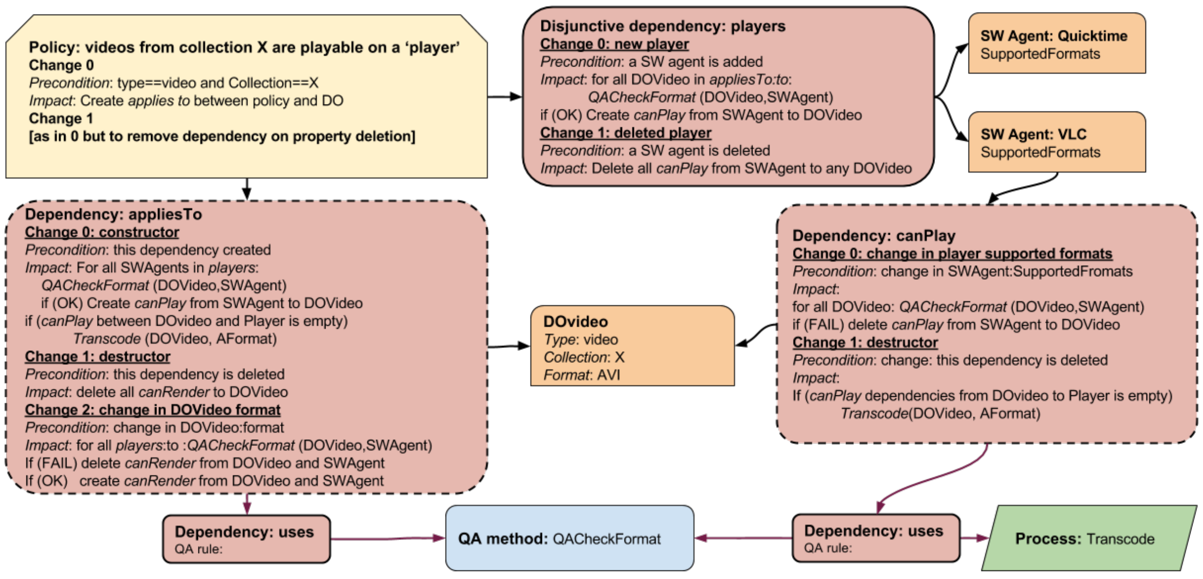

Policy: videos from collection X are playable on a 'player'
Change 0
Precondition: type==video and Collection==X
Impact: Create applies to between policy and DO
Change 1
[as in 0 but to remove dependency on property deletion]
Disjunctive dependency: players
Change 0: new player
Precondition: a SW agent is added
Impact: for all DOVideo in appliesTo:to:
QACheckFormat (DOVideo,SWAgent)
if (OK) Create canPlay from SWAgent to DOVideo
Change 1: deleted player
Precondition: a SW agent is deleted
Impact: Delete all canPlay from SWAgent to any DOVideo
SW Agent: Quicktime
SupportedFormats
SW Agent: VLC
SupportedFormats
Dependency: appliesTo
Change 0: constructor
Precondition: this dependency created
Impact: For all SWAgents in players:
QACheckFormat (DOVideo,SWAgent)
if (OK) Create canPlay from SWAgent to DOVideo
if (canPlay between DOvideo and Player is empty)
Transcode (DOVideo, AFormat)
Change 1: destructor
Precondition: this dependency is deleted
Impact: delete all canRender to DOVideo
Change 2: change in DOVideo format
Precondition: change in DOVideo:format
Impact: for all players:to :QACheckFormat (DOVideo,SWAgent)
If (FAIL) delete canRender from DOVideo and SWAgent
If (OK) create canRender from DOVideo and SWAgent
DOvideo
Type: video
Collection: X
Format: AVI
Dependency: canPlay
Change 0: change in player supported formats
Precondition: change in SWAgent:SupportedFormats
Impact:
for all DOVideo: QACheckFormat (DOVideo,SWAgent)
if (FAIL) delete canPlay from SWAgent to DOVideo
Change 1: destructor
Precondition: change: this dependency is deleted
Impact:
If (canPlay dependencies from DOvideo to Player is empty)
Transcode(DOVideo, AFormat)
Dependency: uses
QA rule:
QA method: QACheckFormat
Dependency: uses
QA rule:
Process: Transcode

Figure 18: Digital Video Art ecosystem: rule-based change management for 2 entities

Processes used in the example

Process	Parameters	Description	Returns
QACheckFormat	A (video), B (rendering software)	Tests if the format of A can be rendered by B.	pass/fail
ProcessTranscode	Image Resource, AFormat	Transcodes the image to the selected format	pass/fail

Change scenarios

Given the DE describing the scenario, and relative rules implemented and submitted to a rule engine and relevant architecture (Figure 17), what follows describes the different type of change and the automated change management put in place by the system.

Change in supported players (*players* dependency)

A player is added or removed from the *supported players* dependency, to indicate that it is one of the supported (or no longer supported) players in the current scenario.

Add a new video player to players dependency

This type of change will activate *change 0* precondition in *players* dependency. The impact will check, for each of the targets of the *appliesTo* dependency (all the video files subject to the policy) whether the new player can play the video file, using the *QACheckFormat* process[17]. If the video file is supported, the rule will create a *canPlay* dependency between player and video file and will not propagate further. This will create the dependencies between video player and playable video files.

Delete an existing video player from player dependency

This change can be due to a user action, as when a video player is deprecated; or it could be a step in a chain of changes to external entities in the DE (propagated change), for example when an operating system update can make a particular video player no longer usable, and a dependency impact removes the player from the *players*.

Step A1 - SWAgent dependency target deleted

This event will trigger the impact of *change 1* in *players*, in turn deleting all the *canPlay* dependencies between the SW and the single video files (Figure 19).

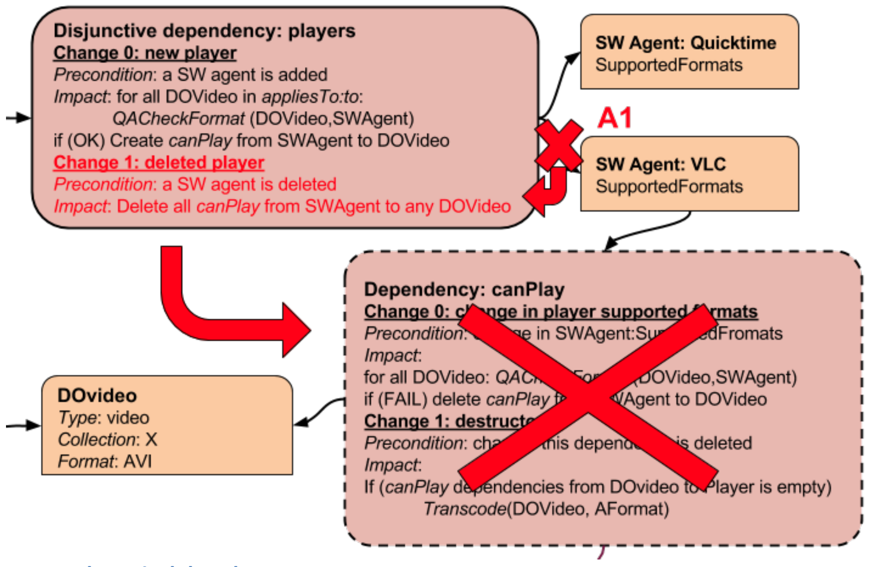

Disjunctive dependency: players
Change 0: new player
Precondition: a SW agent is added
Impact: for all DOVideo in appliesTo:to:
QACheckFormat (DOVideo,SWAgent)
if (OK) Create canPlay from SWAgent to DOVideo
Change 1: deleted player
Precondition: a SW agent is deleted
Impact: Delete all canPlay from SWAgent to any DOVideo
SW Agent: Quicktime
SupportedFormats
A1
SW Agent: VLC
SupportedFormats
Dependency: canPlay
Change 0: change in player supported formats
Precondition: change in SWAgent:SupportedFromats
Impact:
for all DOVideo: QACheckFormat(DOVideo,SWAgent)
if (FAIL) delete canPlay from SWAgent to DOVideo
Change 1: destructor
Precondition: change this dependency is deleted
Impact:
If (canPlay dependencies from DOvideo to Player is empty)
Transcode(DOVideo, AFormat)
DOvideo
Type: video
Collection: X
Format: AVI

Figure 19: Step A1 - one player is deleted

Step A2 - canPlay dependency is deleted

This will in turn activate the *canPlay change 2* (destructor). This will verify if there is any other dependency between the video file and a player, indicating that the video is playable (policy requirement). When that is not the case, the impact will execute the video transcoding process, in order to make the video playable from another player (Figure 20).

[17] This process could itself be implemented in a number of ways, e.g. a simple check for the format being in the supported format list for the player, or as a more complex validation that renders the video file through the player and checks for possible errors.

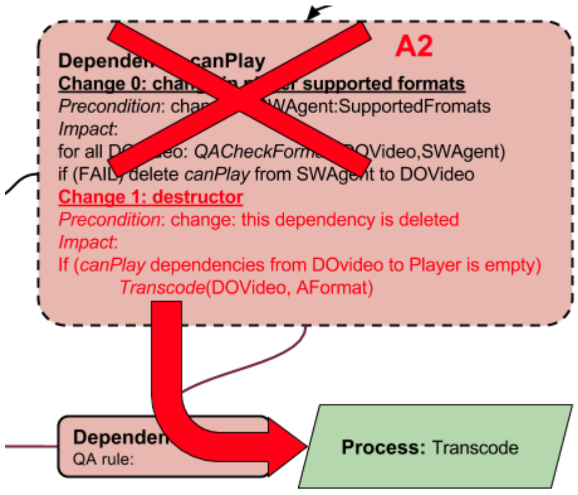

A2
Depende canPlay
Change 0: chan r supported formats
Precondition: cha WAgent:SupportedFromats
Impact:
for all D eo: QACheckForm OVideo,SWAgent)
if (FAIL) delete canPlay from SWAgent to DOVideo
Change 1: destructor
Precondition: change: this dependency is deleted
Impact:
If (canPlay dependencies from DOvideo to Player is empty)
Transcode(DOVideo, AFormat)
Dependen
QA rule:
Process: Transcode

Figure 20: Step A2: the video is transcoded

Step A3 - change in DOVideo format

The transcoding will act on the video file modifying its format. This change will be reported by an event stating that the property *format* of *DOVideo* changed (described by an LRM-Delta). This change will be reported by the repository listener through an *update event*, meaning that such changes can be activated also by other type of changes coming from outside of the digital ecosystem model and rules (as for example in the case of a manual transcoding of a single video file).

This will activate change 2 of the *appliesTo* dependency and will thus create the new *canPlay* dependencies and delete no longer valid ones. Assuming that the video has been transcoded to a supported video format, there will now be a player for the specific video file (represented in the DE by a *canPlay* dependency).

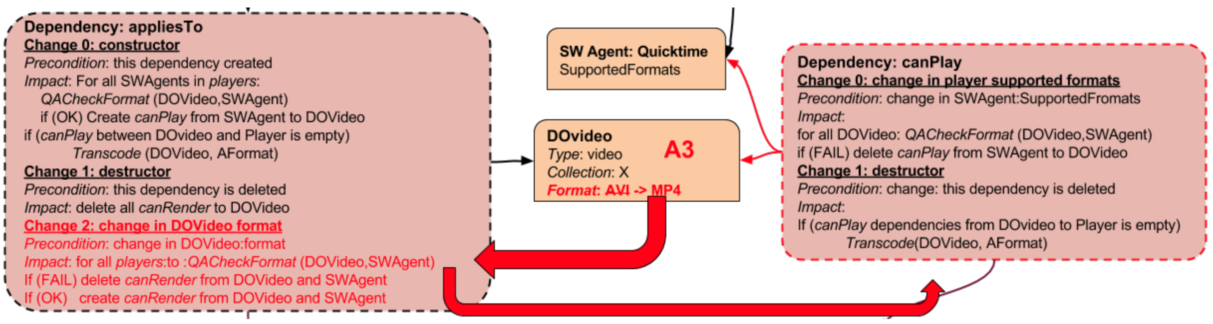

Dependency: appliesTo
Change 0: constructor
Precondition: this dependency created
Impact: For all SWAgents in players:
QACheckFormat (DOVideo,SWAgent)
if (OK) Create canPlay from SWAgent to DOVideo
if (canPlay between DOvideo and Player is empty)
Transcode (DOVideo, AFormat)
Change 1: destructor
Precondition: this dependency is deleted
Impact: delete all canRender to DOVideo
Change 2: change in DOVideo format
Precondition: change in DOVideo:format
Impact: for all players:to :QACheckFormat (DOVideo,SWAgent)
If (FAIL) delete canRender from DOVideo and SWAgent
If (OK) create canRender from DOVideo and SWAgent
SW Agent: Quicktime
SupportedFormats
DOvideo
Type: video
Collection: X
Format: AVI -> MP4
A3
Dependency: canPlay
Change 0: change in player supported formats
Precondition: change in SWAgent:SupportedFromats
Impact:
for all DOVideo: QACheckFormat (DOVideo,SWAgent)
if (FAIL) delete canPlay from SWAgent to DOVideo
Change 1: destructor
Precondition: change: this dependency is deleted
Impact:
If (canPlay dependencies from DOvideo to Player is empty)
Transcode(DOVideo, AFormat)

Figure 21: Step A3: new canPlay dependency is created

QA methodology

The capability of playing a video file can also be validated by a query that will make sure that all the *DOVideo* targets of the *appliesTo* dependency are also target of a *canPlay* dependency. An incoherent state can be reported to the user, or notify other corrective actions. This step could also be implemented as an additional step in change 2 of the *appliesTo* dependency, executed after format migration. In such a case the end user will be warned, as this might indicate a condition where the file has been transcoded to an unsupported format, and executing transcoding again will not solve the issue but initiate an endless loop. In this situation, the wrong format has been chosen in step **A2**, possibly because no suitable transcoding choice is available in the current ecosystem, and the situation requires human intervention.

Change in Transcode or QACheckFormat processes

The dependencies between the *appliesTo*, *players* and *canPlay* dependencies and the *Transcode* and *QACheckFormat* processes implement QA checks on the processes. These processes can be validated by QA methods (e.g. unit tests, manual checks etc.) to be run when there is some update of processed or dependent external entities.

External change and dependency propagation

A very important and powerful feature available from the LRM model is change propagation. We can illustrate this by extending the digital ecosystem include the *Operating System*. Operating system updates can make an existing video player unavailable. The methodology we describe will react to such external change and address the issue.

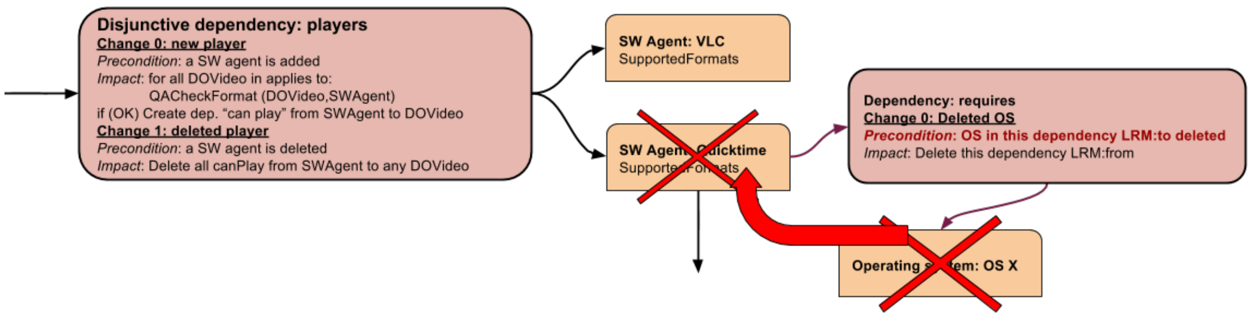

Disjunctive dependency: players
Change 0: new player
Precondition: a SW agent is added
Impact: for all DOVideo in applies to:
QACheckFormat (DOVideo,SWAgent)
if (OK) Create dep. "can play" from SWAgent to DOVideo
Change 1: deleted player
Precondition: a SW agent is deleted
Impact: Delete all canPlay from SWAgent to any DOVideo
SW Agent: VLC
SupportedFormats
SW Agent: Quicktime
SupportedFormats
Dependency: requires
Change 0: Deleted OS
Precondition: OS in this dependency LRM:to deleted
Impact: Delete this dependency LRM:from
Operating system: OS X

Figure 22: External digital ecosystem change and propagation

In Figure 22 we see how an external change (operating system) can drive a change in the DEM model (the *Quicktime* player is no longer available). This change activates the *requires* precondition and impact, removing the player from the list of available players. This will delete the video player propagating in the sequence of change already described in "Delete an existing video player" scenario.

Other type of change and their consequences are described in Appendix 6.

7.2.4. Methodology Descriptions for Policies Dependent on a Single Entity

We start from a policy from space science data, the data policy for the EUMETSAT[18], to illustrate the approach for policy acting on a single resource. This example uses the same concepts illustrated in the previous paragraph. The EUMETSAT's purpose is "*to supply weather and climate-related satellite data, images and products*"[19]. Its data policy[20] defines how the satellite data is made public,

[18] European Organisation for the Exploitation of Meteorological Satellites
[19] http://www.eumetsat.int/website/home/AboutUs/WhoWeAre/index.html

depending on their role and status. From the policy document[22] page 14, we have extracted the following policy for our example, about release of data to free access:

"Meteosat Data and Derived Products older than 24 hours are distributed on request from the EUMETSAT Data Archive in digital and graphical form via the associated operational service in formats which represent both full and partial spatial coverage as well as both full and partial spatial resolution"

In order to define in more detail the ecosystem and policy, we make some assumptions (based on our experience and not describing EUMETSAT services):

1. We assume that initially all data is initially stored into a private repository, accessible only to selected people and organisations (the supporting organisations).
2. A second public repository holds the data accessible to the general public.
3. In order to make the data available, the policy implementation will create a time-based trigger to move the data from the private to the public repository.
4. When the time trigger is issued, the dependency will:
    a. Move the data to the public repository
    b. Create a partial resolution copy

This digital ecosystem (Figure 23) is an effective policy implementation, and allows further defining QA methodologies to ensure correct functioning. The details of this ecosystem are described in Appendix 7.

[20] http://www.eumetsat.int/website/home/AboutUs/LegalInformation/DataPolicy/index.html

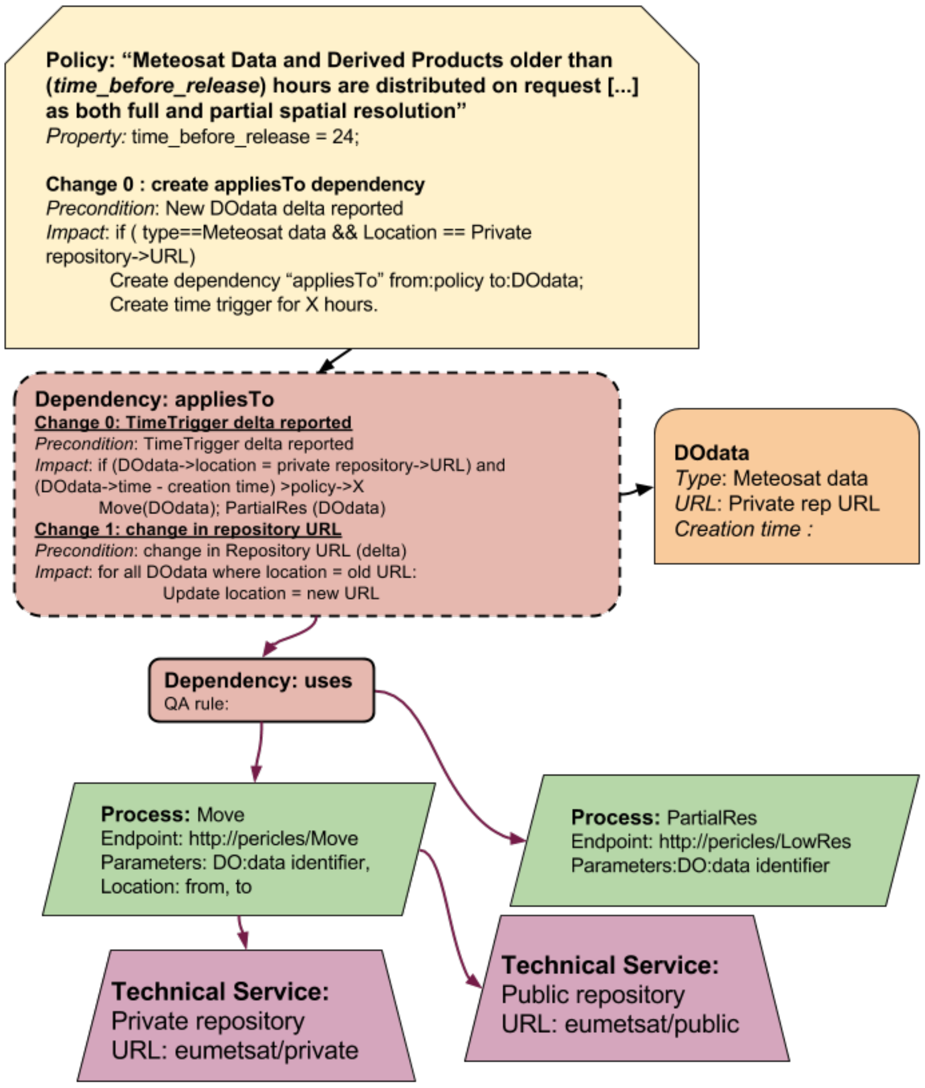

Policy: "Meteosat Data and Derived Products older than (time_before_release) hours are distributed on request [...] as both full and partial spatial resolution"
Property: time_before_release = 24;
Change 0 : create appliesTo dependency
Precondition: New DOdata delta reported
Impact: if (type==Meteosat data && Location == Private repository->URL)
Create dependency "appliesTo" from:policy to:DOdata;
Create time trigger for X hours.
Dependency: appliesTo
Change 0: TimeTrigger delta reported
Precondition: TimeTrigger delta reported
Impact: if (DOdata->location = private repository->URL) and (DOdata->time - creation time) >policy->X
Move(DOdata); PartialRes (DOdata)
Change 1: change in repository URL
Precondition: change in Repository URL (delta)
Impact: for all DOdata where location = old URL:
Update location = new URL
DOdata
Type: Meteosat data
URL: Private rep URL
Creation time :
Dependency: uses
QA rule:
Process: Move
Endpoint: http://pericles/Move
Parameters: DO:data identifier, Location: from, to
Process: PartialRes
Endpoint: http://pericles/LowRes
Parameters:DO:data identifier
Technical Service: Private repository URL: eumetsat/private
Technical Service: Public repository URL: eumetsat/public

Figure 23: EUMETSAT data dissemination policy ecosystem for one resource

Change scenarios

Some of the possible changes to the DE and how the change management will manage these follow:

Change in Digital Objects

Change: new data is produced

When new data is produced, the policy rule will create a dependency *appliesTo* between policy and the data file entity. This rule will also start the time-based trigger that will in fact enact the dependency rule taking care of moving the data to the public repository when allowed.

Change in policy

Change: release period reduced to 12 hours

In that case the rule will automatically use the policy entity property value, and will not need to be updated. This means that changes to the release period are automatically managed.

Change: policy is changed to limit public release to partial resolution copy

The rule needs to be updated so that only the partial resolution copy is moved to the public repository. This requires a simple modification in the impact part of the dependency rule "change 0". This is a manual, but rather simple change from:

```
        Move(DOdata); PartialRes (DOdata);
```

to

```
        Move(PartialRes (DOdata));
```

Change in technical infrastructure

Change in repository URL

We can imagine the situation where a repository (private or public) is updated in case of a change of the Internet domain. The rule *Change 1* (Figure 23) in the *appliesTo* dependency will update the location property of all the *DOdata* objects to the new repository URL. As both the repository URL and the *DOdata* locations are updated, the rule *Change* 0 will still be applied correctly.

7.2.5. Conclusions for Rule-Based Change Management

We have presented a very generic methodology of policy implementation for digital ecosystem that relies on the constructs of LRM dependencies. This methodology uses change propagation to reflect internal and external ecosystem change, giving automated change management. The policy models we presented can be chained for multiple or complex policies, and can be combined with the other PERICLES methodologies, for managing situations that include both manual and automated actions. The methodology is suitable for complex cases, as it is applied by focusing on single policies and dependencies, by making sure all cases are handled correctly, and eventually extending to combine multiple, simple policy implementations that are easier to create. This technique supports the creation of complex digital ecosystem policies and their implementation while dividing the problem into smaller, more manageable parts.

7.3. Quality Assurance for Semantics and User Communities

Using the results of the semantic and user community change analysis techniques developed and described extensively in D4.4 Modelling Contextual Semantics, we are here defining a quality assurance methodology that aims to address evolution in semantics and user communities.

When a change in a domain ontology of significance or UC, the ontology or community drift observatory will record it in the ecosystem model. This will in turn trigger policy checking and automated reporting. This allows alerting users by a tool like Somoclu (reported in D4.3 Content Semantics and Use Context Analysis Techniques, D.4.4 and D4.5 Context-Aware Content Interpretation) when there is a significant, potentially problematic drift in the semantics or user communities, that will then perform a manual analysis to assess and react to the issue.

The intellectual backdrop against which we measure our contribution is [Schlieder, 2010]. In that paper, he describes three types of significant changes affecting LTDP. Type 1 concerns hardware and software obsolescence and amounts to technology drift. His Type 2 is language change, eroding indexing terminology for advanced access in automated environments, with relevant experimental results on semantic drifts reported in D4.4. Schlieder’s Type 3 changes modify cultural value systems, e.g. making fashionable what used to be less accepted the day before. These changes can be modelled e.g. by drifts in UC perception because public appreciation of museum objects is important use-related metadata for access, influencing Type 1 efforts and providing the embedding context for Type 2 ones. At the same time, UC feedback based on artefact value perception from social media could be a new type of indicator for demand/consumption-driven LTDP for collection management.

7.3.1. Drift Threshold for semantic and UC quality assurance

QA can benefit from the interaction between semantic drift monitoring in a statistical environment, and ontology maintenance and development. Lists of drifting terms over periods can be thresholded and fed back to the ontology team for inspection and decision. On the other hand, the reverse process, feedback from ontology developers to human indexers or algorithms, results in indexing terminology consistency maintenance (ITCM), to remedy a long known problem from inter-indexer consistency studies (refs).

A rule, that can be implemented as an LRM dependency, will be activated on any drift delta reported to the model and determine if the drift threshold has been surpassed. When the value is above the threshold, the defined impact procedure will be called to react to the change. Given the nature of drift measurements, in most cases we expect that a human expert will be notified and will have to react and verify the entity of the drift, and possible, if necessary, correction measures.

7.3.2. Monitoring change in Semantics

Currently we can measure only the average term drift rate over a period, plus list the merging/splitting terms pointing at the concepts “behind” them. Manual inspection of these lists gives a first idea of conceptual dynamics (Wittek, Darányi 2014), but we are not able to interpret the distances and directions a term has travelled over a period. This is a *n*-body dynamics problem: given a high number of particles moving in all directions with different speed, drift detection and measurement for individual terms is granted, but it’s not possible for now to decide on a more general level what it means if e.g. in the Tate collection, the term “UK” moved north, closer to the term “nature”, in the epoch between 1796-1800. Therefore, the thresholded alert idea could address the improving/weakening semantic consistency of term clusters over a period as a QA measure only, i.e. how reliable is to use any current term agglomeration for the indexing of incoming documents from an ontology maintenance perspective.

7.3.3. Monitoring change in User Community

By analysing Twitter data, we have characterised the user community surrounding Tate [see PERICLES D4.4, 2016]. Being able to identify change in this community is important to preservation for assessing the social and cultural context of risk, in particular, it is important for the institution (as well as larger cultural and government agencies) to be able to monitor and manage who their audience is for access to the institution and its resources (Schlieder, 2010). Here we use social media for the monitoring of social context with a view to mitigating risk resulting from changes in this context. In the case of the Tate user community identified using social media data, this is largely self-selecting, and we therefore expect it to be fluid and dynamic; any changes are likely to evolve over time (cf. Vaughan, 2015; McCulloh and Carley, 2011). Based on this analysis, we can identify two primary forms of change: (1) the growth (or contraction) of the community (i.e., the properties of the social network); (2) the change in the concerns/interests (i.e., behaviour) of the community. To do this, we take two main approaches: first we examined the network structure of Tumblr over time, using network statistics in order to identify change; second, we explored the content of Tumblr, in order to identify broad topics under discussion and how these can help identify change. By combining both of these approaches, we are able to gain a better picture of the ‘who’ and ‘what’ of the Tate community as expressed via Tumblr. Given the open-ended nature of possibilities associated with such changes in user community, we expect human intervention to be required in response to the automatic identification of change (based on a particular difference threshold being met) and in the identification/validation of an appropriate threshold.

Here we summarise the methods before then presenting an overview of our findings (detailed descriptions of these are presented in the appendix 9): To analyse change in the social media user community around Tate, we harvest data from Tumblr. In contrast to Twitter data which only remains accessible for a limited period of time, Tumblr data access is not temporally limited, and so all historical material remains accessible (with the exception of content deleted or removed by the authors). This is therefore better suited to investigating potential user community changes over time. The Tumblr posts were previously collected for the study of social media content in this project (reported in D4.3 [PERICLES D4.3, 2016], which also gives details of the collection methods and subsequent processing). The analysis of Tumblr data in order to understand user community took two broad steps, the first was social network analysis to describe the network properties over time, and the second was topic modelling of the Tumblr post content (to understand user concerns, described using 5 and 15 topic solutions which provide different levels of granularity) which was also viewed over time to better understand changes.

Combining social network analysis metrics to the Tumblr network and topic modelling to the content of the posts of the Tate community on Tumblr, we have identified an example of change in this community activity relating to the growth in and around 2012: We note that both the 5 and 15 topic models identified this change in the content generated by the Tumblr community in relation to Tate; in particular, the adaptation of this social network and its content to meet its new needs. The 5 topic model identified a temporary change in focus from catalogue data to image data, and a greater focus on the Tate Modern and sharing exhibition information. Although the first two topic changes may indicate an exploration with new media, it is the focus on Tate Modern by the community and sharing/promotion of exhibitions which seems to indicate a more substantive shift in community usage of Tumblr.

For the 15 topic model, although many of the topics are used infrequently and which come and go in usage, in this analysis example we focused on five. From this example analysis, we found that following 2012 there was an increase in the popular describing and critiquing of art objects, along with a temporary focus on Tate Modern artists, and similarly less focus on images relating to exhibitions and performances at Tate Modern. Of these, we note that the change of focus relating to Tate Modern artists rather than exhibitions is interesting, and provides more detail to the general increase in posts relating to Tate Modern identified in the 5 topic model; in contrast, the increase in description and critique of art objects captured by the 15 topic model is only regarded as a temporary change in exploring the use of image descriptions in the 5 topic model. Regardless of these nuances, we view these broad changes as the increase in number of art appreciation posts (possibly by 'Art Lovers' as identified in the previous analysis of Twitter data [PERICLES D4.4, 2016]), as well as an increased interest in the community relating to Tate Modern. Both of these large scale changes of community behaviour are indicative of a social and cultural context, which we expect to be important in understanding the Tate in its broader online and offline community context.

Overall, the results from the two models show similar changes in the Tate Tumblr community (primarily the description of art objects and coverage of Tate Modern), but their different granularity and probabilistic generation mean that they provide detail in different ways, in some cases identifying increase of a topic, and in others the change in use from one topic to a similar one, but with nuanced differences. This would indicate therefore that at least for initial monitoring purposes, it would make sense to include the topics from both models in this process, thereby allowing the greatest insight into community change processes; the disadvantage to this is that there would be a slightly greater amount of data to consider, but in this case it does not seem to be too arduous, given that this would result in 20 topics in total. We note in relation to the 15 topic model, some of these topics occur with a relatively low frequency in the Tumblr data – this may lead to the possibility that such a model over fits the data, however, given that we propose the inclusion of the 5 topic model, then we expect this risk to be mitigated by the use of the broader topics, and greater coverage that this smaller model provides.

Summary

Here we have addressed an important aspect of change in relation to digital preservation, namely community change. This is important since the social and cultural context in which a cultural institution operates, determines how it can serve its community. In particular, we have addressed this question using social network data harvested from Tumblr relating to Tate over the period 2009-2015. In addition to exploring changes in social network relationships over this time, we have built a probabilistic model of the textual content of Tumblr posts using topic modelling. Comparing two models identifying different granularity of content over this 5 year period, we have been able to identify community changes in the use of Tumblr content in relation to an example of major network growth in 2012. This network growth resulted in different focuses by the Tumblr community, with these mainly related to long standing changed of an increase in the proportion of posts relating to Tate Modern and the popular critiquing and presenting to the community of Tate art objects; the different topic models each provided differently nuanced perspectives on these behaviours. Based on the findings of this example analysis, we propose how this could be incorporated into the automatic monitoring of community change for risk assessment. Here we detail two metrics and thresholds which could be used for community change monitoring, namely changes in network properties and changes in post content: we anticipate that the former will be used to identify large scale community changes using proposed thresholds based on the current change example analysis; then the relative changes in topic usage over time will be presented for assessment by a domain expert, since this evaluation will necessarily require human interpretation and knowledge. We also expect that human evaluation will be required in order to ensure that appropriate network change thresholds are being used. In relation to the example analysis of the Tate community Tumblr data, we note that the insights provided by such analysis, for example the growth in the network and the resulting increase in posts relating to art appreciation posts, as well as an increased interest relating to Tate Modern, both provide important indications of the wider social and cultural context of Tate, which is important in understanding the broader online and offline community context.

7.4. Prototypes for supporting change in technology, semantics and user communities

We here briefly sum up the results of task 5.3.2, the implementation of the theory and use cases we presented in Section 7. The concrete implementation details, for sake of brevity, are found in the appendix.

The first example is a Policy driven Digital Ecosystem inspired by CERN LHC data management, reported in Appendix 10. The cross task effort that illustrates the use of the policy model, the policy derivation guidelines, and the process based implementation of Quality Assurance. Further examples of QA and its use in connection to the DEM are illustrated in D3.5, together with policy driven modelling and the DEM policy model.

The second example, reported in Appendix 7, illustrates rule implementation of change management for single entities, based on the EUMETSAT example illustrated in section 7.2.4. This PoC serves to illustrate how change can be managed automatically in the DEM.

The third example addresses the use case of section 7.3.1, and is included in Appendix 8: it is the implementation of a rule to monitor and react to semantic and user community drift, making use of the Drift ontology. This rule will monitor drift recorded in the ontologies, and execute the appropriate actions.

These proof of concept show how the different techniques we propose for Quality Assurance, policy and change management can be implemented in some typical Digital Preservation scenarios, using standard rule language (SPIN) and PERICLES technologies. We believe that the approach we propose can offer an efficient compromise that allows to address, within the PERICLES model based approach, both situations where recurrent and simpler technical change can be addressed automatically (change management), and situations where complex, semantic change can only be reported and needs to be addressed manually (Semantic, UC change). Furthermore, we took into account cases where quality assurance and policies need to be implemented in an existing infrastructure, and we illustrate how this can be addressed with lightweight modelling of a complex situation and simple tests and QA method implementation that don't require a pre-defined infrastructure or a formal model (Appendix 10).

7.5. Conclusion

We have presented the latest developments in Policy, QA, and Change Management. We have reported the final model for policy and QA representation, implemented in the DEM, together with new guidelines for policy derivation, QA and compliance. Rule based change management for DE has been described with use-case examples representing policies and their implementation, supporting automated change management. We have also presented experiments of semantic and user community change observation and QA. Finally, we gave a short summary of the Proof of Concept implementation of the above-mentioned approaches (for sake of brevity, details are included in the appendices 6, 7, 8, 9 and 10). This contribution defines how policy, QA and change management can be automated for the different entities of the PERICLES ecosystem, for a quality assured policy implementation and for automated change management of Digital Ecosystems.

8. Approach to Appraisal

This section focuses on T5.4.1 Modelling of Appraisal Processes and defines the overall approach and methodology. The Technical Appraisal Tool itself, contained in Section 5 above corresponds to work in T5.4.2 Appraisal tools, which describes the implementation of these techniques in software tools that can be used by preservation practitioners.

We introduced our methodology in D5.2, outlined a technical approach and conducted some initial investigations and experiments. The primary focus since then has been on implementing and evaluating the individual components, and on development of a practical appraisal tool.

8.1. Objectives and definitions

We first briefly review and update the objectives and definitions from PERICLES Deliverable D5.2.

Appraisal is a process that in broad terms aims to determine which data should be kept by an organisation. This can include both decisions about accepting data for archival (e.g. acquisition) as well as determining whether existing archived data should be retained.

In traditional paper-based archival practice, appraisal is a largely manual process, which is performed by a skilled archivist or curator. Although archivists are often guided by organisational appraisal policies, such policies are mostly high-level and do not in themselves provide sufficiently detailed and rigorous criteria that can directly be translated into a machine executable form. Thus, much of the detailed decision-making rests with the knowledge and experience of the archivist.

With the increasing volumes of digital content in comparison to paper-based materials, manual appraisal is becoming increasingly impractical. Thus there is a need for automation based on clearly defined appraisal criteria. At the same time, decisions about acquisition and retention are dependent on many complex factors. Hence our aim here is to identify opportunities for automation or semi-automation of specific criteria that can assist and accelerate the human appraisal process.

To summarise, the main objectives of the task were:

1. To identify and define precisely a set of appraisal criteria whose evaluation is both relevant and can potentially be (partially or fully) automated.
2. To provide methods and associated tools that automate the evaluation of specific appraisal criteria.
3. To identify points in the content lifespan where appraisal (and reappraisal) is relevant and in particular, to demonstrate how appraisal is applied in changing environments.

In keeping with the overall PERICLES approach, the aim is to produce a focused set of tools running in a test-bed environment rather than an archive system. On the other hand, since the aim was to produce tools to support the user performing appraisal tasks, we deemed it essential to produce an application front end in which users could be presented with information to support decision-making, in order to prove the validity of our methodology.

Objective 1 was largely completed in D5.2, where we produced a comprehensive catalogue of appraisal criteria and outlined approaches to automation of certain criteria.

In this deliverable, we expand on objective 2, providing technical details on the methodology and implementation of tools to for the evaluation of the selected appraisal criteria.

We also provide an expanded discussion of objective 3, indicating how and where our tools can be applied, building on the continuum approach described in D5.2.

8.2. Narratives

In order to motivate the tool development, we defined two narratives for each case study domain that described the functionality from a user perspective.

8.2.1. Digital art and media

Narrative M1 in Table 2 describes the first step in performing a technical appraisal of a digital artwork collection, enabling a user to identify a specific artwork at risk for further detailed analysis.

Title	**M1:** Perform high-level technical appraisal of video-based artwork collection.
User	The target user is a media conservator who has to manage a large collection of digital video artworks. They are responsible for both assessing key risks to the digital video components of these artworks, to ensure they can be successfully displayed in the future, as well as performing preservation actions where necessary to mitigate for demonstrated risks.
Preconditions	We assume that the DVA ontology has been populated for a given collection. Each DVA has associated video files, players and metadata. In addition, it may also include system-level and operating system requirements, platform requirements and physical display specifications.
Step-by-step narrative	1. The **user** wishes to identify media components of artworks at risk in the collection. They start the **appraisal tool**. 2. The **appraisal tool** gathers data from external sources and computes a risk analysis of the whole collection. 3. The **appraisal tool** presents a list of risks and proximities to the **user**. This includes e.g. risks of obsolete formats, policy violations. The risks can be ranked in different ways (e.g. probability of occurrence, proximity, number of artworks affected). 4. The **user** selects a risk to analyse. The **appraisal tool** presents a visualisation of the magnitude of the risk on the collection. 5. The **user** selects to view the list of artworks affected by a particular risk. The appraisal tool provides a list of artworks ordered according to probability of occurrence, proximity and impact. 6. The **user** selects to analyse a specific artwork.

Table 2: Narrative for identifying digital video artworks at risk

Narrative M2 in Table 3 describes a typical set of steps for performing technical appraisal on an individual artwork. It can be run standalone or following scenario M1. We decided at an early stage in the design that analysis and preservation processes should only be run on individual artworks rather than in bulk.

Title	**M2:** Perform detailed technical appraisal and preservation actions for a given digital video artwork in the collection.
User	A media conservator who is responsible for maintaining the digital video artwork collection.
Preconditions	Same as narrative M1.
Step-by-step narrative	1. The **user** starts the appraisal tool and selects to view a specific digital video artwork, either from narrative M1, name search or browsing a list of artworks. 2. The **appraisal tool** provides a graphical view of the artwork and the risks and proximities are listed in a table below by the appraisal tool. These can be ranked in different ways (e.g. by name of artwork, risk proximity). 3. The **user** selects a particular risk to analyse and the **appraisal tool** presents a list of preservation actions and costs for the artwork. 4. The **user** selects one of the preservation actions to be performed. The **appraisal tool** launches the process compiler for building a workflow for execution.

Table 3: Narrative for risk-impact analysis of an individual artwork

Title	**S1:** Perform high-level technical appraisal of space science experiments.
User	The target user is a data manager who has to manage a collection of space science experiments. They are responsible for both assessing key risks to the experiments to identify barriers to reusing them in the future, as well as performing preservation actions where necessary to mitigate for identified risks.
Preconditions	We assume that an ontology has been built for each experiment and populated for a given collection.
Step-by-step narrative	1. The **user** wishes to identify experiments and components of experiments at risk in the collection. They start the **appraisal tool**. 2. The **appraisal tool** gathers data from external sources and computes a risk analysis of the whole collection. 3. The **appraisal tool** presents a list of risks and proximities to the **user**. This includes e.g. risks of obsolete formats, policy violations. The risks can be ranked in different ways (e.g. probability of occurrence, proximity, number of artworks affected). 4. The **user** selects a risk to analyse. The **appraisal tool** presents a

	visualisation of the magnitude of the risk on the collection. 5. The **user** selects to view the list of experiment instances affected by a particular risk. The appraisal tool provides a list of experiment instances ordered according to probability of occurrence, proximity and impact. 6. The **user** selects to analyse a specific experiment instance.

Table 4: Narrative for identifying experiment instances at risk

The link to the transformation execution is provided as a link in the appraisal tool UI, but not explicitly contained within the tool.

8.2.2. Space science

Narrative S1 in Table 4 describes the first step in performing a technical appraisal of a collection of space science experiments, enabling a user to identify a specific experiment instance at risk for further detailed analysis.

Narrative S2 is the analogue of M2 for space science, which considers the risk-impact analysis of an individual experiment instance.

Title	**S2:** Perform detailed technical appraisal and preservation actions for a given experiment instance in the collection.
User	Same as narrative S1.
Preconditions	Same as narrative S1.
Step-by-step narrative	1. The **user** starts the **appraisal tool** and selects to view a specific science experiment instance, either from narrative S1, name search or browsing a list of experiment instances. 2. The **appraisal tool** provides a graphical view of the experiment and the risks and proximities are listed in a table below by the appraisal tool. These can be ranked in different ways (e.g. by experiment identifier, risk proximity). 3. The **user** selects a particular risk to analyse and the **appraisal tool** presents a list of preservation actions and costs for the experiment. 4. The **user** selects one of the preservation actions to be performed. Where possible, the **appraisal too**l launches the process compiler to build a preservation process. In other cases, the **appraisal tool** will just provide a list of ranked transformation options for the **user** to implement offline.

Table 5: Narrative for risk-impact analysis of an individual science experiment instance

The link to the preservation process execution is contained within the technical appraisal tool, but hands over to other tools within the PERICLES framework.

8.3. Risk types

In order to understand and model the risks and mitigating actions to entities in a digital ecosystem, we produced a categorisation of ecosystem entities termed the component catalogue, described in more detail in D5.2. In order to model software-based and video-based artworks, we also consider hardware components.

The entities occurring in science and media case studies cover the broad types hardware, software, data and user community. We aimed to determine their specific risks that might occur and then to determine suitable data sources. Some sample data sources for a selection of entity types are shown in Table 6 below.

Entity category	**Entity description**	**Potential risks**	**Potential data sources**	**Mitigation**
Hardware	Any	Failure	Manufacturer data, survey data	Replacement (like-for-like) Migration (to different hardware)
		Obsolescence	Search engines	
Software	Operating system-commercial off-the-shelf (COTS)	Obsolescence	Search engines. Release frequencies. Sales of company. Size of user base.	Upgrade (major - new version, minor - upgrade of current version) Virtualisation - same OS, but running on a VM Migration (to a completely different OS)
	Operating system library (e.g. DirectX)	Obsolescence	Search engines. Release frequencies. Sales of company. Size of user base	Upgrade (to new version) Migration (to new library)
	Custom software application (executable only)	Obsolescence	None.	Emulation (rewriting software to have same functionality in a given programming language)
	Custom software application (source available)	Obsolescence	Search engines (for programming language) Software developer availability/cost	Upgrade (Modify existing software, using same language) Emulation (rewriting software to have same functionality in same or different programming language)

	Software application (open source community)	Obsolescence	Search engines. Software repositories (downloads, commits, releases).	Upgrade (major, minor) Migration (to a COTS application) Migration (to a different open source application) Emulation (by custom software)
	Software application (COTS)	Obsolescence	Search engines. Release frequencies. Sales of company. Size of user base.	Upgrade (major, minor) Migration (to a different COTS application) Migration (to an existing open source application) Emulation (by custom software having same functionality)

Table 6: Entity catalogue excerpt relating to risk estimation and data sources

In order to investigate these risks, we harvest data from a number of data sources with the aim of determining the magnitude and proximity of these risks. The most general purpose data sources we found were search engines, which provide data about a wide range of entities and are in many cases in sufficient volumes for meaningful statistical modelling. Beyond that, more specialised data sources are required, for example to examine activity on open source software projects. The evolution of open source development communities influences greatly the availability and obsolescence of such software.

End user community evolution can also be considered, from such information as user activity logs, provided that users can be registered and classified. More indirect measures can also be considered such as the obsolescence of programming languages, for example to maintain custom software applications with available source code, or the availability of software developers for that programming language.

The evolution of policies is, in general, more difficult to characterise using data driven methods. However, we can consider the evolution and obsolescence of publicly accessible standards using this approach, which can be interpreted as policies within the PERICLES ecosystem model.

As is pointed out in section 5.3 of (Falcao, 2010), it is proposed that obsolescence may in itself not be a primary risk, because it only affects the recovery of an artwork after it has failed. However, we may also consider for example that an artwork may need to be redisplayed on currently available PC hardware. Thus a conservator may consider the obsolescence of the envisaged PC platform as a primary risk. Hence any tool should enable the user to examine a range of potential risks.

8.4. Risk assessment workflow

In this section, we present an updated version of the risk assessment workflow, which is illustrated in Figure 24. The workflow aims to encapsulate a common process for performing technical appraisal across the two case study domains, and indicating the components involved.

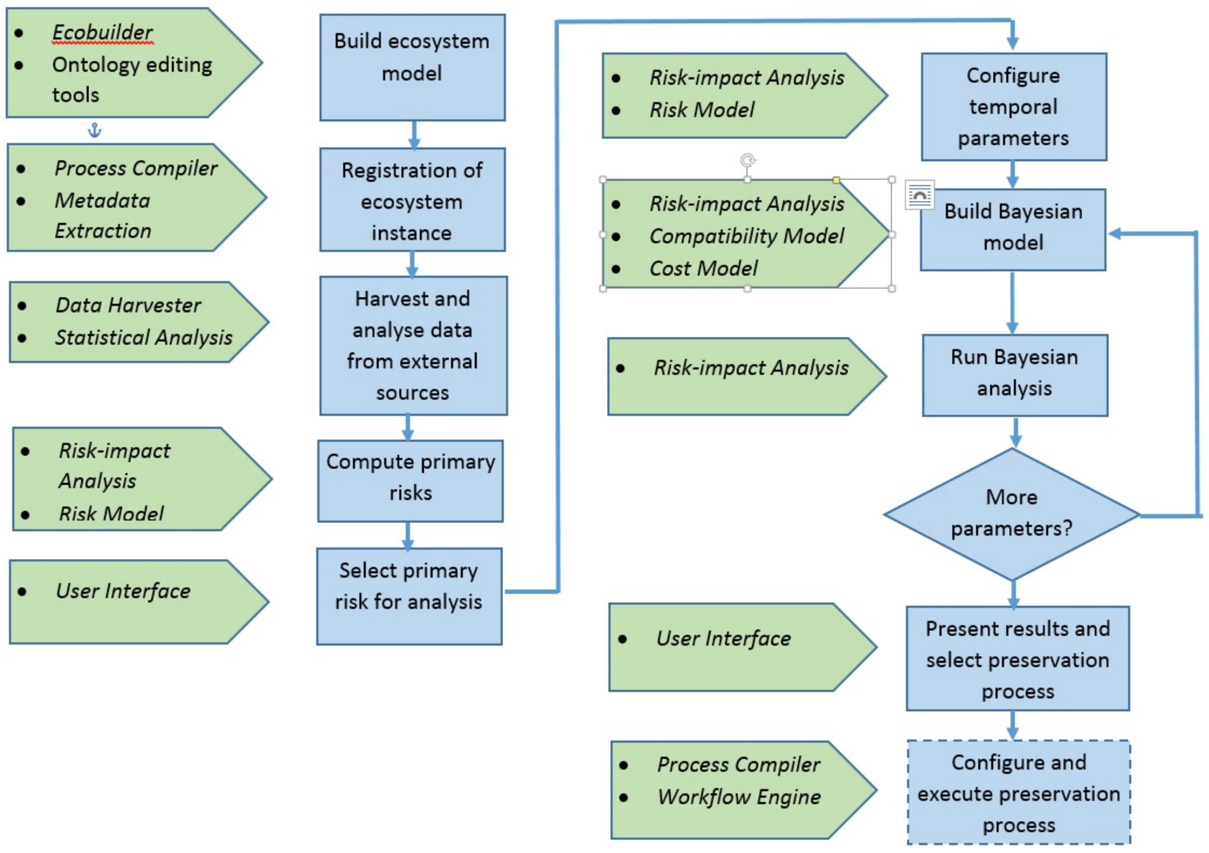


Figure 24: Workflow for risk-impact analysis of ecosystem model instances

The workflow is illustrated in blue and the components involved in green. The various models described in section 5.

Since the initial formulation in PERICLES Deliverable D5.2, we produce the risk mitigation for the primary and secondary risks in a single step. We have also modified the original approach so that we can set a time threshold, which represents a minimum sustainability period for a given object. We then step through a range of differ thresholds to produce multiple potential solutions with different costs and expected sustainability.

The final step in the workflow is the handover execute preservation processes on the PERICLES testbed, which will be described in D6.6.

8.5. Related work

The related work about technical appraisal and risk management can be found in appendix 11.

8.6. Data modelling

8.6.1. Modelling approach

The modelling approach we have taken is motivated by ideas from reliability engineering. Reliability engineering is a branch of systems engineering concerned with dependability in the lifecycle management of a product; the ability of a system or component to function under stated conditions for a specified period of time (IEEE, 1990). It is frequently used in industries such as aviation, which require complex systems such as aircraft to be maintained long beyond the lifetimes of individual components. The primary focus in reliability engineering is on hardware failure, and there has been much less focus on the long term sustainability of software components.

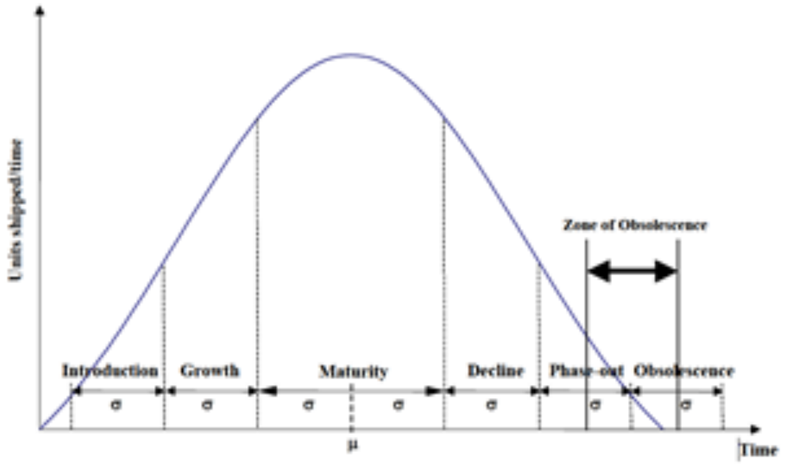


Figure 25: Standardised lifecycle model for a technology

Figure 25 represents a standardised lifecycle model for the units shipped against time. Here μ and σ represent the mean and standard deviation of the distribution function. The lifecycle is divided into a number of phases, termed Introduction, Growth, Maturity, Decline, Phase-out and Obsolescence. These phases are based largely on heuristics rather than rigorous analysis, but form an accepted benchmark supported by a wide range of software tools. To date, reliability engineering techniques appear to have been applied only to a very limited extent in digital preservation.

The Weibull distribution is widely used in reliability engineering to model failure rates of hardware components. The Weibull distribution is defined by the pdf function:

$$f(x) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

where k > 0 is the shape parameter and λ > 0 is the scale parameter.

We aimed to investigate the application of such techniques to model the lifecycle of digital components such as the raw data from Google Trends. The main objective was to estimate the distribution parameters, fitting error and associated confidence intervals. We would then be able to make assertions about the predicted obsolescence date of digital technologies as well as an associated confidence interval. For this we used statistical packages available in R.

8.6.2. External data sources

In this section we briefly describe some of the external data sources that we have used to extract and model risks associated with ecosystem entities. We discuss the approach and the findings from the analysis.

8.6.2.1. Google Trends

Google Trends[21] is a public website, based on Google search that shows how often a particular search term is entered relative to the total search volume across various regions of the world, and in various languages. Data are available since 2004.

Google provides raw trends data as downloadable CSV files. There is also a public API for accessing Google Trends data, which is what we make use of here. The Data Harvester performs authentication for a Google account and then downloads search activity data for a given set of terms. In using Google Trends, there is a risk of ambiguity of search terms, potentially resulting in misleading results. For example, a search on the term “windows” may return results for searches for replacement windows in buildings as well as on computer operating systems. Google Trends is however able to disambiguate search terms (e.g. Windows 8) is labelled as “Operating System”. Hence, by careful selection of search terms used, we were able to minimise this risk.

One advantage of using Google Trends as a data source is that there is data about a huge range of entities including not only digital information such as file formats and software applications, but also hardware such as connector types and displays, and user communities.

We modelled a large number of entity types using the Weibull fitting approach, filtering out those with a high level of interpolation error. The main issue encountered were lack of data, either because the entity being modelled had insufficient data points (e.g. the entity in question was introduced only in the past 2-3 years), or it was introduced a long period before 2004 (e.g. in the 1950s), and thus a large proportion of the trends data is unavailable.

8.6.2.2. GitHub

GitHub[22] is a platform is a web-based Git repository hosting service. Its main features are source code management and distributed version control. GitHub is widely used to host open source software development projects for personal users, communities and businesses, as well as enabling private repositories. As of April 2016, GitHub reported[23] having more than 14 million users and more than 35 million repositories.

For the purposes of PERICLES, GitHub is used to predict the obsolescence of open source software applications that are hosted on the site. In this case we used the number of individual commits per month on each software project. Compared to Google Trends, the volumes were lower, but correlated well.

8.6.2.3. SourceForge

SourceForge is a web-based service that offers software developers an online platform for

[21] https://www.google.co.uk/trends/
[22] https://github.com/
[23] https://github.com/about/press

management of free and open-source software projects. It provides a number of optional features, including a source code repository, bug tracking, mirroring of downloads for load balancing and a wiki for documentation.

As of March 2014[24], the SourceForge repository claimed to host more than 430,000 projects and had more than 3.7 million registered users. In the past several years, many users and project have now migrated to GitHub, other software hosting facilities, or self-host their software. In comparison, the SourceForge attracted at least 33 million visitors in August 2009. For Sourceforge we used downloads per month as the indicator of activity.

8.6.2.4. Stack Exchange

Stack Exchange[25] is a network of question and answer websites on topics in varied fields, modelled on Stack Overflow, a site for programming questions that was the original site in this network. Each site covers a specific topic, and questions, answers, and users are subject to a reputation award process. In order to extract an activity measure, we counted the monthly questions asked and successfully answered on a given topic.

8.6.2.5. Wikipedia

Wikipedia[26] is a free online encyclopedia, launched on 15th January, 2001, which allows its users to edit almost any article. Wikipedia is the largest and most popular general reference work on the Internet and is ranked among the ten most popular websites. Wikipedia consists of more than 40 million articles in more than 250 different languages and as of February 2014, had 18 billion page views and nearly 500 million unique visitors each month. We used the number of individual commits to a page as a measure of the interest in a given topic.

8.6.2.6. Discussion

We have uncovered a number of issues in modelling data from different sources. Availability of data over a suitably long timeframe is a major issue. For Google Trends, data is available only since 2004. Thus for technologies that have existed for a longer period time and have slow obsolescence rates, the predictions were less reliable. Conversely, for recently emerging technologies, there is often insufficient data available to make reliable longer term predictions. Since we are aiming at longer term predictions, neither of these is a huge issue.

Use of the tool is reliant on the availability of the data from public APIs such as Google Trends and SourceForge. All of these are outside the control of the designer of the tool. Any changes would require updates to the tool itself, either to the APIs used on in the Data Harvesting component in case the presentation of the data itself changes.

In many cases that we analysed, the Weibull distribution provided a good fit to the data. There were

[24] https://sourceforge.net/about

[25] http://stackexchange.com/

[26] https://en.wikipedia.org/wiki/Wikipedia

exceptions where we could not achieve a good fit with any choice of parameters. Examples were the various versions of the Android operating system. One simple strategy is to set an error threshold and reject any predictions that exceed this value. However, we are also investigating use of a wider range of distributions in the fitting. Further research is also needed to establish differences between the patterns of change and obsolescence between digital technologies (both open source and proprietary) and technologies which depend on manufacturing.

8.7. Ecosystem models

In this section, we outline the ecosystem model requirements for the technical appraisal tool. As far as possible, we have reused, adapted and in some cases extended the models developed in WP2.

We assume we initially start with an ecosystem containing a set of entities. Each entity has an associated set of metadata, together with a category from the five basic ecosystem types.

The technical appraisal tool requires several models as input.

- Compatibility model.
- Risk model.
- Cost model.

These are derived from ontologies created in WP2 together with heuristic methods. Each of the types of model is described in the sections below. Models are required to be constructed for each specific application area such as digital video artworks.

8.7.1. Compatibility model

The basis of the compatibility model is the ecosystem model that describes the entities and their dependencies in specific scenarios. In PERICLES WP2 the media case study, ontologies have been constructed for Digital Video Art and Software Based Art. For the purposes of developing and testing the technical appraisal tool for the media case study, we have mainly focussed on digital video playback.

The main extensions that we introduced are to extend the ontology with a larger number of instances to facilitate more realistic experiments. These are derived from publicly available sources. We also introduced weights into the model to reflect the degree of compatibility of entities linked by dependencies based on simple heuristics and background research.

8.7.2. Risk model

The risk model for each entity is derived from the data harvesting and analysis of external data sources. The risk model is updated periodically as new data are harvested. For example, datasets from Google are released on a monthly cycle. The processed data are stored in an RDF model.

8.7.3. Cost model

The cost model reflects the transformation cost of replacing one entity by another in the compatibility model. Thus the model is essentially a set of ordered pairs with an associated cost value. The cost model used is highly dependent on the application and context in which the appraisal tool is applied. Currently our cost model is static, but clearly the model would need to be updated periodically, both introduction of new entities as well as changes to underlying costs. Again this is stored as an RDF model in the ERMR.

8.8. Risk assessment

This section outlines how secondary risks can be assessed using belief propagation, based on mapping the ecosystem model instance to a Bayesian network model.

In order to compute secondary risks to complex digital objects, we treat the ecosystem instances E and associated compatibility model as a Bayesian network[27]. A Bayesian network is a probabilistic graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph.

Given a joint probability distribution p(X) for a vector X of states, we aim to find a configuration that attains the maximum value,

$$X^{\mathbf{max}} = argmax_X\ p(X) \tag{1}$$

or in other words satisfies

$$p(X^{max}) = max_{\mathbf{X}}\ p(X). \tag{2}$$

The maximum corresponds to the maximum over all allowed configurations X, which is different to the maximising the individual marginal distributions $p(X_i)$.

The inference algorithm uses a representation of the Bayesian network as a factor graph[28]. Factor graphs have two node types, namely variable nodes and factor nodes. The factor graph has a variable node for each node in the original graph, a factor node for each factor (i.e. dependency) and undirected links to each node variable in the factor. The max-product algorithm exploits the factorisation of (1) to efficiently compute the maximum. The algorithm was first proposed by Pearl (1988).

In the practical implementation, we obtained better results by using non-normalised weights rather than probabilities. The max-product algorithm can be efficiently implemented as message passing in the factor graph as follows. Two types of messages are passed, one from factors f to nodes X, denoted $\mu_{f\rightarrow X}$ and the other from factors to nodes, denoted $\mu_{X\rightarrow f}$. These are computed by the formulae

$$\mu_{f\rightarrow X}(X) = max_{X1,\ldots,Xm}\ (\ f(X,X_1,\ \ldots\ ,\ X_m)\ \Pi_{Y\in N(f)\backslash X}\ \mu_{Xi\rightarrow f}\ (Y)\ ) \tag{3}$$

and

$$\mu_{X\rightarrow f}(X) =\ \Pi_{g\in N(X)\backslash f}\ \mu_{g\rightarrow X}(X). \tag{4}$$

[27] https://en.wikipedia.org/wiki/Bayesian_network

[28] https://en.wikipedia.org/wiki/Factor_graph

Here N(f) is the set of neighbour variable nodes to the factor node f, and conversely N(X) represents the neighbour factor nodes to the variable node X. The messages are passed once from an arbitrarily chosen root node X_{root}. We can then recover the maximum value from

$$p(X^{max}) = \max_{Xroot} \left(\Pi_{g\in N(Xroot)} \mu_{g\to X}(X) \right) \qquad (5)$$

And the configuration attaining the maximum value by

$$X^{max} = \mathrm{argmax}_{Xroot} \left(\Pi_{f \text{ iin } N(X)} \mu_{f\to X}(X) \right) \qquad (6)$$

We obtain the maximum configuration values for the otter nodes by storing in each factor node the configuration attaining the maximum value in (3). Once we have the maximum value for the root node, we can compute the maxima for the other variable nodes recursively by backtracking through the factor graph.

In the practical implementation, we perform all operations in the log domain, due to potential rounding errors in computing with small values.

In order to apply the belief propagation approach, we assume the resulting factor graphs are acyclic. In some situations, there may be cyclic dependencies between nodes. Belief propagation may also be employed in this case, although the solutions may not be unique as in the acyclic case.

8.9. Conclusions

In Task T5.4 we have conducted both a study and classification of appraisal criteria in the context of evolving digital ecosystems, as well as developing methods and practical tools for specific criteria.

In D5.2, we produced a large catalogue of appraisal criteria, covering a wide variety of perspectives. Whilst some criteria can be relatively easily automated, many require both analytical techniques as well as extensive background knowledge. Our approach was therefore based on the principle of evaluating specific criteria and providing this information in an easily digestible form to a curator. Final appraisal decisions are based on combining multiple criteria. A typical appraisal policy for a museum or gallery may typically combine twenty or more criteria. Decisions may also be based on external factors such as available funding.

In the technical appraisal work, we have adopted a data-driven predictive approach to evaluating sustainability of complex digital objects. This extends previous approaches, which were based on reactive approaches, termed “technology watch”, and confined to relatively simple digital objects. In order to consider many of the risks that might impact digital objects, it is necessary to consider the entities and dependencies in the surrounding digital ecosystem which this work exploits. We also exploited a synergy between digital preservation and reliability engineering, a branch of systems engineering, which could potentially yield further interesting results for both areas.

Since an important part of the appraisal approach is informing expert curators, we deemed it essential to develop a user facing tool. Currently the tool is constrained by the background ontologies and data harvesting sources used. Extending these using the methods demonstrated in this work would greatly enhance the applicability of the tool. This could best be done as a community activity, supported by initiative such as ontology design patterns.

9. Conclusion

This deliverable has presented the developed tools and approaches for digital ecosystem management. An effective digital ecosystem management involves a representation of the ecosystem which can be expressed with the Digital Ecosystem Model. The model itself is part of D3.5, but the EcoBuilder tool has been described here, since it is a tool for ecosystem management. EcoBuilder enables the creation of digital ecosystem models for scenario experts and does not need ontology experts for the model instantiation.

Policies play an important role in ecosystem, they describe the behaviour of the entities and restrict them. Policies are a basis for processes that implement them. A generic policy model has been developed and represented here, it does not enforce executable policies, but an implementation can be given. The model can be integrated inside other models; an exemplary integration has been done with the Digital Ecosystem Model. Quality assurance ensures that policies are correctly applied and a model for this has been created as well. The policy model is complemented with prototypes for supporting change management by the use of rule engines. It has demonstrated how change can be addressed automatically and how complex change can be detected, but only reported. A guidance about implementing policies and quality assurance within existing infrastructure has been given.

The Policy Editor allows to edit predefined policies. Editing allows to fill in values for policies (e.g. retention time) and combine policies. The tool can work standalone or can be integrated into existing infrastructure to pull or output the policies. It is not bound to a certain policy structure; the templates can be high or low level policies.

Ecosystem models enable the analysis of the entities. Two strands of analysis have been covered here. Appraisal and Risk analysis use the entities from the model and include external data source for ranking. A theory for measuring risks has been developed and this theory has been implemented with the appraisal tool.

The other strand is user community change. An experiment with Tumblr data has been done. Posts about “Tate” have be analysed and the topics have been modelled around those activities. The aim was to detect shift in the community. This experiment has showed the grow and decline of topics which can then be used as trigger for notification to other models.

The last covered tool is the ERMR. It allows to store models and digital objects and provides retrieval interfaces. A definition of policies that operate on the data management level can be defined and it provides a messaging system. This allows to connect ERMR with external systems and invoke external processes. Since it is a generic tool for storing of models and digital objects, the output of all tools that were mentioned here can be kept on ERMR.

10. References

Akritidou M., et al. (2015) Deliverable 6.3 Specification of Architecture, Components and Design Characteristics. Available online at: http://pericles-project.eu/deliverables/14 (accessed on 26 September 2016).

Atkinson, R., & Flint, J. (2001). Accessing hidden and hard-to-reach populations: Snowball research strategies. *Social research update, 33(1), 1-4*.

Baxter, R. et al (2017). Deliverable 6.6 Language for change management. Will be available online at: http://pericles-project.eu/deliverables/80 (status: pending)

Becker, C., Kulovits, H., Rauber, A., & Hofman, H., et al. (2008, June). Plato: A service oriented decision support system for preservation planning. In Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries (pp. 367-370). ACM.

Bentley C. (2010). PRINCE2: A practical handbook. Routledge.

Biermann J., et al. (2014). Deliverable 5.1.1 Initial report on preservation ecosystem management. Available online at: http://pericles-project.eu/deliverables/12 (accessed on 26 September 2016).

Biermann J., et al. (2015). Deliverable 5.2 Basic tools for Digital Ecosystem management. Available online at: http://pericles-project.eu/deliverables/53 (accessed on 26 September 2016).

Biernacki, P., & Waldorf, D. (1981). Snowball sampling: Problems and techniques of chain referral sampling. *Sociological methods & research, 10(2)*, 141-163.

Blei, D. (2012). Topic modeling and digital humanities. Journal of Digital Humanities 2 (1),8-11.

Brokerhof A. W., Bülow A. E. (2016). The QuiskScan—a quick risk scan to identify value and hazards in a collection, Journal of the Institute of Conservation, 39:1, 18-28, DOI: 10.1080/19455224.2016.1152280.

Chanod, J.-P., Lagos, N., Vion-Dury, J.-Y. (2014). Deliverable 3.2 Linked Resource Model. Available online at: http://pericles-project.eu/deliverables/62 (accessed on 26 September 2016).

Corubolo, F. et al. (2014). Deliverable 4.1 Initial version of environment information extraction tools. Available online at: http://pericles-project.eu/deliverables/17 (accessed on 26 September 2016).

Dergiades, T., Milas, C., & Panagiotidis, T. (2014). Tweets, Google trends, and sovereign spreads in the GIIPS. Oxford Economic Papers, gpu046.

Graf R., Gordea S. (2013). A Risk Analysis of File Formats for Preservation Planning. Proceedings iPRES 2013; ed. José Borbinha, Michael Nelson, Steve Knight. http://purl.pt/24107.

Grant, A., et al. (2015). “Final version of integration framework and API implementation”, PERICLES Deliverable D6.4. Available online at http://pericles-project.eu/deliverables/54 (accessed on 26 September 2016).

Harvey R. (2007). DCC Digital Curation Manual: Instalment on Appraisal and Selection. http://www.dcc.ac.uk/resources/curation-reference-manual/completed-chapters/appraisal-and-selection.

Institute of Electrical and Electronics Engineers (1990). IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York, NY ISBN 1-55937-079-3.

ISO 16363 (2012), Space data and information transfer systems -- Audit and certification of trustworthy digital repositories. International standard. http://www.iso.org/iso/catalogue_detail.htm?csnumber=56510

ISO/IEC 31010 (2009), Risk management – Risk assessment techniques. International standard. http://www.iso.org/iso/catalogue_detail?csnumber=51073.

Kenney, A. R., McGovern, N. Y., Botticelli, P., Entlich, R., Lagoze, C., & Payette, Set al. (2002). Preservation risk management for web resources. INFORMATION MANAGEMENT JOURNAL-PRAIRIE VILLAGE-, 36(5), 52-61.

Lagos, N., Vion-Dury, J.-Y. (2015). Deliverable 3.3 Semantics for change management. Available online at: http://pericles-project.eu/deliverables/47 (accessed on 26 September 2016).

Lagos, N., Vion-Dury, J.-Y. (2017). Deliverable 3.4 Language for change management. Will be available online at: http://pericles-project.eu/deliverables/74 (status: pending)

Lawrence, G. W., Kehoe, W. R., Rieger, O. Y., Walters, W. H., & Kenney, A. R., et al. (2000). Risk Management of Digital Information: A File Format Investigation. Council on Library and Information Resources, 1755 Massachusetts Avenue, NW, Suite 500, Washington, DC 20036.

Library of Congress (2016). Sustainability of Digital Formats: Planning for Library of Congress Collections. Digital Formats http://www.digitalpreservation.gov/formats/sustain/sustain.shtml.

McDonald, S. (2000). Environmental Determinants of Lexical Processing Effort – Unpublished Ph.D. thesis, University of Edinburgh.

McCulloh, I., & Carley, K. M. (2011). *Detecting change in longitudinal social networks*. MILITARY ACADEMY WEST POINT NY NETWORK SCIENCE CENTER (NSC).

McCulloh, I., & Carley, K. M. (2011). *Detecting change in longitudinal social networks*. MILITARY ACADEMY WEST POINT NY NETWORK SCIENCE CENTER (NSC).

McHugh, A., Innocenti, P., Ross, S. (2008). "Assessing risks to digital cultural heritage with DRAMBORA". International Documentation Committee of the International Council of Museums (CIDOC) 2008, Athens, Greece, 15–18 September 2008.

Mimno et al. (2014). Care and feeding of topic models: Problems, diagnostics, and improvements. *Handbook of Mixed Membership Models and Their Applications, 3-34*.

Pearl J. (1988). Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference (2nd ed.). San Francisco, CA: Morgan Kaufmann. ISBN 1-55860-479-0.

Schlieder, C. (2010). Digital heritage: Semantic challenges of long-term preservation. Semantic Web 1.1, 2: 143-147.

Scholte T., Wharton, G. (2011). Installation Art Subjected to Risk Assessment – 91 Jeffrey Shaw's Revolution as Case Study, Agnes W. Brokerhof, Tatja Scholte, Bart Ankersmit, Gaby Wijers, Simone Vermaat in Inside Installation etc. http://cultureelerfgoed.nl/sites/default/files/publications/inside-installations-theory-and-practice-in-the-care-of-complex-artworks.pdf.

Stamatelatos, M. (2000). Probabilistic Risk Assessment: What Is It And Why Is It Worth Performing It?. NASA Office of Safety and Mission Assurance, 4(05), 00.

Stavropoulos et al., 2016. Deliverable 4.4 Modelling Contextualised Semantics. Available online at: http://pericles-project.eu/deliverables/90 (accessed on 26 September 2016).

TRAC (2007) Trustworthy Repositories Audit & Certification (TRAC): Criteria and Checklist. (Chicago, IL: Centre for Research Libraries; Dublin, OH: OCLC Online Computer Library Centre). http://www.crl.edu/PDF/trac.pdf.

Vermaaten, S., Lavoie, B., & Caplan, P. (2012). Identifying threats to successful digital preservation: The SPOT model for risk assessment. *D-Lib Magazine*, *18*(9), 4.

Wang, et al. (2011). Random field topic model for semantic region analysis in crowded scenes from tracklets. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on* (pp. 3441-3448). IEEE.

Wang, Y-C, Burke, M. and Kraut, R. (2013) Gender, topic, and audience response: an analysis of user -generated content on Facebook. In Mackay, W.E., Brewster, S., and Bødker, S. (Eds.). *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)* (pp. 31-34) New York, NY: ACM.

Wittek, P., Darányi, S., & Liu, Y. H. (2014, June). A vector field approach to lexical semantics. In *International Symposium on Quantum Interaction* (pp. 78-89). Springer International Publishing.

Zheng L. (2011). Knowledge Representation and Decision Support for Managing Product Obsolescence. PhD thesis Virginia Polytechnic Institute and State University. https://theses.lib.vt.edu/theses/available/etd-12192011-195720/unrestricted/Zheng_L_D_2011.pdf.

APPENDICES

1. Appendix: ERMR GUI Screenshots

Screenshots showing the graphical view of the object store and triple store are shown in Figure 26 and Figure 27.

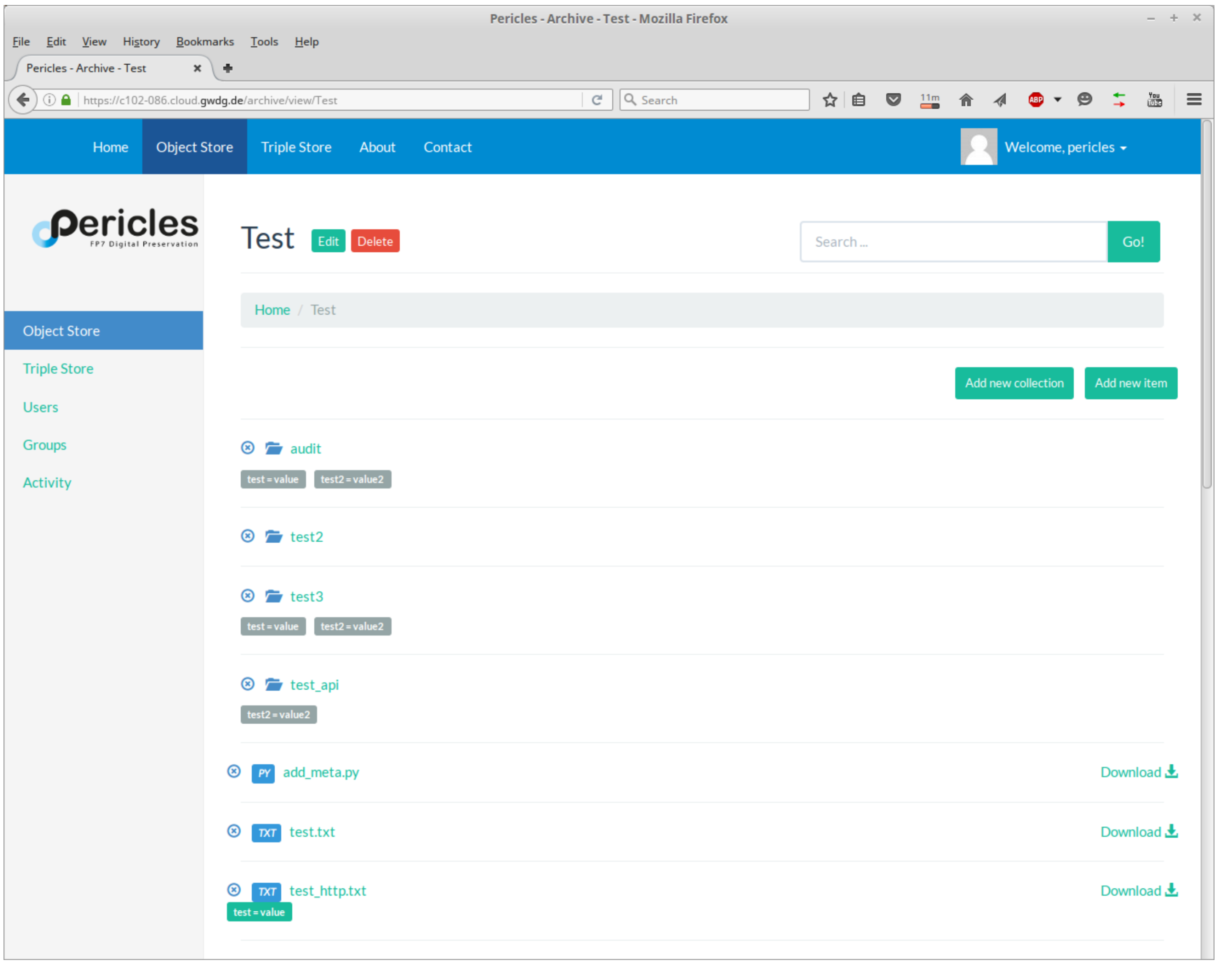
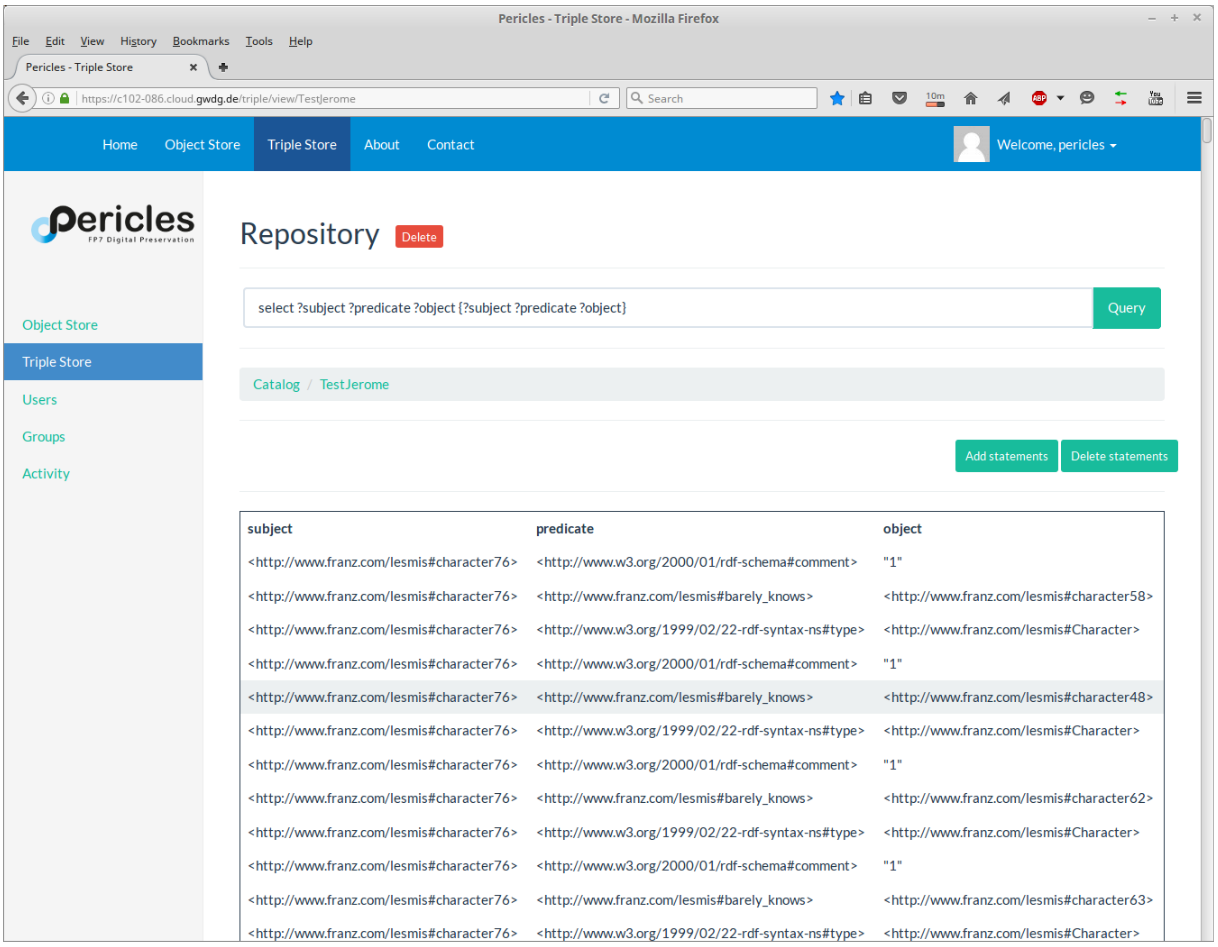


Figure 26: The object store (screenshot)


Pericles - Triple Store - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Pericles - Triple Store
https://c102-086.cloud.gwdg.de/triple/view/TestJerome
Search
Home Object Store Triple Store About Contact
Welcome, pericles
Pericles FP7 Digital Preservation
Object Store
Triple Store
Users
Groups
Activity
Delete
Query
Catalog / TestJerome
Add statements
Delete statements

Repository

select ?subject ?predicate ?object {?subject ?predicate ?object}

subject	predicate	object
<http://www.franz.com/lesmis#character76>	<http://www.w3.org/2000/01/rdf-schema#comment>	"1"
<http://www.franz.com/lesmis#character76>	<http://www.franz.com/lesmis#barely_knows>	<http://www.franz.com/lesmis#character58>
<http://www.franz.com/lesmis#character76>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://www.franz.com/lesmis#Character>
<http://www.franz.com/lesmis#character76>	<http://www.w3.org/2000/01/rdf-schema#comment>	"1"
<http://www.franz.com/lesmis#character76>	<http://www.franz.com/lesmis#barely_knows>	<http://www.franz.com/lesmis#character48>
<http://www.franz.com/lesmis#character76>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://www.franz.com/lesmis#Character>
<http://www.franz.com/lesmis#character76>	<http://www.w3.org/2000/01/rdf-schema#comment>	"1"
<http://www.franz.com/lesmis#character76>	<http://www.franz.com/lesmis#barely_knows>	<http://www.franz.com/lesmis#character62>
<http://www.franz.com/lesmis#character76>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://www.franz.com/lesmis#Character>
<http://www.franz.com/lesmis#character76>	<http://www.w3.org/2000/01/rdf-schema#comment>	"1"
<http://www.franz.com/lesmis#character76>	<http://www.franz.com/lesmis#barely_knows>	<http://www.franz.com/lesmis#character63>
<http://www.franz.com/lesmis#character76>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://www.franz.com/lesmis#Character>

Figure 27: The triple store (screenshot)

2. Appendix: ERMR API

2.1. Object Store REST API

2.1.1. Cloud Data Management Interface

The object store can be used to organize collections and digital objects in the store. It implements the Cloud Data Management Interface (CDMI) that defines the functional interface that applications may use to create, retrieve, update and delete data elements from the Object Store. In addition, metadata can be set on collections and their contained data elements through this interface.
The root URI path for the PERICLES entity registry demonstrator is https://141.5.100.67 and the CDMI web service is accessible at **/api/cdmi**.

2.1.2. Collections

2.1.2.1. Create a collection using HTTP

Synopsys

To create a new collection object, the following request shall be performed:

```
PUT <root URI>/api/cdmi/<CollectionName>/<NewCollectionName>/
```

where:

- <root URI> is the path to the registry.
- <CollectionName> is zero or more intermediate collection that already exist, with one '/' between each pair of collection names.
- <NewCollectionName> is the name for the collection to be created.

Response Status

HTTP Status	Description
201 Created	The new collection was created
401 Unauthorized	The authentication credentials are missing or invalid.
403 Forbidden	The client lacks the proper authorization
404 Not Found	The resource was not found at the specified URI

Example

PUT to the collection URI to create a collection:
```
PUT /api/cdmi/MyCollection/ HTTP/1.1
Host: 141.5.100.67
```

Response:
```
HTTP/1.1 201 Created
```

2.1.2.2. Create a collection using CDMI

Synopsys

To create a new collection object, the following request shall be performed:

```
PUT <root URI>/api/cdmi/<CollectionName>/<NewCollectionName>/
```

Where:

- <root URI> is the path to the registry.
- <CollectionName> is zero or more intermediate collection that already exist, with one slash (i.e., "/") between each pair of collection names.
- <NewCollectionName> is the name specified for the collection to be created.

Request Headers

Header	Type	Description	Requirement
Accept	Header String	“application/cdmi-container”	Optional
Content-Type	Header String	“application/cdmi-container”	Mandatory
X-CDMI-Specification-Version	Header String	“1.1”	Mandatory

Request Body

Field Name	Type	Description	Requirement
metadata	JSON Object	Metadata for the collection object	Optional

Response Headers

Header	Type	Description	Requirement
Content-Type	Header String	“application/cdmi-container”	Mandatory
X-CDMI-Specification-Version	Header String	“1.1”	Mandatory

Response Message Body

Field Name	Type	Description	Requirement
objectType	JSON String	"application/cdmi-container"	Mandatory
objectID	JSON String	ObjectID of the object	Mandatory
objectName	JSON String	Name of the object	Mandatory
parentURI	JSON String	URI for the parent object	Mandatory
parentID	JSON String	Object ID of the parent object	Mandatory
metadata	JSON Object	Metadata for the object	Mandatory

Response Status

HTTP Status	Description
201 Created	The new collection was created
400 Bad Request	The request contains invalid parameters.
401 Unauthorized	The authentication credentials are missing or invalid.
403 Forbidden	The client lacks the proper authorization
404 Not Found	The resource was not found at the specified URI

Example

PUT to the URI the collection object name and metadata:
```
PUT /api/cdmi/MyCollection/ HTTP/1.1
Host: 141.5.100.67
Accept: application/cdmi-container
Content-Type: application/cdmi-container
X-CDMI-Specification-Version: 1.1

{
   "metadata": {}
}
```

Response:

```
HTTP/1.1 201 Created
Content-Type: application/cdmi-container
X-CDMI-Specification-Version: 1.1

{
   "objectType" : "application/cdmi-container",
   "objectID" : "00007ED900104E1D14771DC67C27BF8B",
   "objectName" : "MyCollection/",
   "parentURI" : "/",
   "parentID" : "00007E7F0010128E42D87EE34F5A6560",
   "metadata" : {
                 ...
                },
}
```

2.1.2.3. Delete a collection using HTTP

Synopsys

To delete an existing container object, including all contained children, the following request shall be performed:

```
DELETE <root URI>/api/cdmi/<CollectionName>/<TheCollectionName>/
```

Where:

- <root URI> is the path to the registry.
- <CollectionName> is zero or more intermediate collection objects.
- <TheCollectionName> is the name of the collection object to be deleted.

Response Status

HTTP Status	Description
204 No Content	The collection was deleted
401 Unauthorized	The authentication credentials are missing or invalid.
403 Forbidden	The client lacks the proper authorization
404 Not Found	The resource was not found at the specified URI

Example

DELETE to the collection URI:

```
DELETE /api/cdmi/MyCollection/ HTTP/1.1
Host: 141.5.100.67
```

Response:
```
HTTP/1.1 204 No Content
```

2.1.2.4. Delete a collection using CDMI

Synopsys

To delete an existing container object, including all contained children, the following request shall be performed:

```
DELETE <root URI>/api/cdmi/<CollectionName>/<TheCollectionName>/
```

Where:
- <root URI> is the path to the registry.
- <CollectionName> is zero or more intermediate collection names.
- <TheCollectionName> is the name of the collection to be
- deleted.

Request Headers

Header	Type	Description	Requirement
X-CDMI-Specification-Version	Header String	“1.1”	Mandatory

Response Status

HTTP Status	Description
204 No Content	The collection was deleted
401 Unauthorized	The authentication credentials are missing or invalid.
403 Forbidden	The client lacks the proper authorization
404 Not Found	The resource was not found at the specified URI

Example

DELETE the collection at URI:
```
DELETE /api/cdmi/MyCollection/ HTTP/1.1
Host: 141.5.100.67
```

Response:
```
HTTP/1.1 204 No Content
```

2.1.2.5. Read a collection using CDMI

Synopsys

To read all fields from an existing collection object, the following request shall be performed:

```
GET <root URI>/api/cdmi/<CollectionName>/<TheCollectionName>/
```

Where:
- <root URI> is the path to the registry.
- <CollectionName> is zero or more intermediate collection objects.
- <TheCollectionName> is the name specified for the collection object to be read from.

Request Headers

Header	Type	Description	Requirement
Accept	Header String	“application/cdmi-container”	Optional
X-CDMI-Specification-Version	Header String	“1.1”	Mandatory

Response Headers

Header	Type	Description	Requirement
Content-Type	Header String	“application/cdmi-container”	Mandatory
X-CDMI-Specification-Version	Header String	“1.1”	Mandatory

Response Message Body

Field Name	Type	Description	Requirement
objectType	JSON String	“application/cdmi-container”	Mandatory
objectID	JSON String	ObjectID of the object	Mandatory
objectName	JSON String	Name of the object	Mandatory
parentURI	JSON String	URI for the parent object	Mandatory
parentID	JSON String	Object ID of the parent object	Mandatory
metadata	JSON Object	Metadata for the object	Mandatory
children	JSON Array of JSON Strings	Name of the children objects in the collection object.	Mandatory

Response Status

HTTP Status	Description
200 OK	The metadata for the collection is provided in the message body.
400 Bad Request	The request contains invalid parameters or field names.
401 Unauthorized	The authentication credentials are missing or invalid.
403 Forbidden	The client lacks the proper authorization
404 Not Found	The resource was not found at the specified URI

Example

GET to the collection object URI to read all the fields of the collection object:

```
GET /api/cdmi/MyCollection/ HTTP/1.1
Host: 141.5.100.67
Accept: application/cdmi-container
X-CDMI-Specification-Version: 1.1
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-container
X-CDMI-Specification-Version: 1.1

{
   "objectType" : "application/cdmi-container",
   "objectID" : "00007ED900104E1D14771DC67C27BF8B",
   "objectName" : "MyCollection/",
   "parentURI" : "/",
   "parentID" : "00007E7F0010128E42D87EE34F5A6560",
   "metadata" : {
                   ...
                },
   "children" : [
                  "child1",
                  "child2",
                  …
                ]
}
```

2.1.2.6. Update a collection using CDMI

Synopsys

To update some or all fields in an existing collection object, the following request shall be performed:

```
PUT <root URI>/api/cdmi/<CollectionName>/<TheCollectionName>/
```

To add, update, and remove specific metadata items of an existing collection object, the following request shall be performed:

```
PUT <root URI> /api/cdmi/ <CollectionName> / <TheCollectionName> /
?metadata:<metadataname>
```

Where:
<root URI> is the path to the registry.
<CollectionName> is zero or more intermediate collection objects.
<TheCollectionName> is the name of the collection object to be updated.

Request Headers

Header	Type	Description	Requirement
Accept	Header String	“application/cdmi-container”	Optional
X-CDMI-Specification-	Header String	“1.1”	Mandatory

Version			

Request Body

Field Name	Type	Description	Requirement
metadata	JSON Object	Metadata for the collection object	Optional

Response Status

HTTP Status	Description
204 No Content	The collection content was updated
400 Bad Request	The request contains invalid parameters.
401 Unauthorized	The authentication credentials are missing or invalid.
403 Forbidden	The client lacks the proper authorization
404 Not Found	The resource was not found at the specified URI

Example

PUT to the collection object URI to set new metadata :
```
PUT /api/cdmi/MyCollection/ HTTP/1.1
Host: 141.5.100.67
Content-Type: application/cdmi-container
X-CDMI-Specification-Version: 1.1


{
"metadata" : { ...
            }
}
```

Response:
```
HTTP/1.1 204 No Content
```

2.1.3. Data Objects

2.1.3.1. Create an object using HTTP

Synopsys

The following HTTP PUT creates a new data object at the specified URI:

```
PUT <root URI>/api/cdmi/<CollectionName>/<DataObjectName>
```

Where:

- <root URI> is the path to the CDMI cloud.
- <CollectionName> is zero or more intermediate collections that already exist, with one slash (i.e., "/") between each pair of collection names.
- <DataObjectName> is the name specified for the data object to be created

Request Headers

Header	Type	Description	Requirement
Content-Type	Header String	The content type of the data to be stored as a data object	Optional
Content-Range	Header String	A valid range-specifier	Optional

Request Body

The request message body contains the data to be stored.

Response Status

HTTP Status	Description
201 Created	The new data object was created
401 Unauthorized	The authentication credentials are missing or invalid
403 Forbidden	The client lacks the proper authorization
404 Not Found	The resource was not found at the specified URI

Example

PUT to the collection URI the data object name and contents:

```
PUT /api/cdmi/MyCollection/MyDataObject.txt HTTP/1.1
Host: 141.5.100.67
Content-Type: text/plain;charset=utf-8
Content-Length: 37

This is the Value of this Data Object
```

Response:
```
HTTP/1.1 201 Created
```

2.1.3.2. Create an object using CDMI

Synopsys

To create a new data object, the following request shall be performed:

```
PUT <root URI>/api/cdmi/<CollectionName>/<DataObjectName>
```

Where:
- <root URI> is the path to the registry.
- <CollectionName> is zero or more intermediate collections that already exist, with one slash (i.e., "/") between each pair of collection names.
- <DataObjectName> is the name specified for the data object to be created.

Request Headers

Header	Type	Description	Requirement
Accept	Header String	“application/cdmi-object”	Optional
Content-Type	Header String	“application/cdmi-object”	Mandatory
X-CDMI-Specification-Version	Header String	“1.1”	Mandatory

Request Message Body

Field Name	Type	Description	Requirement
mimetype	JSON String	Mime type of the data contained within the value field	Optional
metadata	JSON Object	Metadata for the data object	Optional
value	JSON String	The data object value	Optional

Response Headers

Header	Type	Description	Requirement
Content-Type	Header String	“application/cdmi-object”	Mandatory
X-CDMI-Specification-Version	Header String	“1.1”	Mandatory

Response Message Body

Field Name	Type	Description	Requirement
objectType	JSON String	“application/cdmi-object”	Mandatory
objectID	JSON String	ObjectID of the object	Mandatory
objectName	JSON String	Name of the object	Mandatory
parentURI	JSON String	URI for the parent object	Mandatory
parentID	JSON String	Object ID of the parent object	Mandatory
mimetype	JSON String	MIME type of the value of the data object	Mandatory
metadata	JSON Object	Metadata for the object	Mandatory

Response Status

HTTP Status	Description
201 Created	The new data object was created
400 Bad Request	The request contains invalid parameters.
401 Unauthorized	The authentication credentials are missing or invalid.
403 Forbidden	The client lacks the proper authorization
404 Not Found	The resource was not found at the specified URI

Example

PUT to the collection URI the data object name and contents:

```
PUT /api/cdmi/MyCollection/MyDataObject.txt HTTP/1.1
Host: 141.5.100.67
Accept: application/cdmi-object
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.1

{
   "mimetype" : "text/plain",
   "metadata" : { ...
   },
   "value" : "This is the Value of this Data Object"
}
```

Response:

```
HTTP/1.1 201 Created
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.1
{
   "objectType" : "application/cdmi-object",
   "objectID" : "00007ED90010D891022876A8DE0BC0FD",
   "objectName" : "MyDataObject.txt",
   "parentURI" : "/MyContainer/",
   "parentID" : "00007E7F00102E230ED82694DAA975D2",
   "mimetype" : "text/plain",
   "metadata" : {
       "cdmi_size" : "37"
   }
}
```

2.1.3.3. Delete an object using HTTP

Synopsys

The following HTTP DELETE deletes an existing data object at the specified URI:

```
DELETE <root URI>/api/cdmi/<CollectionName>/<DataObjectName>
```

Where:

- <root URI> is the path to the CDMI cloud.
- <CollectionName> is zero or more intermediate collections.
- <DataObjectName> is the name of the data object to be deleted.

Response Status

HTTP Status	Description
204 No Content	The data object was deleted
400 Bad Request	The request contains invalid parameters.
401 Unauthorized	The authentication credentials are missing or invalid.
403 Forbidden	The client lacks the proper authorization
404 Not Found	The resource was not found at the specified URI

Example

DELETE to the data object URI:
```
DELETE /api/cdmi/MyCollection/MyDataObject.txt HTTP/1.1
Host: 141.5.100.67
```

Response:
```
HTTP/1.1 204 No Content
```

2.1.3.4. Delete an object using CDMI

Synopsys

The following HTTP DELETE deletes an existing data object at the specified URI:

```
DELETE <root URI>/api/cdmi/<CollectionName>/<DataObjectName>
```

Where:
- <root URI> is the path to the CDMI cloud.
- <CollectionName> is zero or more intermediate collections.
- <DataObjectName> is the name of the data object to be deleted.

Request Headers

Header	Type	Description	Requirement
X-CDMI-Specification-Version	Header String	“1.1”	Mandatory

Response Status

HTTP Status	Description
204 No Content	The data object was deleted
400 Bad Request	The request contains invalid parameters.
401 Unauthorized	The authentication credentials are missing or invalid.
403 Forbidden	The client lacks the proper authorization
404 Not Found	The resource was not found at the specified URI

Example

DELETE the data object URI:
```
DELETE /api/cdmi/MyCollection/MyDataObject.txt HTTP/1.1
Host: 141.5.100.67
X-CDMI-Specification-Version: 1.1
```

Response:
```
HTTP/1.1 204 No Content
```

2.1.3.5. Read an object using HTTP

Synopsys

The following HTTP GET reads from an existing data object at the specified URI:

```
GET <root URI>/api/cdmi/<CollectionName>/<DataObjectName>
```

Where:
<root URI> is the path to the registry.
<CollectionName> is zero or more intermediate collections.
<DataObjectName> is the name of the data object to be read from

Request Headers

Header	Type	Description	Requirement
Range	Header String	A valid range specifier	Optional

Response Headers

Header	Type	Description	Requirement
Content-Type	Header String	The mimetype of the data object	Mandatory

Response Message Body

The response message body is the content of the data object.

Response Status

HTTP Status	Description
20O OK	The data object content was returned in the response.
401 Unauthorized	The authentication credentials are missing or invalid
403 Forbidden	The client lacks the proper authorization
404 Not Found	The resource was not found at the specified URI

Example 1

GET to the data object URI to read the value of the data object:
```
GET /api/cdmi/MyCollection/MyDataObject.txt HTTP/1.1
Host: 141.5.100.67
```

Response:
```
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 37

This is the Value of this Data Object
```

Example 2

GET to the data object URI to read the first 11 bytes of the value of the data object:
```
GET /api/cdmi/MyContainer/MyDataObject.txt HTTP/1.1
Host: 141.5.100.67
Range: bytes=0-10
```

Response:
```
HTTP/1.1 206 Partial Content
Content-Type: text/plain
Content-Range: bytes 0-10/37
Content-Length: 11
```

2.1.3.6. Read an object using CDMI

Synopsys

The following HTTP GET reads from an existing data object at the specified URI:

```
GET <root URI>/api/cdmi/<CollectionName>/<DataObjectName>
```

```
GET <root URI>/api/cdmi/<CollectionName>/<DataObjectName>
?value:<range>;...
```

```
GET <root URI>/api/cdmi/<CollectionName>/<DataObjectName>
```

```
?metadata:<prefix>;...
```

Where:

- <root URI> is the path to the CDMI cloud.
- <CollectionName> is zero or more intermediate collections.
- <DataObjectName> is the name of the data object to be read from.
- <range> is a byte range of the data object value to be returned in the value field.
- <prefix> is a matching prefix that returns all metadata items that start with the prefix value.

Request Headers

Header	Type	Description	Requirement
Accept	Header String	“application/cdmi-object”	Optional
X-CDMI-Specification-Version	Header String	“1.1”	Mandatory

Response Headers

Header	Type	Description	Requirement
Content-Type	Header String	“application/cdmi-object”	Mandatory
X-CDMI-Specification-Version	Header String	“1.1”	Mandatory

Response Message Body

Field Name	Type	Description	Requirement
objectType	JSON String	“application/cdmi-object”	Mandatory
objectID	JSON String	ObjectID of the object	Mandatory
objectName	JSON String	Name of the object	Mandatory
parentURI	JSON String	URI for the parent object	Mandatory

parentID	JSON String	Object ID of the parent object	Mandatory
mimetype	JSON String	MIME type of the value of the data object	Mandatory
metadata	JSON Object	Metadata for the object	Mandatory
value	JSON String	data object value	Conditional

Response Status

HTTP Status	Description
20O OK	The data object content was returned in the response.
401 Unauthorized	The authentication credentials are missing or invalid
403 Forbidden	The client lacks the proper authorization
404 Not Found	The resource was not found at the specified URI

Example

GET to the data object URI to read all fields of the data object:

```
GET /api/cdmi/MyCollection/MyDataObject.txt HTTP/1.1
Host: 141.5.100.67
Accept: application/cdmi-object
X-CDMI-Specification-Version: 1.1
```

Response:

```
HTTP/1.1 200 OK
X-CDMI-Specification-Version: 1.1
Content-Type: application/cdmi-object

{
   "objectType" : "application/cdmi-object",
   "objectID" : "00007ED90010D891022876A8DE0BC0FD",
   "objectName" : "MyDataObject.txt",
   "parentURI" : "/MyCollection/",
   "parentID" : "00007E7F00102E230ED82694DAA975D2",
   "mimetype" : "text/plain",
   "metadata" : {
       "cdmi_size" : "37"
   },
   "valuerange" : "0-36",
   "value" : "This is the Value of this Data Object"
}
```

2.1.3.7. Update an object using HTTP

Synopsys

The following HTTP PUT updates an existing data object at the specified URI:

```
PUT <root URI>/api/cdmi/<CollectionName>/<DataObjectName>
```

Where:
<root URI> is the path to the CDMI cloud.
<CollectionName> is zero or more intermediate collections.
<DataObjectName> is the name of the data object to be updated.

Request Headers

Header	Type	Description	Requirement
Content-Type	Header String	The mime type of the data to be stored as a data object	Optional
Content-Range	Header String	A valid range-specifier	Optional

Request Body

The request message body contains the data to be stored.

Response Status

HTTP Status	Description
204 No Content	The data object content was updated
401 Unauthorized	The authentication credentials are missing or invalid
403 Forbidden	The client lacks the proper authorization
404 Not Found	The resource was not found at the specified URI

Example

PUT to the data object URI to update the value of the data object:

```
PUT /api/cdmi/MyCollection/MyDataObject.txt HTTP/1.1
Host: 141.5.100.67
Content-Type: text/plain
Content-Length: 37
```

This is the value of this data object

Response:

```
HTTP/1.1 204 No Content
```

2.1.3.8. Update an object using CDMI

Synopsys

The following HTTP PUT updates an existing data object at the specified URI:

```
PUT <root URI>/api/cdmi/<CollectionName>/<DataObjectName>
```

```
PUT <root URI>/api/cdmi/<CollectionName>/<DataObjectName>
?value:<range>
```

```
PUT <root URI>/api/cdmi/<CollectionName>/<DataObjectName>
?metadata:<metadataname>
```

Where:

- <root URI> is the path to the registry.
- <CollectionName> is zero or more intermediate collections that already exist, with one slash (i.e., "/") between each pair of collection names.
- <DataObjectName> is the name specified for the data object to be created.
- <range> is a byte range for the data object value to be updated.

Request Headers

Header	Type	Description	Requirement
Content-Type	Header String	“application/cdmi-object”	Mandatory

X-CDMI-Specification-Version	Header String	“1.1”	Mandatory

Request Message Body

Field Name	Type	Description	Requirement
mimetype	JSON String	Mime type of the data contained within the value field	Optional
metadata	JSON Object	Metadata for the data object	Optional
value	JSON String	The data object value	Optional

Response Status

HTTP Status	Description
204 No Content	The data object was updated
400 Bad Request	The request contains invalid parameters.
401 Unauthorized	The authentication credentials are missing or invalid.
403 Forbidden	The client lacks the proper authorization
404 Not Found	The resource was not found at the specified URI

Example

PUT to the data object URI to set new field values:
```
PUT /api/cdmi/MyCollection/MyDataObject.txt HTTP/1.1
Host: 141.5.100.67
Accept: application/cdmi-object
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.1
{
   "mimetype" : "text/plain",
   "metadata" : {
        "colour": "red",
    },
    "value" : "This is the Value of this Data Object"
}
```

Response:

```
HTTP/1.1 204 No Content
```

2.1.4. Find REST API

Synopsys

We have defined a minimal find API accessible under /api/find. It can search in names or metadata of collections and data objects and returns matching objects.

The following HTTP GET query the registry:

```
GET <root URI>/api/find?findTerms=<term>
```

```
GET <root URI>/api/find?findTerms=<term>&where=<where>
```

Where:

- <root URI> is the path to the registry.
- <term> is the keyword we are looking for.
- <where> is the place we are looking, it can be “name”, “metadata” or “both”.The default value is “both”.

Response Message Body

Field Name	Type	Description	Requirement
result	JSON Array of JSON Strings	List of URI for the matching object.	Mandatory

Response Status

HTTP Status	Description
200 OK	The result are returned in the response body.
400 Bad Request	The request contains invalid parameters.
401 Unauthorized	The authentication credentials are missing or invalid.
403 Forbidden	The client lacks the proper authorization

Example

GET to the find URI to get matching objects:
```
GET /api/find?findTerms=test HTTP/1.1
Host: 141.5.100.67
```

Response:
```
HTTP/1.1 200 OK

{
 "result" : ["/MyContainer/",
             "/MyCollection/MyDataObject.txt",
           … ]
}
```

2.2. Triple Store REST API

The Pericles triple store API acts as a mediator between clients and the internal triple store we are using. It interprets the requests of the client and forward a request in the correct format for the triple store.
The root URI path for the PERICLES entity registry demonstrator is https://141.5.100.67 and the triple store web service is accessible at **/api/triple**.

2.2.1. List repositories

2.2.1.1. Synopsys

To list existing repositories, the following request shall be performed:

```
GET <root URI>/api/triple
```

where:

- <root URI> is the path to the registry.

2.2.1.2. Response Message Body

The response message contains a JSON list of repositories information (to be refined).

2.2.1.3. Response Status

HTTP Status	Description
200 OK	The list is returned

2.2.1.4. Example

GET a list of repositories:
```
GET /api/triple HTTP/1.1
Host: 141.5.100.67
```

Response:
```
HTTP/1.1 200 Ok


[ { "title": "DemoLondon", "writable": true, "id": "DemoLondon" },
 { "title": "Test", "writable": true, "id": "Test" } ]
```

2.2.2. Create a repository

2.2.2.1. Synopsys

To create a new repository, the following request shall be performed:

```
PUT <root URI>/api/triple/<NewRepositoryName>
```

where:

- <root URI> is the path to the registry.

- <NewRepositoryName> is the name for the repository to be created.

2.2.2.2. Response Status

HTTP Status	Description
201 Created	The new repository was created

2.2.2.3. Example

PUT to the triple store URI to create a repository:
```
PUT /api/triple/MyRepository HTTP/1.1
Host: 141.5.100.67
```

Response:
```
HTTP/1.1 201 Created
```

2.2.3. Delete a repository

2.2.3.1. Synopsys

To delete a repository, the following request shall be performed:

```
DELETE <root URI>/api/triple/<repositoryName>
```

where:
- <root URI> is the path to the registry.
- <repositoryName> is the name of the repository to be deleted.

2.2.3.2. Response Status

HTTP Status	Description

204 No Content	The repository was deleted

2.2.3.3. Example

DELETE a repository:
```
Delete /api/triple/MyRepository HTTP/1.1
Host: 141.5.100.67
```

Response:
```
HTTP/1.1 204 No Content
```

2.2.4. Add triples to a repository

2.2.4.1. Synopsys

The following HTTP PUT/POST requests add triples to a repository. A PUT request empty the repository first while a POST request add triples.

```
PUT <root URI>/api/triple/<repositoryName>/statements
POST <root URI>/api/triple/<repositoryName>/statements
```

Where:
- <root URI> is the path to the registry.
- <repositoryName> is the name of the repository.

2.2.4.2. Request Headers

Header	Type	Description	Requirement
Content-Type	Header String	The content type of the triples • “text/plain” for ntriples • “application/rdf+xml” for RDF • “text/turtle” for Turtle	Mandatory

2.2.4.3. Request Body

The request message body contains the data to be stored.

2.2.4.4. Response Status

HTTP Status	Description
201 Created	The triples were added

2.2.4.5. Example

PUT triples to the repository URI:

```
PUT /api/triple/MyRepository/statements HTTP/1.1
Host: 141.5.100.67
Content-Type: text/plain

<http://www.pericles.org/models#process1>
<http://www.pericles.org/models#name> "Ingest" .
<http://www.pericles.org/models#process1>
<http://www.pericles.org/models#description>  "Ingest  documents  in  the
registry" .
<http://www.pericles.org/models#process1>
<http://www.pericles.org/models#identity>      "7d4e14c8-1adf-4cfd-b0b8-
ede46944b006" .
<http://www.pericles.org/models#process1>
<http://www.pericles.org/models#version> "0.1" .
```

Response:
```
HTTP/1.1 201 Created
```

2.2.5. List triples of a repository

2.2.5.1. Synopsys

To list triples contained in an existing repository, the following request shall be performed:

```
GET <root URI>/api/triple/<repositoryName>
GET <root URI>/api/triple/<repositoryName>/statements
```

where:
- <root URI> is the path to the registry.
- <repositoryName> is the name of the repository to list.

2.2.5.2. Request Headers

Header	Type	Description	Requirement
Accept	Header String	“application/json” “application/rdf+xml”	Optional

2.2.5.3. Response Message Body

By default the response message contains a JSON list of triples. Specifying “application/rdf+xml” as the Accept header can be used to obtain the result in XML.

2.2.5.4. Response Status

HTTP Status	Description
200 OK	The list is returned

2.2.5.5. Example

GET a list of repository triples:
```
GET /api/triple/MyRepository HTTP/1.1
Host: 141.5.100.67
```

Response:
```
HTTP/1.1 200 Ok
```

```
[
        [
        "<http://www.pericles.org/models#process3>",
        "<http://www.pericles.org/models#version>",
        "\"1.0\""
        ],
        [
        "<http://www.pericles.org/models#process3>",
        "<http://www.pericles.org/models#identity>",
        "\"3ecdb028-40ec-453a-b4eb-21ad9234ac5e\""
        ],
        [
        "<http://www.pericles.org/models#process3>",
        "<http://www.pericles.org/models#description>",
        "\"Extract metadata in a document of the registry\""
        ],
        [
        "<http://www.pericles.org/models#process3>",
        "<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>",
        "<http://www.pericles.org/models#process>"
        ],
...
]
```

2.2.6. Delete all triples of a repository

2.2.6.1. Synopsys

To delete all triples of a repository, the following request shall be performed:

```
DELETE <root URI>/api/triple/<repositoryName>/statements
```

where:

- <root URI> is the path to the registry.
- <repositoryName> is the name of the repository that contains triples to be deleted.

2.2.6.2. Response Status

HTTP Status	Description
204 No Content	The triples were deleted

2.2.6.3. Example

DELETE a repository:
```
Delete /api/triple/MyRepository/statements HTTP/1.1
Host: 141.5.100.67
```

Response:
```
HTTP/1.1 204 No Content
```

2.2.7. Query a repository

2.2.7.1. Synopsys

To send a SPARQL query to a repository, the following request shall be performed:

```
GET <root URI>/api/triple/<repositoryName>?query=<SparqlQuery>
```

where:
- <root URI> is the path to the registry.
- <repositoryName> is the name of the repository to query.
- <SparqlQuery> is a sparql query encoded as a URI.

2.2.7.2. Request Headers

Header	Type	Description	Requirement
Accept	Header String	“application/json” “application/sparql-results+xml” “application/sparql-results+json”	Optional

2.2.7.3. Response Message Body

By default the response message contains a JSON dictionary (“application/json”):
- “values stores a list of tuples
- “name” stores a list of names for the returned tuples

Specifying “application/sparql-results+xml” or “application/sparql-results+json” can be used to obtain these output :

https://www.w3.org/TR/rdf-sparql-XMLres/

https://www.w3.org/2001/sw/DataAccess/json-sparql/

2.2.7.4. Response Status

HTTP Status	Description
200 OK	The result is returned

2.2.7.5. Example

GET to evaluate a SPARQL query on a repository:

```
GET
/api/triple/MyRepo?query=select%20?s%20?p%20?o%20%7B?s%20?p%20?o%7D
HTTP/1.1
Host: 141.5.100.67
```

Response:

```
HTTP/1.1 200 Ok


{
      "values": [
      [
            "<http://www.pericles.org/models#version>",
            "\"1.0\""
      ],
      [
            "<http://www.pericles.org/models#identity>",
            "\"3ecdb028-40ec-453a-b4eb-21ad9234ac5e\""
      ],
      [
            "<http://www.pericles.org/models#name>",
            "\"Convert\""
      ],
      ...
      ],
      "names": [
      "p",
      "o"
      ]
}
```

3. Appendix: Rule Execution Metadata in the ERMR

As mentioned in Section 3, the ERMR Listener can run Python scripts stored in a special collection of the object store. We can add a specific metadata to the script to define the rule execution condition for this script. This corresponds to the topic of the MQTT protocol. The format is "OPERATION/OBJECT/PATH" where:

- OPERATION = create, delete, update
- OBJECT = resource, collection, repository or + for anything
- PATH = collection path, repository name with # that can match a subtree

For instance, the metadata "topic" = "create/+/path/to/watch/#" will execute the associated script each time something is created in the collection "/path/to/watch".

4. Appendix: Deployment Options for the Policy Editor

The diagrams below illustrate various different integration configurations. In the second diagram (Figure 28), a single adapter offers access to both the Policy Editor and the Ecosystem. Policies will then be persisted as an Ecosystem entity. Another diagram (Figure 29) shows a configuration wherein the Policy Editor can query an Ecosystem, but policies are persisted separately to a dedicated storage component. There is no support for executing processes. The bottom diagram (Figure 30) shows a minimal stand-alone configuration. The Policy Editor can persist and load policies via a file-based storage. Neither Ecosystem nor Process Execution Engine are available. In practice, the choice of the integration configuration depends primarily on the intended usage of the PE in an application and of what is technically possible and cost-effective. Figure 31 displays a possible configuration that could integrate with the ERMR and Process Model Compiler of PERICLES.

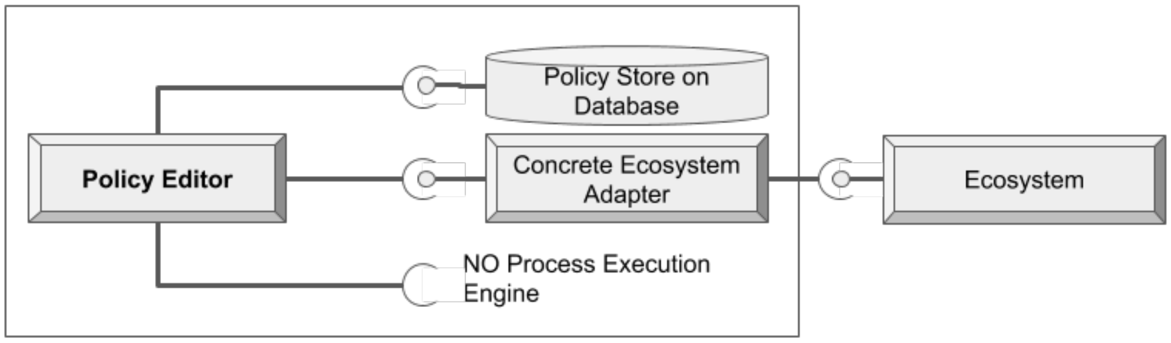


Figure 28: Policy Editor Deployment: database policy store and ecosystem

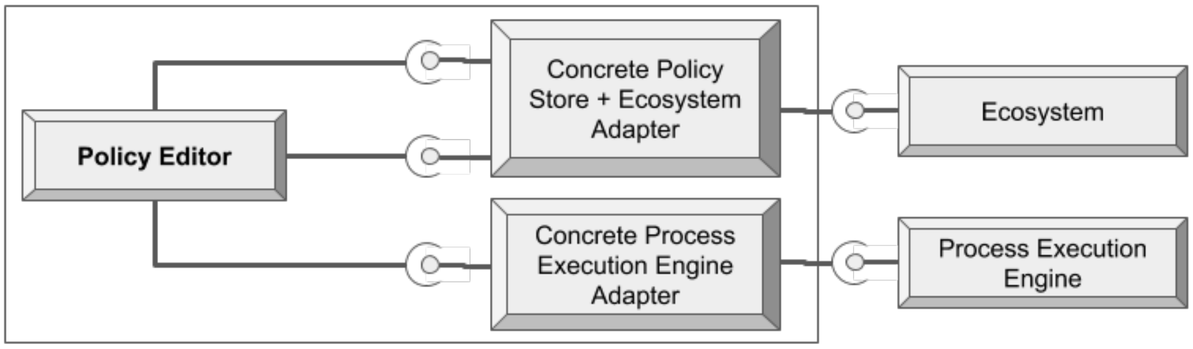

Policy Editor
Concrete Policy Store + Ecosystem Adapter
Concrete Process Execution Engine Adapter
Ecosystem
Process Execution Engine

Figure 29: Policy Editor Deployment: database policy store, digital ecosystem and process execution engine

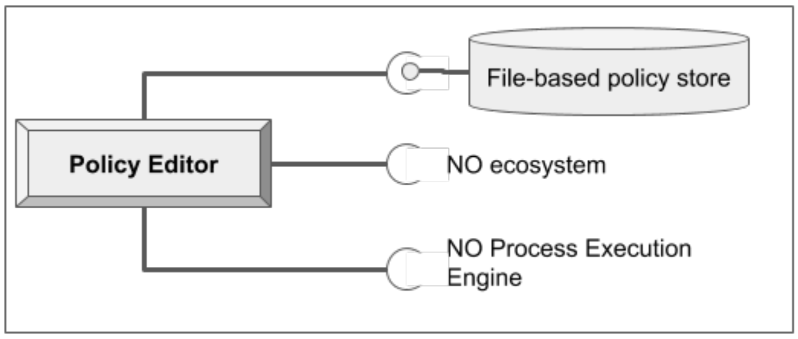


Figure 30: Policy Editor Deployment: only a file-based policy storage

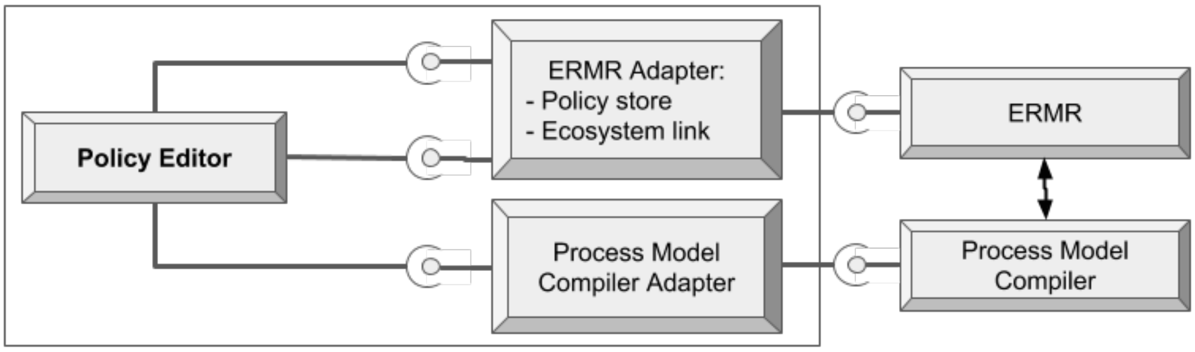


Figure 31: Policy Editor integration within PERICLES infrastructure

On the one extreme end, the PE can be used as a fully standalone tool without integration to any other component. This is a very lightweight approach that allows a policy creator to quickly create a (printed) set of policies without the overhead and costs incurred of integrating the PE into a bigger system. Of course the drawback here is that automatic validation of policies cannot be done nor will the policy creator be helped by an ecosystem model for filling in the blanks in the policies. A typical real-life example where this configuration could be used would be by (upper) management in an organization, creating high-level organization-wide policies. The other extreme configuration entails full integration in a preservation infrastructure: policies become an inherent part of the ecosystem, are defined using that same ecosystem and can automatically be validated on the ecosystem if there is an appropriate policy validator component such as the PERICLES Process Compiler available.

Typically, there is a significant cost associated to this level of integration though as custom adapter components are to be developed that are the interfaces between the PE and the various components. Therefore, this kind of configuration makes most sense in a preservation environment that is relatively stable, mature and large enough to warrant the cost.

5. Appendix: Policy Template Files

This appendix describes the content of policy template files in more detail. A policy template file contains blueprints for policies and includes:

- properties with content that may or may not contain variables. These variables replace the values of the properties of the applicable policy model, in which the properties get an empty value by default. In other words, policy templates give a parametrized default value to one or more of the model properties. In case a policy template refers to properties that are not mentioned in the policy model, which is not an unlikely scenario as policy templates can originate from various sources while the PE will use a single policy model, a translation wizard will guide the policy creator in resolving these discrepancies by allowing him to specify the mapping between policy template properties and policy model properties.
- specification of variables:
    - variables can be local to a policy or global to the whole policy set. In the former case, using the PE to change the value of the variable will only have an effect on the policy that the variable belongs to. In the latter case, the values of variables (identified by their name) are propagated to all other locations where that variable is used.
    - as the same policy template can be instantiated multiple times, global variables that are defined in it will be used in each policy instance. This is typically not the desired behaviour. To deal with this, variable names can be annotated with a suffix that indicates that each instantiation of the policy will create a new global variable. This behaviour can be overridden by the policy creator who, instead of automatically creating a new variable name, can select an existing variable name for the new variable.
    - variables are typed:
        - in case the type is “global”, the variable is shared among policies. In this case, the variable properties (currently limited to the type) are defined in a separate JSON section in the template file.
        - otherwise, type values are defined by the template, the model, the digital ecosystem model or manually entered in the PE.
- an (optional) list of lower level policies.

6. Appendix: Other type of change

We list here the remaining type of change for the DVA change management scenario, in Section 7.2.

Add a new video to a collection

A new video is added to *collection X*; this can mean that a video is attributed to that collection or a new DO is added with the property specified. This will trigger *change 0* of the policy, that in turn will create an *appliesTo* dependency between the policy and video file.

This will also activate the *appliesTo change 0* (constructor) that will check if any of the players can play the video file; creating the relative *canPlay* dependencies.

At the end of this step, the impact will check if there is any *canPlay* dependency between the player and the video file; if that's not the case, it will trigger transcoding, that will issue step **A3**, creating the *canPlay* dependency between the transcoded video file and the players for the transcoded format.

Change in player's supported formats

When updating a video player, it may be that the list of supported formats is also updated (old formats may be dropped; and new formats added). As an example, the QuicktimeVR format was introduced in 1994 and is unsupported by the currently version of Quicktime (files can still be rendered by Quicktime version 7 that needs to be separately installed).

In this situation, *canPlay change 0* will be activated, and will check, for all videos, if the format is still supported, deleting unsupported *canPlay* dependencies.

Any deletion of a *canPlay* dependency will then activate *change 1* (destructor); this will check if the video can still be played by any player, and if that's not the case, will trigger transcoding, following steps A2 and A3.

Change in video format

Any change in video format will carry the sequence of steps A3 (both when the change is internal; for example for an impact that will trigger transcoding, or external, for changes in the ecosystem for example if a user updates a video file manually).

7. Appendix: Rule based change management Proof of Concepts

This example is the implementation of the section 7 rule based change management for a single resource: the EUMETSAT data policy. Please refer to that section for the background of this PoC. This PoC makes use of a Digital Ecosystem Model build using the EcoBuilder and provides the SPIN rules for the model described. It is designed to be executed on a test environment built using the functional architecture illustrated in Figure 1. The example is described in section 7.2.4. The example consists on a rule implementation for the data policy that governs the release of data to the public, so it regards both policy implementation, QA and change management. The rule will enact the policy and also manage change for example in the release period of the policy. Table 7 indicates the mapping between the example and this rule implementation.

EUMETSAT example	**Generic rule**
`time_before_release`	`policy_property_x`
`appliesTo`	`dependency_1`
`EUMETSAT policy`	`policy_1`
`DOdata`	`digital_object_1`

Table 7: Mapping between the EUMETSAT example notions and notions used for the generic example

For every newly stored data in the private repository, relevant information should be populated in the ontology. More specifically:

- Every single DO should be stored in the ontology, through its relevant properties and values instantiations.
- Every single DO should be connected to a policy through the use of `lrm:Dependency` descriptors. For example, for `digital_object_1` the following triples regarding the policy dependency may exist:

  ```
  dependency_1      lrm:from     digital_object_1
  dependency_1      lrm:to       policy_1
  ```

  This type of dependency can be created automatically using the rule described in the original diagram, that is activated when new data is stored in the private repository. Since policy cannot include precondition-impact, this rule can be represented by an LRM dependency between the policy and itself, that can created for the purpose of populating the model.

7.1. Change Scenario 1: value in a policy parameter changed

A change in the value of the corresponding policy property affects the instance of the policy and thus of an attached notion in the corresponding instance of dependency. The change is described within the ontology, as an instance of delta, with the following triples (see also Figure 32):

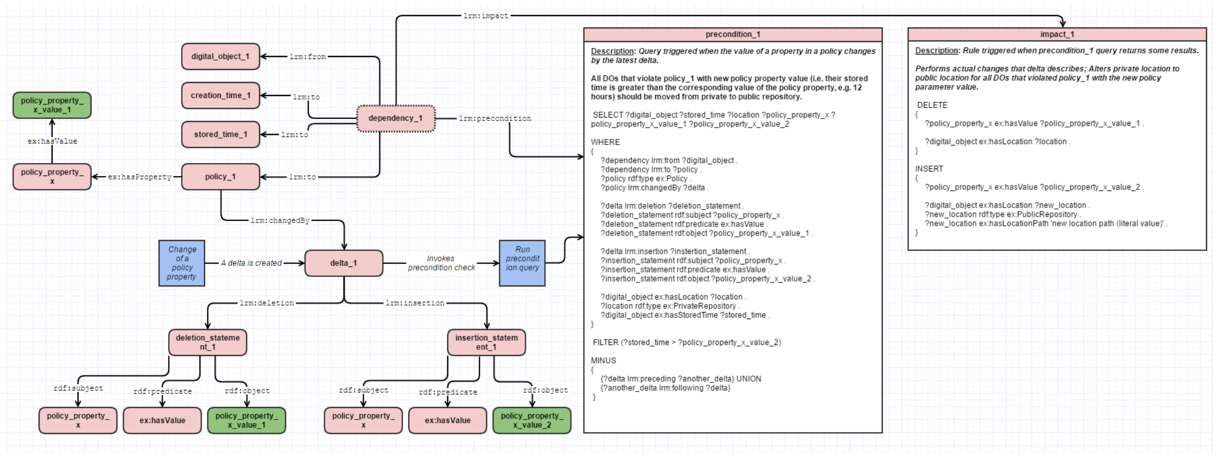
```
policy_1                 lrm:changedBy     delta_1
delta_1                  rdf:type          lrm:RDF_Delta
delta_1                  lrm:deletion      deletion_statement_1
deletion_statement_1     rdf:subject       policy_property_x
deletion_statement_1     rdf:predicate     ex:hasValue
deletion_statement_1     rdf:object        policy_property_x_value_1
delta_1                  lrm:insertion     insertion_statement_1
insertion_statement_1    rdf:subject       policy_property_x
insertion_statement_1    rdf:predicate     ex:hasValue
insertion_statement_1    rdf:object        policy_property_x_value_2
```

7.1.1. The role of precondition and impact

For every instance of dependency, we may define through `lrm:precondition` and `lrm:impact` properties specific SPIN rules that are triggered upon a new change (delta); here, the delta is related to the change of the value in the policy property, from 24 hours to 12 hours. The query, as seen in Figure 32, requests for all DOs that violate the policy, by taking into account the new value of the policy property defined.

We also assume that a time based process (triggered every certain time) will generate a delta in the 'stored time' property of an object, that indicates that the object has been in available in private for that amount of time.

If results are derived from this SPARQL query defined as precondition value, then the impact takes action. The rule described as value of the corresponding impact accomplishes the following tasks: (i) updates the ontology by performing the actual changes that delta describes (now, `ex:hasValue` points to the new value of policy's property), (ii) alters private location to public location for all DOs that were returned from preconditions query.



Description: *Query triggered when the value of a property in a policy changes by the latest delta.*

All DOs that violate policy_1 with new policy property value (i.e. their stored time is greater than the corresponding value of the policy property, e.g. 12 hours) should be moved from private to public repository.

```
 SELECT ?digital_object ?stored_time ?location ?policy_property_x ?
policy_property_x_value_1 ?policy_property_x_value_2

WHERE
{
   ?dependency lrm:from ?digital_object .
   ?dependency lrm:to ?policy .
   ?policy rdf:type ex:Policy .
   ?policy lrm:changedBy ?delta .

   ?delta lrm:deletion ?deletion_statement .
   ?deletion_statement rdf:subject ?policy_property_x .
   ?deletion_statement rdf:predicate ex:hasValue .
   ?deletion_statement rdf:object ?policy_property_x_value_1 .

   ?delta lrm:insertion ?instertion_statement .
   ?insertion_statement rdf:subject ?policy_property_x .
   ?insertion_statement rdf:predicate ex:hasValue .
   ?insertion_statement rdf:object ?policy_property_x_value_2 .

   ?digital_object ex:hasLocation ?location .
   ?location rdf:type ex:PrivateRepository .
   ?digital_object ex:hasStoredTime ?stored_time .
}

 FILTER (?stored_time > ?policy_property_x_value_2)

MINUS
{
   {?delta lrm:preceding ?another_delta} UNION
   {?another_delta lrm:following ?delta}
 }
```

Description: *Rule triggered when precondition_1 query returns some results.*

Performs actual changes that delta describes; Alters private location to public location for all DOs that violated policy_1 with the new policy parameter value.

```
 DELETE
{
   ?policy_property_x ex:hasValue ?policy_property_x_value_1 .

   ?digital_object ex:hasLocation ?location .
}

INSERT
{
   ?policy_property_x ex:hasValue ?policy_property_x_value_2 .

   ?digital_object ex:hasLocation ?new_location .
   ?new_location rdf:type ex:PublicRepository .
   ?new_location ex:hasLocationPath 'new location path (literal value)' .
}
```

Figure 32: A change is described within the ontology, as an instance of delta

7.2. Change Scenario 2: total time during which the DO is stored changed

When a DO is newly stored in a repository, a counter indicating the time passed since the data was stored, is attached to the DO; here we call it as *stored time*. *Stored time* represents the age of the object within the repository. As time passes by, the *stored time* of the DO increases: consider a time based trigger that will change the stored time every X units of measurement of time (e.g. every 1 hour). For every change, a time-related delta will be created, with the following triples (see also Figure 33):

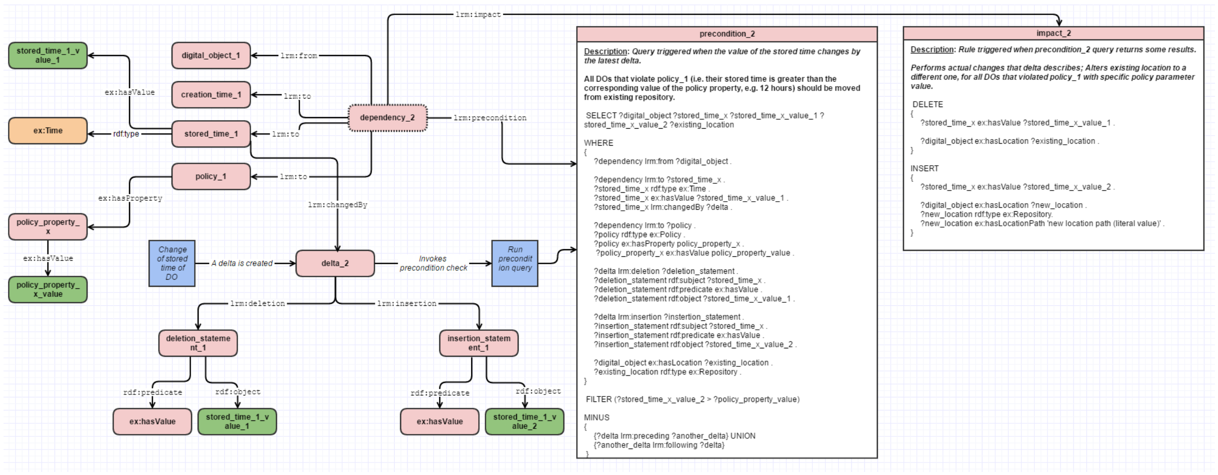
`stored_time_1`	`lrm:changedBy`	`delta_2`
`delta_2`	`rdf:type`	`lrm:RDF_Delta`
`delta_2`	`lrm:deletion`	`deletion_statement_2`
`deletion_statement_2`	`rdf:predicate`	`ex:hasValue`

```
deletion_statement_2     rdf:object        stored_time_1_value_1
delta_2                  lrm:insertion     insertion_statement_2
insertion_statement_2    rdf:predicate     ex:hasValue
insertion_statement_2    rdf:object        stored_time_1_value_1
```

7.2.1. The role of precondition and impact

As also seen in previously examined change scenario, given the policy that describes the need to move the DO into a different repository when the stored time exceeds a specific total time limit, we create an instance of dependency between the DO and its current stored time. The precondition of the dependency will be triggered when the stored time value changes according to the latest delta. As seen in Figure 33, the precondition will query for all DOs that violate the policy, by taking into account the new value of the stored time of DOs.

and impact here. If results are returned, then the SPIN rule included in the relevant impact takes action, which triggers the move process for changing the location of the derived DOs.



Description: *Query triggered when the value of the stored time changes by the latest delta.*

All DOs that violate policy_1 (i.e. their stored time is greater than the corresponding value of the policy property, e.g. 12 hours) should be moved from existing repository.

```
 SELECT ?digital_object ?stored_time_x ?stored_time_x_value_1 ?
stored_time_x_value_2 ?existing_location

WHERE
{
   ?dependency lrm:from ?digital_object .

   ?dependency lrm:to ?stored_time_x .
   ?stored_time_x rdf:type ex:Time .
   ?stored_time_x ex:hasValue ?stored_time_x_value_1 .
   ?stored_time_x lrm:changedBy ?delta .

   ?dependency lrm:to ?policy .
   ?policy rdf:type ex:Policy .
   ?policy ex:hasProperty policy_property_x .
    ?policy_property_x ex:hasValue policy_property_value .

   ?delta lrm:deletion ?deletion_statement .
   ?deletion_statement rdf:subject ?stored_time_x .
   ?deletion_statement rdf:predicate ex:hasValue .
   ?deletion_statement rdf:object ?stored_time_x_value_1 .

   ?delta lrm:insertion ?instertion_statement .
   ?insertion_statement rdf:subject ?stored_time_x .
   ?insertion_statement rdf:predicate ex:hasValue .
   ?insertion_statement rdf:object ?stored_time_x_value_2 .

   ?digital_object ex:hasLocation ?existing_location .
   ?existing_location rdf:type ex:Repository .
}

 FILTER (?stored_time_x_value_2 > ?policy_property_value)

MINUS
{
   {?delta lrm:preceding ?another_delta} UNION
   {?another_delta lrm:following ?delta}
}
```

Description: *Rule triggered when precondition_2 query returns some results.*

Performs actual changes that delta describes; Alters existing location to a different one, for all DOs that violated policy_1 with specific policy parameter value.

```
 DELETE
{
   ?stored_time_x ex:hasValue ?stored_time_x_value_1 .

   ?digital_object ex:hasLocation ?existing_location .
}

INSERT
{
   ?stored_time_x ex:hasValue ?stored_time_x_value_2 .

   ?digital_object ex:hasLocation ?new_location .
   ?new_location rdf:type ex:Repository.
   ?new_location ex:hasLocationPath 'new location path (literal value)' .
}
```

Figure 33: Dependency and delta graph for a change in value of stored time of a digital object

8. Appendix: Semantic and User Community drift QA PoC

This final example makes use of the same change management concepts and architecture, applied this time to change in semantics and User Communities described in Section 10. We here define a use scenario, and the SPIN rule implementation for the threshold validation.

8.1. Policy and drift model use scenario

In this situation, an organisation wants to maintain its collections usable by its user communities, a common but very broad objective for digital preservation. This high level policy is expressed in a guidance policy:

Policy 0: "*The content of our collections will remain usable by its user communities"*

In order to validate this high level policy, one possible quality assurance methodology can be defined in the human driven, expensive process:

QA method 0: Every 10 years, run a user study (focus group, survey) to discover possible issues with usability of content. If there are some issues with the content, it is highly likely that these will be discovered by the user studies; still it is also expensive to run such user studies, and some sudden changes may be not revealed until the next run of the user study. For this reason, the organisation creates a derived policy (dependent on policy 0) that can be run automatically, based on concept drift as an indicator of significant change in semantics:

Policy 1: *"If drift in the concepts monitored is above 30% inform the collection manager"*

In order to validate such policy, the methodologies described in Section 10 are used, to automatically monitor and report significant drifts:

QA method 1: When there is change, compute concept drift in the ontologies used to record metadata and index the collections. The concept drift will be recorded in the Digital Ecosystem Model ontology. A SPIN rule (described in the next paragraph) will alert if any newly recorded drift value in the collection ontologies is above the threshold (30%).

Finally, a similar policy is defined to monitor drift in user community:

Policy 2: *"If drift in the User Community is above 30% inform the collection manager"*

In this case, the implementation (QA method) is completely analogous to the one earlier described. The analysis of the user community topics will be run periodically and update the DEM model drift ontology (using LRM deltas) so that an automated rule will verify the threshold and generate alerts if it is surpassed. The alerts sent to the collection manager will include a list of the terms surpassing the threshold so there is a precise knowledge on the changes that are causing the alert. In both cases, the alerts will need to be validated and handled manually.

8.2. Drift Ontology and threshold definition

The drift ontology is a model of concept drift measures between two *concepts* or *two versions of a concept in two models* that represent changes in time. Its main notions are *ConceptDrift* and its three subclasses: *LabelConceptDrift*, *IntensionConceptDrift* and *ExtensionConceptDrift* [Stavropoulos et al., 2016; PERICLES D4.4., 2016; Wang et al., 2011]. The object properties named *'from'* and *'to'* connect the concept(s)[29] under drift analysis and the data property named *'value'* declares the measure of drift (range between zero and one). An example of triples describing the aforementioned relations is given below:

```
intension_concept_drift_1     rdf:type     :Intension_Concept_Drift
intension_concept_drift_1     :from        :DigitalArtwork
intension_concept_drift_1     :to          :DigitalArtwork
```
[30]

[29] Concepts may belong to any defined class in the ontology.

[30] It should be noted that instances of `DigitalArtwork` which are connected with the instance of

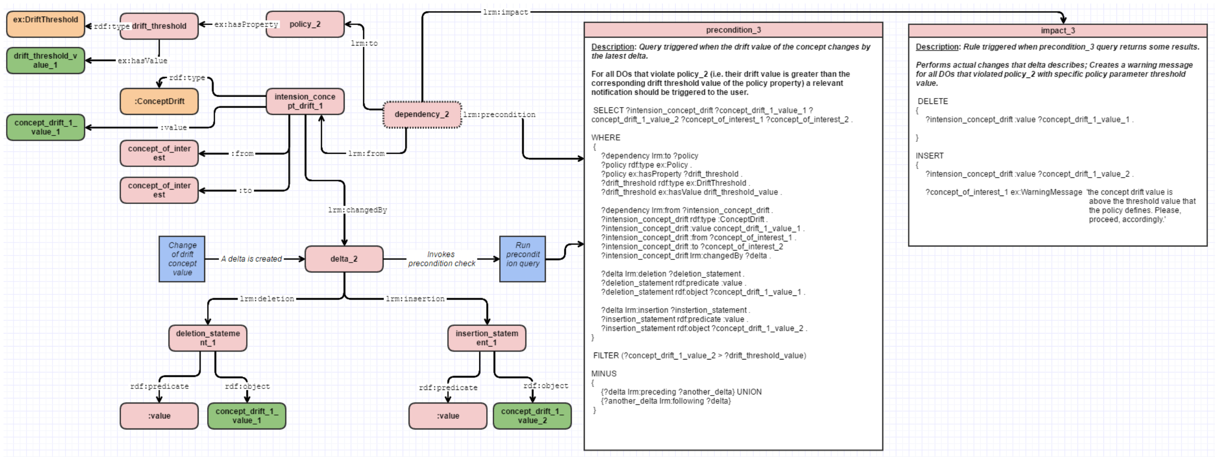
```
intension_concept_drift_1      :value      '0.4'
```

where we examine the *IntensionConceptDrift* of the *DigitalArtwork* concept in two different models within time.

We can model the policy that monitors if the drift value of a concept exceeds a specific threshold, according to the following triples:

```
concept_drift_policy    rdf:type          ex:Policy
concept_drift_policy    ex:hasProperty    drift_threshold
drift_threshold         rdf:type          ex:DriftThreshold
drift_threshold         ex:hasValue       '0.3'
```

In order to monitor if the new drift value of a specific concept drift exceeds the relevant drift threshold, we create a dependency between the policy the concept of interest, the precondition and impact of which carry the SPIN rules that do all relevant actions. Rules and relations can be seen in detail in Figure 34:



precondition_3

<u>Description</u>: ***Query triggered when the drift value of the concept changes by the latest delta.***

For all DOs that violate policy_2 (i.e. their drift value is greater than the corresponding drift threshold value of the policy property) a relevant notification should be triggered to the user.

```
 SELECT ?intension_concept_drift ?concept_drift_1_value_1 ?
concept_drift_1_value_2 ?concept_of_interest_1 ?concept_of_interest_2 .

WHERE
{
   ?dependency lrm:to ?policy
   ?policy rdf:type ex:Policy .
   ?policy ex:hasProperty ?drift_threshold .
   ?drift_threshold rdf:type ex:DriftThreshold .
   ?drift_threshold ex:hasValue drift_threshold_value .

   ?dependency lrm:from ?intension_concept_drift .
   ?intension_concept_drift rdf:type :ConceptDrift .
   ?intension_concept_drift :value concept_drift_1_value_1 .
   ?intension_concept_drift :from ?concept_of_interest_1 .
   ?intension_concept_drift :to ?concept_of_interest_2
   ?intension_concept_drift lrm:changedBy ?delta .

   ?delta lrm:deletion ?deletion_statement .
   ?deletion_statement rdf:predicate :value .
   ?deletion_statement rdf:object ?concept_drift_1_value_1 .

   ?delta lrm:insertion ?instertion_statement .
   ?insertion_statement rdf:predicate :value .
   ?insertion_statement rdf:object ?concept_drift_1_value_2 .
}

 FILTER (?concept_drift_1_value_2 > ?drift_threshold_value)

MINUS
{
   {?delta lrm:preceding ?another_delta} UNION
   {?another_delta lrm:following ?delta}
 }
```

impact_3

<u>Description</u>: ***Rule triggered when precondition_3 query returns some results.***

Performs actual changes that delta describes; Creates a warning message for all DOs that violated policy_2 with specific policy parameter threshold value.

```
 DELETE
{
   ?intension_concept_drift :value ?concept_drift_1_value_1 .

}

INSERT
{
   ?intension_concept_drift :value ?concept_drift_1_value_2 .

   ?concept_of_interest_1 ex:WarningMessage  'the concept drift value is
                                              above the threshold value that
                                              the policy defines. Please,
                                              proceed, accordingly.'
```

Figure 34: Dependency and delta graph for a change in drift value of a specific concept in a model

9. Appendix: Investigations of Community Change

In the following, we describe the Tumblr data collection process and resulting data; we then describe the social network analysis and properties of the Tate Tumblr network; next, we describe our analysis of the textual content of Tumblr posts from the Tate network using topic modelling and relate this to changes in the Tate community over time, illustrating this with an example from 2012. After summarising and discussing these results together, we propose potentially useful measures and suggested thresholds for incorporating into any form of monitoring community change within

`intension_concept_drift_1` via the object properties `:from` and `:to`, are not the same; they have different URIs corresponding to different representation models.

the Tumblr data relating to Tate.

9.1. Data Collection Method

Tumblr posts were previously collected for the study of social media content in this project (reported in D4.3 [PERICLES D4.3, 2016]): due to Tumblr's design, it is convenient to search for specific tags rather than solely for terms, and therefore a snowball methodology is used to spiral outwards from initial hits to other posts or blogs that may be of relevance ([Biernacki & Waldorf, 1981; Atkinson & Flint, 2001]). Use of the substring 'tate' in case-insensitive search system captures tagged posts, as well as mentions of the term itself. A proportion of false positive terms, are also retrieved.

For the purposes of the present study we apply strict filtering rules in order to limit material returned to material containing either the string 'tate' with appropriate word boundaries, or material containing the Tate's hostname. For the Tumblr data, a search was completed for posts of any age containing the term 'Tate'. Of the original 70,000 posts from 01-Feb-2005 to 24-03-2015, 3,093 were examined in this present process, textual data from the 'body', 'caption', 'description' fields gave a total of 473,680 words, 2,793,500 characters (excluding HTML and with URLs normalized; for the topic modelling analysis, further processing [stopword removal and stemming; using the using the 422 function words from [McDonald, 2000] and the Porter stemming algorithm (http://tartarus.org/martin/PorterStemmer/), respectively] resulted in a corpus of 286,360 words; 1,838,736 characters). With contrast to the two other social media platforms and search term approach used ('tate'), we note that the snowball sample method used gives lower precision than the approaches used for the other social media platforms (e.g. for Twitter, 22,00/222,356, or ~10%); however, this is unsurprising given the highly connected and diffuse characteristics of these social media networks, in many cases Tumblr posts linked to material not solely about Tate nor relevant to art in general (we discuss this in further detail in [PERICLES D4.3, 2016]; a more general description of the data is also presented in that deliverable).

9.2. Network analysis of Tumblr data

Network analysis was performed on the cumulative Tumblr data, with these cumulative measures updated on a daily basis. Links (edges) between users relevant to the Tate community were identified on the basis of Tumblr connections, with data processed using custom software created in Python. Note that these are shown only from 2009, since before this date there were not enough suitable connections between nodes. Network statistics were calculated at the relevant intervals using the R software package and were average betweenness, number of clusters, density, and average degree, important for identifying network and community change (McCulloh and Carley, 2011). These are shown in Figure 36 (to account for the variation in values across the statistical measures, the y axis is shown as a logarithmic scale; raw counts of clusters can be found in Figure 37). Network graphs describing the Tumblr data are also shown in Figure 35, with clusters labelled with node user names where relevant (data from 2015 is not included in this analysis since it is incomplete). Note that edges are here illustrated as grey lines, and that in cases where there is a high density of connection between nodes, this shows up as areas of solid grey (e.g., 7/2014).

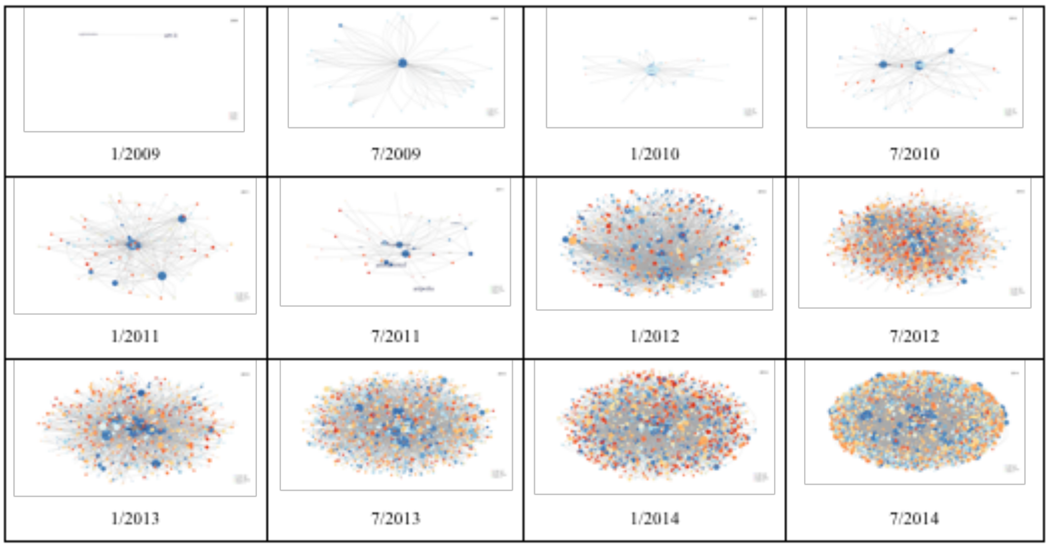


Figure 35: Tumblr network over time

Examining Figure 35, which shows 6 monthly snapshots of the data, we can see that after a very small start to the network in 1/2009, this quickly grows with a greater number of users later in that year (7/2009), however it is not until mid 2010 (7/2010), that the nodes increase and in turn form more clusters. This pattern continues in 2011 (1/2011-7/2011), which can be seen more obviously in terms of increase in number of clusters in Figure 37).

The year 2012 however sees a massive growth in the number of users (nodes) in the Tate Tumblr network, and with this an increase in the number of clusters (1/2012-7/2012); with this pattern clearly captured by the average betweenness measure showing a great increase in the connectedness of the nodes, and its effective inverse, network density which shows a reduction (Figure 36). From 2012 until the end of our time period (mid 2014; 1/2012-7/2014), network grown continues, but at a much more steady pace (although note that this is illustrated using a logarithmic scale), reflected in average betweenness and network density; in general the remaining network measure, average degree, is shows a steady increase throughout the whole time period except for two bumps relating to network growth in 2009 and 2012 (Figure 36). Considering Tumblr cluster counts in isolation (Figure 37): here we can see that clusters in general relate to an increase in network size, such as in 2012 and 2014, however this measure is noticeably erratic, with this more noticeable since these network metrics are calculated on cumulative daily data (e.g., the trough in mid 2012 after the peak in early 2012).

Viewing these measures as a whole, we can see the development of a network relating to a Tumblr community sharing an interest in Tate. Although to a lesser or greater extent, average betweenness, density, and average degree are a function of network size (nodes and edges), we note that there is additionally some utility in the number of clusters within the network for better understanding its structure. However, as the erratic pattern of Figure 37 attests, in practice care may need to be taken in interpreting changes in this measure.

In the next paragraph, we supplement the view of network change identified statistically, with analysis relating to the content of the Tumblr community, and how this can help us to understand change.

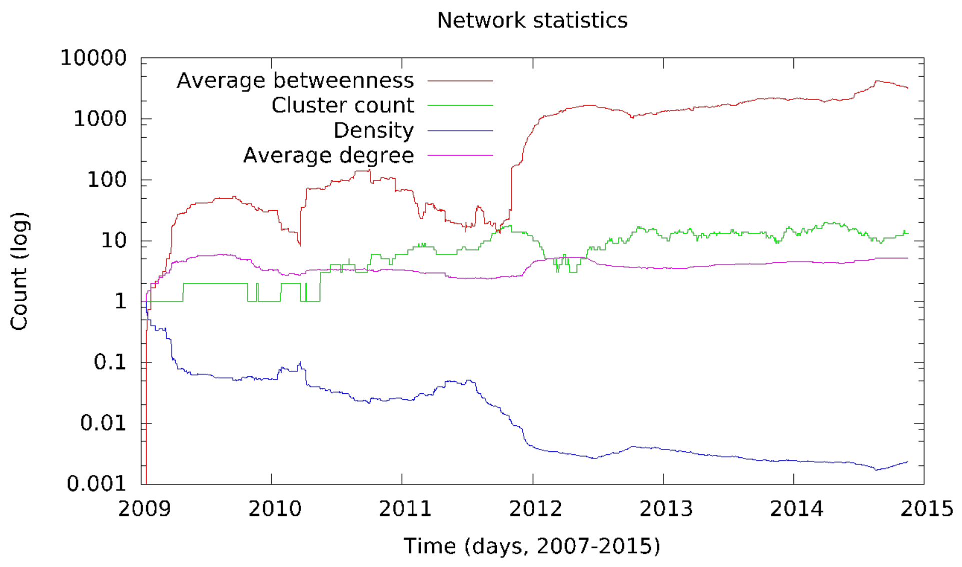


Figure 36: Tumblr network statistics

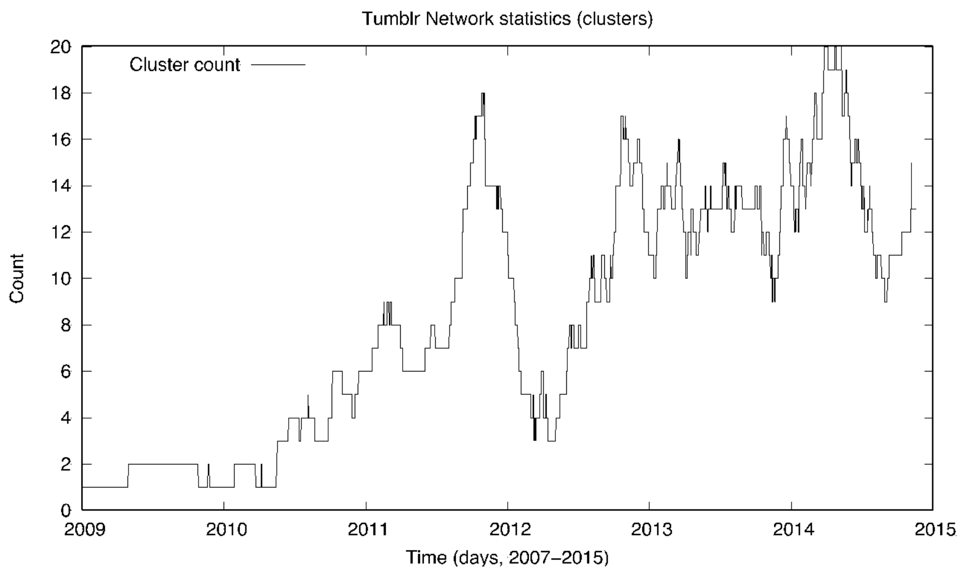


Figure 37: Tumblr network cluster counts

9.3. Topic modelling of Tumblr content

Analysis of content can provide better understanding of communities in the Tumblr data. Here we adopt an unsupervised machine learning method – topic modelling – which we apply to the text of Tumblr posts in order to identify broad themes. Topic models aim to uncover hidden thematic structures or 'topics' that occur in a collection of documents utilising unsupervised machine-learning techniques (Blei, 2012). A topic consists of a cluster of words or phrases that show similar patterns of occurrence; documents may relate to more than one topic, and topic modelling calculates a weight with which each topic relates to a particular document. We used Latent Dirichlet allocation (LDA) for topic modelling (Blei 2012). As a generative technique, LDA starts with a model that is then used to describe the data by adjusting the parameters to fit the model. The assumption is that the whole corpus of documents contains k number of topics (specified by the user), and that each document talks about these k topics (to a greater or lesser extent). Therefore, each word in a document depends on both the topics selected for that document as well as the word distribution within each of these topics. This intuition is operationalized as a Bayesian Network that models this document generation process.

Since topic modelling is a probabilistic method there are several possible solutions to representing the data, with this process requiring input from the researcher in an iterative process. Here we briefly describe our method: Using the processed data from 'body', 'caption', 'description' fields as described above (cf. Wang et al. 2013), topic modelling was performed on the 3,093 documents using the LDA package in Mallet (http://mallet.cs.umass.edu) to generate models for a variety of number of topics, ranging from 3-20 (3, 5, 7, 10, 15, 20). This was in order to select a number of topics which best describe the data. In all topic modelling described here, default parameter settings were used except in the case of α where a relatively low value (0.01) was specified in order to generate topics which relate more distinctly to particular documents (cf. Mimno et al. 2014). Two researchers familiar with the data set visually inspected the model outputs to evaluate the 'topic keys' (words most representative of the topic), to determine whether they contained a disproportionate number of poor topics which would indicate a poor description of the data (specifically topics which were too general, too specific, repetition with other topics, or internally inconsistent). These key words were then used to manually search through documents containing the respective topics in models considered suitable for our analysis, in order to get a better sense of that topic for interpretation and naming of the topics. Following this process we settled on two topic models which appeared to provide a good summary of the data: these specified 5 and 15 topics, and they are shown (with their topic keys) in Table 8 and Table 9 respectively, in descending order of their proportion in the data. In the following section, we describe these in more detail in relation to the Tumblr Tate community data.

Topic Label	Topic ID	Key Items	Proportion

URL/Modern	4	removedurl tate modern art london week museum exhibit matiss	0.32
URL/ArtworkProperties	0	removedurl paint tate work paper cm removedimg canva artist	0.31
IMG/description	1	removedimg work exhibit show tate art piec time paint	0.26
ExhibitionInfo	2	art work artist exhibit tate museum modern galleri perform	0.20
Foreign	3	de video art la le artist pari film en	0.02

Table 8: Five topic model of Tumblr data

Topic Label	Topic ID	Key Items	Proportion
IMG/PaintCharacteristics	10	paint work artist removedurl removedimg imag figur colour form	0.22
IMG/Exhibition/Art	1	removedimg work show time exhibit make veri year love	0.21
URL/Description/Materials	12	removedurl tate cm paper oil canva sourc collect removedimg	0.20
URL/Modern/Performance	6	removedurl tate modern removedimg artist music perform sound tank	0.20
Exhibition/Film	9	art work tate exhibit artist museum film present perform	0.14
Descriptions/Britain/URL	14	art exhibit galleri london tate removedurl artist britain british	0.14
3D	3	space sculptur piec galleri yellow black build tate rothko	0.10
ArtworkContext/URL	0	art cultur artist removedurl work polit commun peopl beui	0.08
Modern/Artists/URL	7	kusama tate hirst modern yayoi removedurl damien exhibit room	0.08
URL/StIves/Landscape	5	st iv removedurl landscap war tate sea mso picasso	0.06
URL/Exhibition/London	13	removedurl galleri london august novemb matiss tate modern lincoln	0.05
URL/Modern/Podcast	11	removedurl art modern week podcast museum exhibit matiss barlow	0.05

Cities	4	citi al eliasson removedurl walk york film london project	0.02
Video	2	video art artist instal exhibit paik de pari includ	0.01
Foreign	8	de la le en du dan art au par	0.01

Table 9: Fifteen topic model of Tumblr data

9.4. Overview of the topics

5 Topic solution: The five topic description of the Tumblr data shows four contentful and more frequently used topics, with the fifth (Foreign) showing much lower usage (and which relates mainly to non-English texts): The two topics with a similarly great level of use are URL/Modern and URL/ArtworkProperties (0.32 and 0.31, respectively). Of these, the first relates mainly to the Tate Modern (e.g., exhibitions, or passing references to the gallery), with the second relating to the physical properties of artworks (e.g., factual information such as canvas size or materials used). Both of these topics feature URLs which presumably are being referenced by the author and are – to a greater or lesser degree – the subject of their post.

The topic with the third greatest proportion is IMG/description (0.26), which contrasts nicely with the previous topic URL/ArtworkProperties, as it refers to a linked image (which the post relates to), and a description of the art object, however rather than being concerned with the factual physical and material aspects, the IMG/description topic gives a personal perspective and interpretation of the art object (e.g., the flow of the brushstrokes, or the meaning of the scene). The final topic, ExhibitionInfo (0.20), provides information about exhibitions, perhaps especially promotional material advertising exhibitions.

15 Topic solution: The 15 topic description of the data can be grouped into four main clusters based on the proportion of topic usage in the Tumblr data, although as may be expect given the greater number of topics, the proportions found for topic usage are lower than for those of the 5 topic solution: The first group with proportions of 0.20 or greater within the data are IMG/PaintCharacteristics (0.22), IMG/Exhibition/Art (0.21), URL/Description/Materials (0.20), and URL/Modern/Performance (0.20). The first topic mentions descriptions of paint, techniques and characteristics in relation to an image(s), the second again contains an image as well discussing exhibitions in relation to art, the third gives a factual description of an art object in terms of materials and referencing a URL, and the fourth most frequently used topic includes a URL along with content relating to (Tate) modern and performance.

The second more frequent grouping of topics (with proportions of between 0.10 and 0.19), are Exhibition/Film (0.14), Descriptions/Britain/URL (0.14), 3D (0.10). The first of these topics contains reference to exhibition along with mention of film, the second topic contains factual catalogue-type information relating to (Tate) Britain along with a URL, and the third topic of this group contains materials and charactertics of three dimensional art objects.

The third group have usage proportions of between 0.5 and 0.9 in the Tumblr data, and are ArtworkContext/URL (0.08), Modern/Artists/URL (0.08), URL/StIves/Landscape (0.06), URL/Exhibition/London (0.05), and URL/Modern/Podcast (0.05). Of these topics, the first describes the context of the art object more generally (and less prominently a URL), the second relates to (Tate) Modern and names of artists from the past 20 or so years (and also a URL), the third to some extent mentions (Tate) St Ives, as well as other concepts such as 'landscape', and the final topic contains a URL along with (Tate) Modern, which appears to be in the context of exhibitions, and also mentions 'podcast'.

Finally, the three least used topics are Cities (0.02), Video (0.01), and Foreign (0.01), which are respectively, mentions of the word 'city/ies' or names of cities, references to video (as in art object, but also videos posted on social media), and non-English words.

Comparing the two topic models run on the Tumblr data, it is unsurprising that the 15 topic solution provides greater granularity than the 5 topic model. However, what seems more apparent between the two models is that the 5 topic model is more abstract, giving a better sense of the concepts and types of posts it is describing (e.g., description of an image, materials used, advertising an exhibition), whereas the 15 topic model provides more information about specific content (whether content relates to 3D, Tate Modern artists, St Ives, Tate Britain). Both of these models, and their respective granularity, have advantages and disadvantages when describing the data in light of identifying community change. We discuss this briefly in relation to the data, below.

9.5. Exploring Tumblr content over time: analysis and an example

So far we have shown how social network analysis provides information about the size and relationships between the Tate community identified on Tumblr, and topic modelling provides information about the content of Tumblr posts. In this paragraph we provide analysis of how the Tumblr topics identified in the previous paragraph change over time, and how these can be related back to the network changes identified in the first paragraph; this will in turn give us a better understanding of user community change, especially in terms of how their concerns – expressed through Tumblr posts – change over time. In addition to describing the results, we will provide examples of community change identified in this data, and in the final section describe how these can be incorporated into an automatic process to identify community change.

Topic relationships to documents used to create the two models in the previous section were used to provide the primary topic representing each Tumblr post (i.e. the single topic showing the highest proportion of usage in each document); these were then summed for each 6 month period (January-June 2009, July-December 2009, January-June 2010, etc. until the final period in our data collection, January-June 2015). We note that 4 posts used in the previous generation of the topic models were excluded since they dated from before 2009; this left 3,089 remaining posts (as per the social network analysis, above). Usage of content described using the 5 topic model is shown in Figure 38, and the 15 topic model is shown in Figure 39; the y axis is the frequency of posts which are primarily described by that topic.

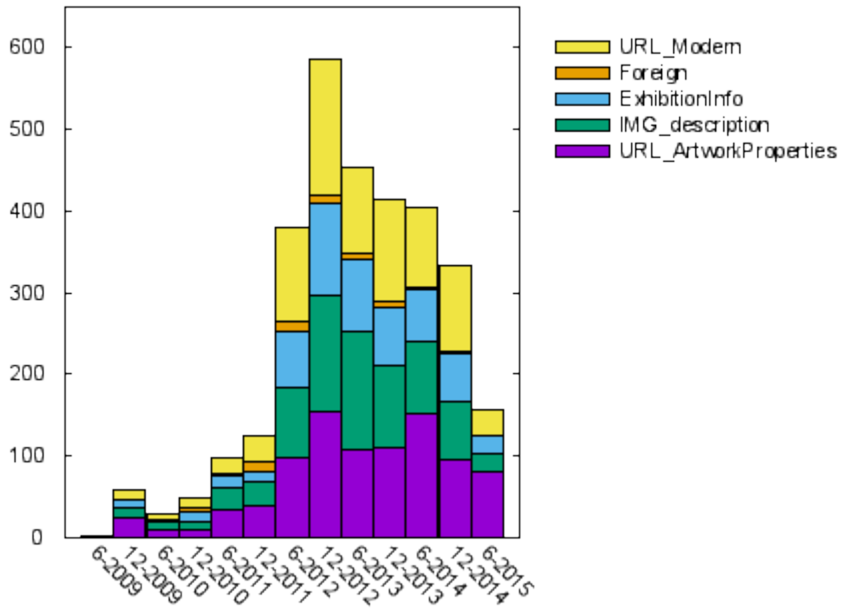


Figure 38: Tumblr topic frequency over time (2009-2015)(5 topic model)

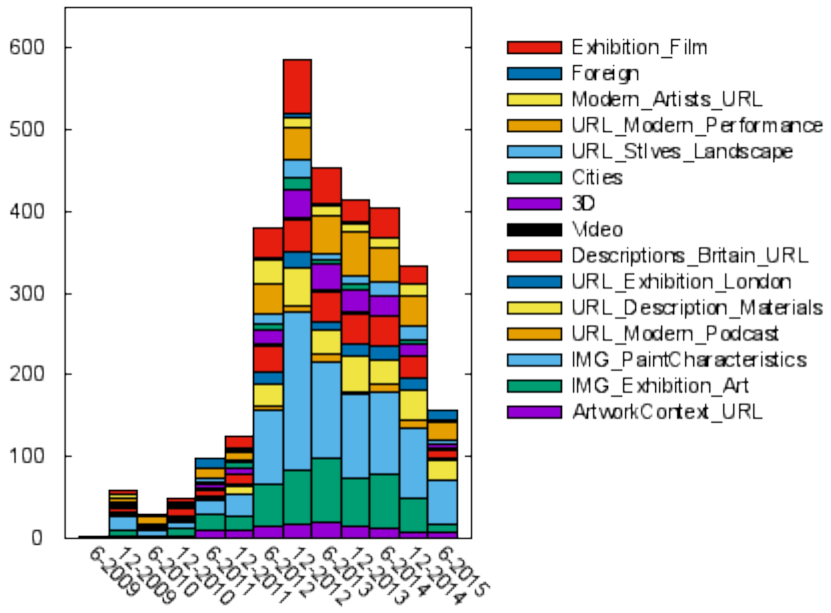


Figure 39: Tumblr topic frequency over time (2009-2015) (15 topic model)

Both the 5 and 15 topic models of the Tumblr data over time show the rapid increase in posting activity in 2012, reaching a peak in the latter half of that year; activity remains high over the next two years, albeit generally declining; the first half of 2015 has relatively low Tumblr posting activity for the Tate community, but this may be due to an incomplete collection of data for this period. The peak of activity identified in 2012 appears related to the massive network growth also shown in the social network metrics of Figure 36. Using this data as an example, we now explore how this community growth in 2012, and following years, is reflected in the change of content in the Tate Tumblr community.

Looking in particular at the Tate Tumblr community use of the 5 topics over time, we note content usage changes as follows: From 2012 onwards, we see a decline in use of URL_ArtworkProperties (relating to catalogue descriptions of art objects), which picks up again in 2014; with this dip in usage coinciding with a bump in IMG_description (which relates more to images and their exhibition rather than catalogue data) in late 2012 and 2013. In addition, there is a consistently higher use of ExhibitionInfo following 2012 (giving details of exhibitions without relation to a specific image), and a relatively higher proportion of posts primarily concerned with URL_Modern (perhaps indicated a higher profile of the Tate Modern and modern works within the collection). We note the very rare use of primarily Foreign content, with the exception of the latter half of 2010.

In summary, the 5 topic model shows that this network and resulting community change appears to indicate an apparent initial focus on images relating to exhibits, but this appears to have been used in place of posts sharing catalogue information of art objects. This change has also resulted in greater sharing of exhibitions at the Tate (possibly in promotion) and reference to the Tate Modern.

In terms of the 15 topic model, we find that posts primarily about IMG_Exhibition_Art and URL_Modern_Performance dip around 2012, with Modern_Artistists_URL, in contrast, peaking around this time; in general, IMG_Paint_Characteristics and URL_Description_Materials posts increase in proportion and stay relatively more common in and following 2012. We note that the remaining topics were used relatively infrequently or inconsistently across this time period, and so we do not discuss them in detail.

In summary, the 15 topic data appears to reveal that the rapid network increase around 2012 temporarily focused on Modern_Artists_URL (posts of links relating to Tate Modern artists), while posts relating to exhibitions in general (IMG_Exhibition_Art) and performances at the Tate Modern (URL_Modern_Performance) decreased around this time; this network and resulting community change resulted in a continuing, greater number of detailed posts focussing on the physical properties and painterly aspects of art objects (IMG_Paint_Characteristics and URL_Description_Materials), which in practice may be the popular describing and critiquing of objects in the Tate catalogue. More generally, we see that the granularity of the 15 topic model provides detailed topics which come in and out of usage, rather than the fairly consistent usage (albeit with variations in proportion) which is found with the 5 topic model.

10. Appendix: Policy driven Digital Ecosystem inspired by CERN LHC data management

This example is a cross task effort as an application of the Policy and QA Model (section 7.1) using the EcoBuilder. The application of the policy and QA model for a digital ecosystem is exemplified with a scenario inspired by the CERN LHC data management and quality assurance procedures as worked out at a PERICLES evaluation workshop on the policy model in Brussels (October 2015), and further refined and reviewed after the workshop. This scenario is implemented using the EcoBuilder's Java API and applies the policy and QA model using the policy driven modelling strategy described in D3.5. The resulting model was sent to an ERMR test instance. This example further aims to show the level of complexity that can be described for the scenario without getting into details that would make the description too hard to comprehend. Entities from the DEM-Analysis model [D3.5] are introduced to facilitate the analysis of the resulting model, which is shown in Figure 40 .

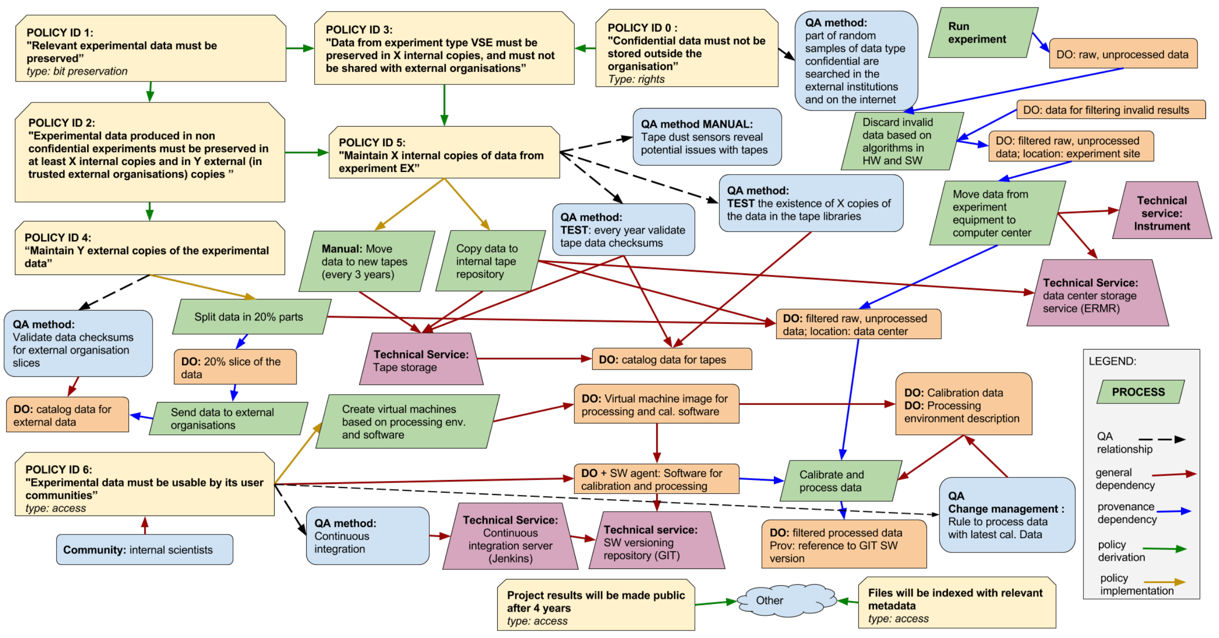

POLICY ID 1: "Relevant experimental data must be preserved" type: bit preservation
POLICY ID 3: "Data from experiment type VSE must be preserved in X internal copies, and must not be shared with external organisations"
POLICY ID 0 : "Confidential data must not be stored outside the organisation" Type: rights
QA method: part of random samples of data type confidential are searched in the external institutions and on the internet
Run experiment
DO: raw, unprocessed data
DO: data for filtering invalid results
POLICY ID 2: "Experimental data produced in non confidential experiments must be preserved in at least X internal copies and in Y external (in trusted external organisations) copies "
POLICY ID 5: "Maintain X internal copies of data from experiment EX"
QA method MANUAL: Tape dust sensors reveal potential issues with tapes
Discard invalid data based on algorithms in HW and SW
DO: filtered raw, unprocessed data; location: experiment site
QA method: TEST the existence of X copies of the data in the tape libraries
Move data from experiment equipment to computer center
Technical service: Instrument
POLICY ID 4: "Maintain Y external copies of the experimental data"
Manual: Move data to new tapes (every 3 years)
Copy data to internal tape repository
QA method: TEST: every year validate tape data checksums
Technical Service: data center storage service (ERMR)
QA method: Validate data checksums for external organisation slices
Split data in 20% parts
DO: filtered raw, unprocessed data; location: data center
DO: 20% slice of the data
Technical Service: Tape storage
DO: catalog data for tapes
LEGEND:
PROCESS
DO: catalog data for external data
Send data to external organisations
Create virtual machines based on processing env. and software
DO: Virtual machine image for processing and cal. software
DO: Calibration data
DO: Processing environment description
QA relationship
general dependency
POLICY ID 6: "Experimental data must be usable by its user communities" type: access
DO + SW agent: Software for calibration and processing
Calibrate and process data
QA Change management : Rule to process data with latest cal. Data
provenance dependency
QA method: Continuous integration
Technical Service: Continuous integration server (Jenkins)
Technical service: SW versioning repository (GIT)
DO: filtered processed data Prov: reference to GIT SW version
policy derivation
Community: internal scientists
policy implementation
Project results will be made public after 4 years type: access
Other
Files will be indexed with relevant metadata type: access

Figure 40: CERN LHC inspired example DEM using the implementation of the Policy and QA model

10.1. Policy derivation

The CERN example describes which policies underlie the creation, processing, and preservation of experimental data. It defines quality assurance criteria to ensure these policies and describes the ongoing processes. As initial step two very generic preservation policies are introduced:

POLICY ID 1:

"Relevant experimental data must be preserved".

POLICY ID 0 :

"Confidential data must not be stored outside the organisation"

Policy 1 is refined into more specific policies which define the rules of handling confidential and non-confidential experimental data:

POLICY ID 2 for non-confidential data:

"Experimental data produced in non-confidential experiments must be preserved in at least X internal copies and in Y external (trusted external organisations) copies "

Policy 2 is further refined into a maintenance policy:

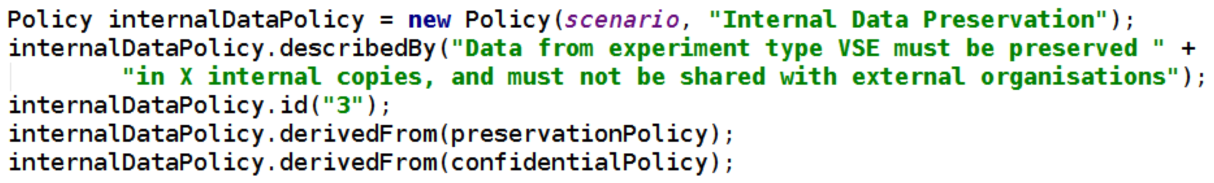
POLICY ID 4: "Maintain Y external copies of the experimental data".

For confidential data the policy 3 is derived from policy 1 and further constrained by policy 0:

POLICY ID 3 for confidential data:

"Data from experiment type VSE must be preserved in X internal copies, and must not be shared with external organisations".

The implementation of policy 3 is shown in Figure 41:



```
Policy internalDataPolicy = new Policy(scenario, "Internal Data Preservation");
internalDataPolicy.describedBy("Data from experiment type VSE must be preserved " +
        "in X internal copies, and must not be shared with external organisations");
internalDataPolicy.id("3");
internalDataPolicy.derivedFrom(preservationPolicy);
internalDataPolicy.derivedFrom(confidentialPolicy);
```

Figure 41: Extract of the CERN example implementation using the EcoBuilder API, showing the policy with ID 3.

Finally, policy 5 regulates the creation of data copies and an access policy 6 relates to the accessibility of the data by the user communities:

POLICY ID 5: "Maintain X internal copies of data from experiment EX"

POLICY ID 6: "Experimental data must be usable by its user communities"

Three further data related preservation policies are mentioned in the diagram but not implemented in detail, in order to keep the diagram readable:

Metadata policy: Files will be indexed with relevant metadata

Publication policy: Project results will be made public after 4 years

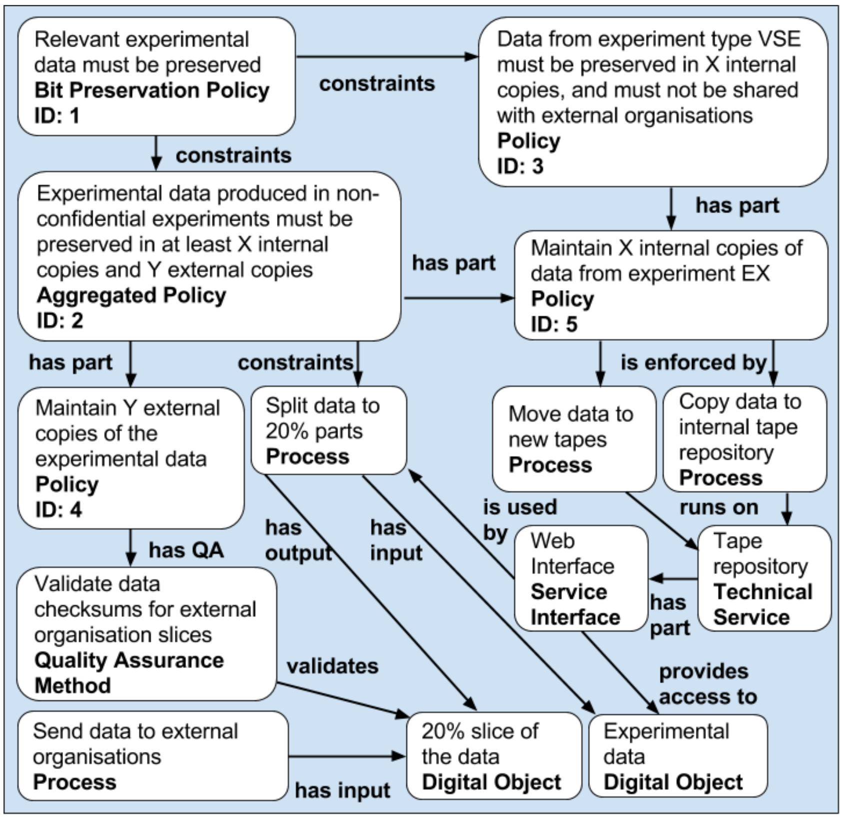

Relevant experimental data must be preserved
Bit Preservation Policy
ID: 1
constraints
Data from experiment type VSE must be preserved in X internal copies, and must not be shared with external organisations
Policy
ID: 3
constraints
has part
Experimental data produced in non-confidential experiments must be preserved in at least X internal copies and Y external copies
Aggregated Policy
ID: 2
has part
Maintain X internal copies of data from experiment EX
Policy
ID: 5
has part
constraints
is enforced by
Maintain Y external copies of the experimental data
Policy
ID: 4
Split data to 20% parts
Process
Move data to new tapes
Process
Copy data to internal tape repository
Process
is used by
runs on
has output
has input
has QA
Web Interface Service Interface
Tape repository Technical Service
has part
Validate data checksums for external organisation slices
Quality Assurance Method
validates
provides access to
Send data to external organisations
Process
has input
20% slice of the data
Digital Object
Experimental data
Digital Object

Figure 42: Portion of a DEM diagram created by policy derivation and mapping

A simplified example of such a DEM constructed from policies is illustrated in Figure 42. The example starts with the generic policy ID 1 (step 1). This very generic policy is then specialised (step 2) for the experiment categories into non-confidential experiments (ID 2) and VSE experiments (ID 3). The policy ID 4 and 5 are a further specialisation of ID 2 and 3 and are at a lower level where entities of the DEM can be associated. Then processes and processed entities are identified and linked to the policies, a QA method and information about the infrastructure is added (step 3-6).

A graphical representation together with an accompanying description can form already a useful tool for describing the preservation processes and their requirements in an institution. Once this has been created, a formal DE description can be created, which allows to applicate external tools for an automated analysis and quality assurance.

10.1.1.1. Quality assurance

Quality assurance criteria are implemented as special policies, with the purpose of assuring the quality of normal policies. Following the policy based modelling strategy they are created directly after the policies, and linked to them. Quality Assurance can be defined also independently of a policy, to validate the functionality of other ecosystem entities.

QA 1: Checksums of data must be valid

This criterion assures the quality of data handling policies, especially policy 2 and 3 which define the data copy creation.

QA 2: The tape dust sensors must be in a valid state

This QA criterion assures that the internally preserved copies of the experimental data on tape storages won't be damaged by dust, by performing concrete checks on the dust that could be present on the tape system area and would damage the tapes. It is therefore linked to policy 5, which defines the maintenance of internal copies of all experimental data.

QA 3: Every year validate tape data checksums

This quality assuring policy is derived from QA 1 and also linked to policy 5 (policy 5 is derived from policy 2 and 3. The derivation and quality assurance relations are visualised in the following image:

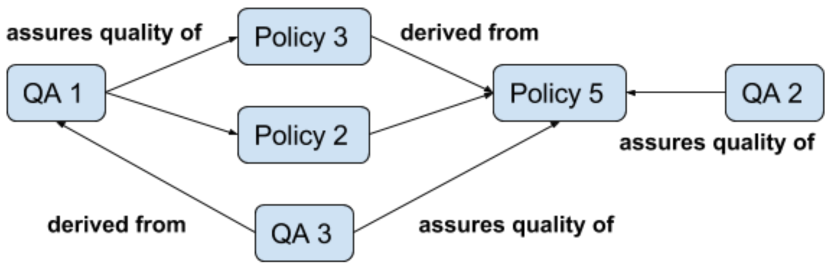


Figure 43: Derivation of the policies and QA relations

Quality assurance methods are specific processes, which implement the defined quality assurance rules. They are defined in the next step and linked to the policies and quality assurance criteria.

The quality assurance method

QAM 1: Validation of data checksums for external organisation slices

is an implementation of QA 1. Quality assurance methods can also be linked directly to normal policies, as the method

QAM 2: Search samples of confidential data in external institutions and on the internet

which is an implementation of the confidentiality policy.

At this point all digital ecosystem entities which are constrained by the policies are modelled, and linked to the policies and to the quality assuring entities. The most important digital object of the CERN example is the produced experimental data. An experiment process creates the raw data, which is filtered by a second process that discards invalid datasets based on filtering rules. The filtered data is moved from the experimental side to a computer on the data center where it is further processed. All versions of the experimental data are constrained by the digital ecosystem's policies, therefore the processes which process the data are designed to implement the policies, and the storages are constrained by the policies.

The software for the further processing of the experimental data is integrated into a virtual machine. An image for this virtual machine is created by a process, which requires digital objects describing the processing environment and information about the calibration of the data. It is versioned in a git repository on the department's server. A Jenkins continuous integration service is running on the same server. It is linked to

QAM 3: Continuous Integration

which assures the quality of the usability policy 6.

This policy defines the community of internal scientists and has the external scientists as main target community. It constraints the processing of experimental data in a way that it demands to take care of the usability of the processing results.

In the CERN example these are the technical services:

TS 1: A tape storage for the preservation of the experimental data

TS 2: The data center storage service (ERMR)

TS 3: The department's development server with the continuous integration services (Jenkins), and a versioning repository (GIT)

TS 4: The instruments needed to produce the experimental data

The tape storage, the data center storage and the instruments are related to the experimental data and underlie the policies handling experimental data.

As a next the step policy based modelling strategy suggests to model the digital ecosystems processes. They are linked to the policies that they implement. Digital ecosystem entities can be defined as input and output of the processes, it makes therefore sense to model the processes once the other entities are defined.

P1: Run experiment

P2: Discard invalid data based on algorithms in HW and SW

P3: Move data from experiment equipment to computer center

P4: Calibrate and process data

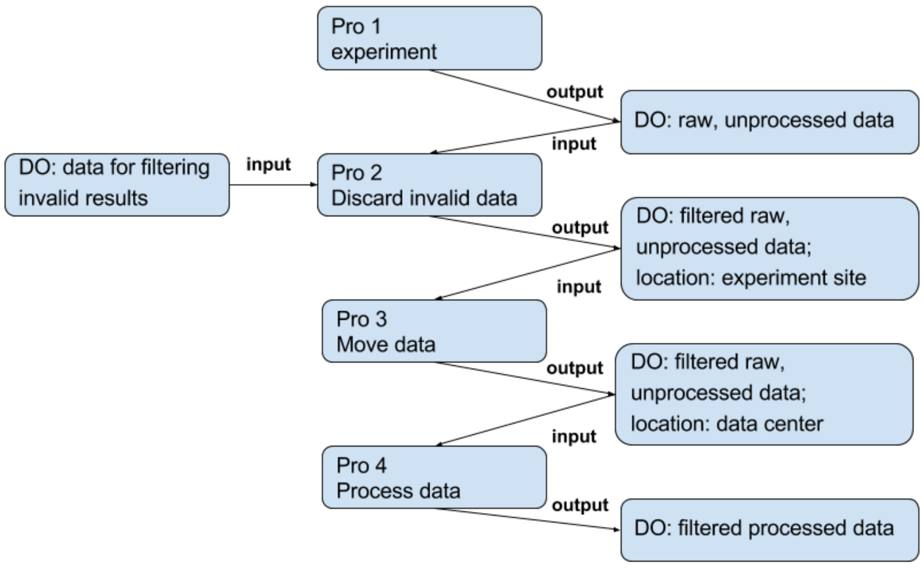
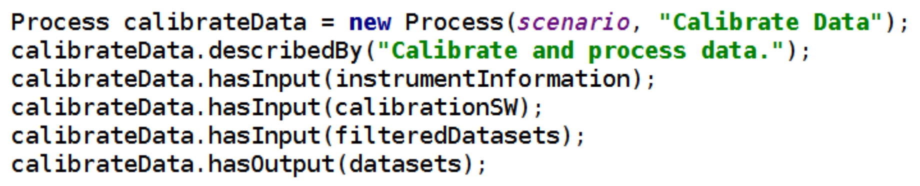

Pro 1
experiment
output
DO: raw, unprocessed data
input
DO: data for filtering invalid results
input
Pro 2
Discard invalid data
output
DO: filtered raw, unprocessed data; location: experiment site
input
Pro 3
Move data
output
DO: filtered raw, unprocessed data; location: data center
input
Pro 4
Process data
output
DO: filtered processed data

Figure 44: Processing of the experimental data



```
Process calibrateData = new Process(scenario, "Calibrate Data");
calibrateData.describedBy("Calibrate and process data.");
calibrateData.hasInput(instrumentInformation);
calibrateData.hasInput(calibrationSW);
calibrateData.hasInput(filteredDatasets);
calibrateData.hasOutput(datasets);
```

Figure 45: Implementation of the calibration process P4.

Preservation on internal tape repositories:

P5: Copy data to internal tape repository

P6: Manual: Move data to new tapes (every 3 years)

The processes in this example handle the experimental data to fulfil the requirements of the policies. The policy

POLICY ID 4:

“Maintain Y external copies of the experimental data”

for non-confidential data is implemented through the processes

P7: Split data in 20% parts

P8: Send data to external organisations

The data is splitted into parts for a better handling, whereby each data slice underlies the general policies for data handling, as:

POLICY: Files will be indexed with relevant metadata

which demands to store metadata together with each part. Also the usability policy, POLICY 6, has to be considered by the process implementation.

An image for the processing environment of the software which processes further and calibrates the filtered raw data is created by process 9 and needs calibration information and a description of the environment as input.

P9: Create virtual machines based on processing env. and software

The image is used to create the software agent which executes the calibration process. Further more detailed entities are modelled for the communities, human agents, and the technical infrastructure.

10.1.1.2. Entities for the analysis of the model

The last step introduces meta entities which support the analysis of the defined model. A scenario entity defines a subset of entities with the purpose of providing a designated view on the digital ecosystem. This reduces the calculation complexity for graph analysis and directs the view on the important entities for a designates aspect of the model. Scenarios are a useful tool for quality assurance, as they are complete aggregations of the quality assuring criteria and methods, the related policies, the constrained entities, and the meta entities describing their relations. A purpose entity describes the purpose of a scenario or of other entities, and significance entities allow to define how significant entities are for a designated purpose.

Two scenarios are defined for the CERN example:

S1: Processing of experimental data

S2: Preservation of experimental data

Both scenarios include all versions of the experimental data. S1 further includes the data processes and the policies which constrain data processing, whereas S2 includes the preservation related policies, processes, and storage infrastructure.

```
DEM-Scenario:PreservationScenario
        a             DEM-Analysis:Scenario ;
        rdfs:label    "Preservation Scenario"@en ;
        LRM:hasPart  DEM-Scenario:DataCenterStorage , DEM-Scenario:ExternalCopiesPolicy , DEM-
Scenario:TapeDustSensorsVerification , DEM-Scenario:RawDatasets , DEM-Scenario:InternalCopies , DEM-
Scenario:MetadataIndexingPolicy , DEM-Scenario:VerifyTapeDataCheckSums , DEM-Scenario:CheckSums , DEM-
Scenario:Preservation , DEM-Scenario:ValidRawDatasets , DEM-Scenario:TapeCatalog , DEM-
Scenario:Datasets , DEM-Scenario:InternalScientists , DEM-Scenario:TapeDustSensorsState , DEM-
Scenario:ChecksumCalculation , DEM-Scenario:InternalDataPreservation , DEM-
Scenario:TapeDataCheckSums , DEM-Scenario:ExternalDataCatalog , DEM-Scenario:CopyNumberCheck , DEM-
Scenario:TapeStorage , DEM-Scenario:PublicDataPreservation .
```

Figure 46: The preservation scenario provides a view on the subset of preservation related entities

Scenarios can also investigate rather small aspects of the model, as S3 which is the subset of entities related to the handling of confidential data, and S4 which is the view on the entities which are involved in ensuring the usability of experimental data:

S3: Handling of confidential data

S4: Use of experimental data

The purpose of S1 is to investigate and analyse the aspect of experimental data processing separated from entities which are not of relevance for this scenario, and to depict the data processing flow as a whole for a better understanding and problem solving.

S2 has the purpose to provide a designated view on the preservation aspects of the model, especially for the analysis of the compliance of the preservation policies, and the early identification and solution of preservation related problems.

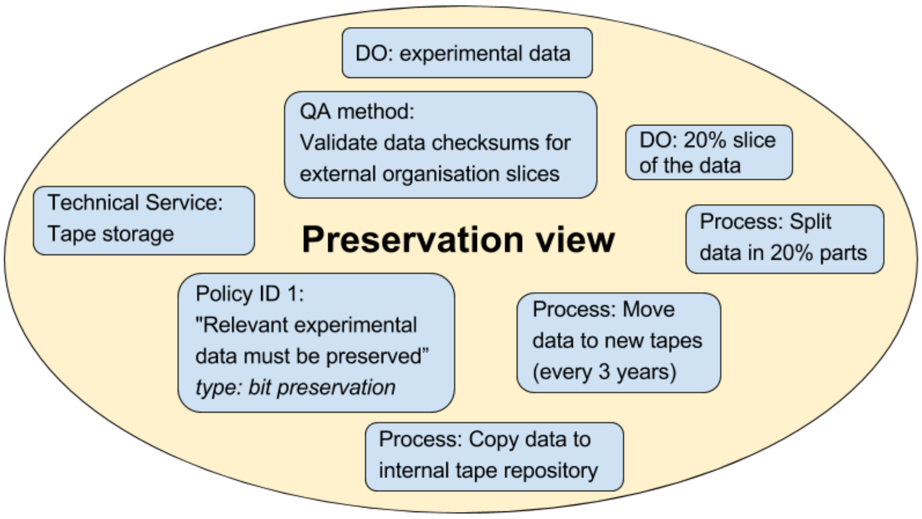

DO: experimental data
QA method: Validate data checksums for external organisation slices
DO: 20% slice of the data
Technical Service: Tape storage
Preservation view
Process: Split data in 20% parts
Policy ID 1: "Relevant experimental data must be preserved" type: bit preservation
Process: Move data to new tapes (every 3 years)
Process: Copy data to internal tape repository

Figure 47: Scenario 2 provides the preservation view on the CERN example model

Significance values and weighted relations are added to define how significant the entities and relations are for the scenario. The modeller is free to choose the weights depending on the planned analysis method. In this example weights are values between 1 and 0. For the preservation scenario the most important entities are the experimental data object and the high level preservation policy, which get a value of 1. The data splitting process and the data slices are a bit less important and would get a medium significance value of 0.6. The process to create the VM gets a low significance value of 0.2, because the preservation policies and workflows won't be immediately affected in case of an error of this process.

Annotations are quite similar to a Scenario regarding their ability to enrich entities with information for analysis, but in contrast to scenarios is their main purpose to add arbitrary information snippets to entities instead of defining aspect oriented subsets of the whole model. Two annotation are added to the example to specify if an entity is of relevance for internal procedures, or also for external scientists:

Annotation 1: @internal

Annotation 2: @external

A third one is used to annotate data which is used at the experiment, and a few more annotations add calibration parameter values directly to the instrument entity, and quality evaluation information to datasets:

Annotation 3: @experiment

Annotation 4: @parameter_A=true

Annotation 5: @quality=high

The resulting model can be analysed using static analysis to identify issues with the model or the underlying digital ecosystem. Furthermore, the model can serve to analyse planned or unplanned change of the evolving digital ecosystem. Triggers added to the model can be used to execute the QA methods and validate the state of the policies in case of specified events. In a semi-automated approach the user will be informed in case of identified errors.

11. Appendix: Related work to technical appraisal and risk management

Technical appraisal is often focused on characterisation of an object in its current state. However, a further dimension of appraisal is the effect of passing time: that is, the potential that events might occur in the future that limit the potential for the ongoing preservation of material. The DCC Digital Curation Manual identifies risk management as increasingly central to discussion of appraisal and selection (Harvey, 2007), permitting risks such as reduced accessibility, interpretability or ability to render material to be balanced against the consequences of that outcome. Traditional risk analysis is based on risk-impact (mitigation) analysis, for example as specified in (ISO/IEC 31010 Risk management – Risk assessment techniques, 2009). This is a process, usually iterative, in which the following sequence of steps is typically taken: identification of risks; assessment of the severity and potential consequences of those risks (such as financial consequences, impact on schedule or technical performance, and so forth); planning for mitigation; implementation of mitigating actions based on the plan developed. As risks evolve, they are tracked and documented.

The general-purpose project management methodology PRINCE2 (Bentley, 2010) specifies a series of steps in building and applying a risk management strategy. Risk management was brought into the forefront of preservation by the Cornell Library study into file format migration, reported in (Lawrence et al, 2000).

Many of the essential characteristics of a risk management toolkit were determined by PRISM (Kenney et al, 2002). Several existing risk management frameworks are explicitly intended to support preservation activities. These include DRAMBORA, the Digital Repository Audit Method Based on Risk Assessment (McHugh, Innocenti and Ross, 2008); TRAC (TRAC, 2007) which includes risk-oriented terms in a checklist of key terms; and the SPOT model (Vermaaten, Lavoie, & Caplan, 2012), which focuses on risks to essential properties of digital objects. Many of these approaches are qualitative and are a form of self-assessment, requiring the application of detailed technical knowledge.

Various tools are designed to support risk management in digital preservation planning, such as PLATO (Becker et al, 2008). Such tools are primarily reactive rather than predictive. That is, they can be used to detect events such as the discontinuation or a piece of software or format, but are not able to forecast such occurrences.

Within the art conservation community, the assessment of risk for a time-based media work of art was examined within the context of the Culture 2000 'Inside Installations' project (Scholte and Wharton, 2011). More generally within the Cultural Heritage field two methods have emerged for risk assessment, namely the Cultural Property Risk Assessment Method (CPRAM) and the ABC method. Both of these are based broadly on ISO/IEC 31010, and identify risks and their potential consequences systematically. Again they require subjective assessment of risks into e.g. high, medium and low levels. In (Brokerhof and Bülow, 2009), the authors point out that due to financial constraints and the sheer volume of digital material, it is no longer viable for heritage institutions to apply such manual techniques to digital material.

A considerable amount of recent research into risk analysis is available, much of which applies quantitative models in the forecasting of risk. Stamatelatos (2000) recommends the use of probabilistic risk analysis for the deconstruction and evaluation of risk associated with elements of complex entities. For the analysis of events that have occurred to ascertain the cause, fault tree analysis may be used; for the analysis of events yet to occur, event tree analysis may be used. Zheng (2011) provides a detailed analysis of risk modelling in order to support decision-making in management of product obsolescence, which may straightforwardly be adapted to the purposes of forecasting and managing software obsolescence. Risk analysis may use publicly available resources for informational purposes; for example, Graf and Gordea (2013) demonstrate the use of PRONOM, Freebase and DBpedia data to evaluate file format obsolescence. Registries of file format risks have also used quite widely. The Library of Congress's File Format Sustainability Factors (Library of Congress, 2016) are often used for risk assessment of file formats. These are manually assigned risk values on a scale from 1-9 assigned by an expert, rather than computed from empirical data. They also do not however reflect the proximity of the risks.