Project no. 269977

APARSEN
Alliance for Permanent Access to the Records of Science Network

Instrument: Network of Excellence

Thematic Priority: ICT 6-4.1 – Digital Libraries and Digital Preservation

D25.2 INTEROPERABILITY STRATEGIES

Document identifier:	**APARSEN-REP-D25_2-01-1_7**
Due Date:	2013-09-30
Submission Date:	2013-09-30
Work package:	WP25
Partners:	CSC, ESA, KNAW-DANS, KB, FORTH, FRD, UNITN
WP Lead Partner:	FORTH
Document status	Released
URN	urn:nbn:de:101-20140516189

Delivery Type	Report
Author(s)	As below
Approval	David Giaretta, Simon Lambert
Summary	The report describes a methodology for capturing, modelling, managing and exploiting various interoperability dependencies in digital preservation.
Keyword List	Interoperability, Dependency management, Performability, Migration, Emulation
Availability	☒ Public

Document Status Sheet

Issue	Date	Comment	Author
0.1	2012-06-01	Outline of the deliverable.	Y. Tzitzikas (FORTH)
0.2	2012-07-12	Extension of the sections that describe the rule-based approach.	Y. Tzitzikas, Y. Marketakis, Y. Kargakis (FORTH)
0.4	2012-09-03	Addition of the new publications (iPres12 and ISWC 12).	Y. Tzitzikas (FORTH)
0.5	2012-09-28	Addition of discussion about task-hierarchies.	Y. Tzitzikas (FORTH)
0.6	2012-11-11	Articulation with the rest APARSEN WPs and the deliverable D25.1. More about the RDF/S comparison.	Y. Tzitzikas (FORTH)
0.7	2013-04-26	Inclusion of material from the submitted D25.1. Sketch of the prototype which is under development.	Y. Tzitzikas, Y. Kargakis (FORTH)
0.8	2013-05-07	Integration of the results from D25.1 and other improvements. Initial review of the draft.	B. Bazzanella (UNITN)
0.9	2013-06-03	Additions and improvements based on the comments on v0.8. More information about the prototype.	Y. Tzitzikas, C. Lantzaki, Y. Kargakis (FORTH)
1.0	2013-07-05	More information about the prototype. Testing the prototype.	Y. Tzitzikas, Y. Kargakis (FORTH)
1.1	2013-08-23	Description of the DANS case study. Restructuring of the entire deliverable. Description of various options for the applicability of the approach. Enrichment of the concluding remarks.	Y. Tzitzikas, Y. Kargakis (FORTH)
1.2	2013-08-26	Improvements based on the feedback received from Jeffrey van der Hoeven.	Y. Tzitzikas (FORTH)
1.4	5/09/2013	Extra information about the contents of the Knowledge Base of the prototype, which is now called Epimenides. Improvements based on feedback received from Katrin Molch.	Y. Tzitzikas, Y. Kargakis (FORTH)
1.5	2013-09-23	Discussion of 5stars Linked Data from a dependency management point of view. Improvements and clarifications based on the comments from Kirnn Kaur.	Y. Tzitzikas, Y. Kargakis (FORTH)
1.6	2013-09-25	Addition of appendix with the results of the evaluation. Addition of examples at the beginning of the deliverable for making clear the results to the reader.	Y. Tzitzikas, Y. Kargakis, Y. Marketakis (FORTH)
1.7	2013-09-26	Final editorial and formatting amendments	S. Lambert (STFC)

Main Contributors

Partner	Persons	Main contribution
17. FORTH	Yannis Tzitzikas, Yannis Kargakis, Yannis Marketakis Christina Lantzaki	The technical part of the paper. The design and development of the prototype. Coordination of the activities for producing this deliverable.
25. UNITN	Barbara Bazzanella	Articulation with the results of the deliverable D25.1. Comments and suggestions on various versions of this deliverable.
12 DANS	Rene van Horik	Close collaboration for defining uses cases for DANS. Used and provided feedback about the prototype.
13. KB	Jeffrey van der Hoeven	Review and comments on the entire deliverable.
23. FRD	Emanuele Bellini	Used the prototype system.

Internal Reviewers

Person	Organization	Notes
Jeffrey van der Hoeven	KB (National Library of the Netherlands)	Member of the SCAPE project
Kirnn Kaur	British Library	

External Reviewers

Person	Organization	Notes
Jinsongdi Yu	Jacobs University, Bremen, Germany	Member of the SCIDIP-ES project.
Katrin Molch	Deutsches Zentrum für Luft- und Raumfahrt (DLR), German Aerospace Center, Earth Observation Center | German Remote Sensing Data Center | Information Technology | Oberpfaffenhofen, Germany	Member of the SCIDIP-ES project

Abstract: This deliverable analyses the main intelligibility objectives (as identified in D25.1) through a dependency point of view in order to propose a modelling approach that can automate task-performability checking and thus assist usability and preservation planning. Specifically, a methodology for capturing, modelling, managing and exploiting various interoperability dependencies is proposed.

Project information

Project acronym:	**APARSEN**
Project full title:	**Alliance for Permanent Access to the Records of Science Network**
Proposal/Contract no.:	**269977**

Project coordinator: Simon Lambert/David Giaretta

Address:	STFC, Rutherford Appleton Laboratory Chilton, Didcot, Oxon OX11 0QX, UK
Phone:	+44 1235 446235
Fax:	+44 1235 446362
Mobile:	+44 (0)7770326304
E-mail:	simon.lambert@stfc.ac.uk / david.giaretta@stfc.ac.uk

CONTENT

EXECUTIVE SUMMARY

Context

A quite general view of the digital preservation problem and its associated tasks (e.g. intelligibility and task-performability checking, risk detection, identification of missing resources for performing a task) is to approach it from a *dependency management* point of view. This deliverable will analyze the main intelligibility objectives (as identified in D25.1) through a dependency point of view in order to propose a modelling approach that can automate task-performability checking and thus assist usability and preservation planning.

Specifically, a methodology for capturing, modelling, managing and exploiting various interoperability dependencies is proposed. Firstly, we describe methods for modelling tasks and their dependencies (of conjunctive or disjunctive nature). Secondly, we elaborate on the reasoning services required and investigate technologies that can be used for realizing them. Since there may exist several methods to fill an intelligibility gap (if there are disjunctive dependencies), we will investigate which reasoning techniques can be exploited.

Advancement of the State-of-the-Art

In the context of this Work Package (WP), we have advanced past rule-based approaches of dependency management for modeling *converters* and *emulators* and we have demonstrated that this modeling enables more advanced digital preservation services. Specifically, these services can greatly reduce human effort required for periodically checking (monitoring) whether a task on a digital object is performable.

Difference with related work in the field of digital preservation:

- CASPAR project: this project has contributed to the formalization of the notions of intelligibility, modules and knowledge dependencies in a OAIS-compliant manner, introducing the idea that digital preservation means not just preserving streams of bits, but preserving information and knowledge (in particular by means of Semantic Representation Information), which makes digital resources intelligible and reusable in the long term. No implementation of the rule-based dependency management approach was undertaken.
- SCIDIP-ES project: this project is limited to implementing only the basic dependency management approach (not migration/emulation aware). It does not research migration/emulation techniques in the context of interoperability strategies.
- KEEP project: this project aims at developing emulation services to enable accurate rendering of both static and dynamic digital objects. The overall aim of the project is to facilitate universal access to cultural heritage resources. KEEP has created an Emulation Framework (EF) which provides additional services aiming at building a more solid ground for the emulation preservation strategy. KEEP is depending on existing and future emulators, and has not created an emulator itself. In comparison to the work done in KEEP, in this project we are not confined to software dependencies and emulation strategies only, but we elaborate on task dependencies in general, and we also consider migration/conversion strategies.

Results of this research:

In the context of APARSEN Task 2520, we:

- have advanced the dependency model with migration/emulation
- have created proof-of-concept datasets and services
- have advanced the methods for comparing RDF/S models (to better exploit blank node anonymity)
- have designed and implemented a proof of concept prototype which is web accessible

Apart from this deliverable, the following outcomes of this research have been peer-reviewed and published:

- the new dependency model and services were described in a paper accepted at iPRES 2012 (paper Tzitzikas, Marketakis & Kargakis, 2012)
- the methods for comparing RDF/S models were described in a full paper accepted at ISWC (International Semantic Web Conference) (Tzitzikas, Lantzaki & Zeginis, 2012) and a demonstration of this paper

1 INTRODUCTION

This section contains a description of the deliverable and related tasks as originally defined in the Description of Work (DoW) of the APARSEN project

1.1 DESCRIPTION OF THE DELIVERABLE

D25.2: Interoperability strategies:

This deliverable will analyze the main intelligibility objectives (identified in D25.1) through a dependency point of view for proposing a modeling approach that can automate task-performability checking and thus assist usability and preservation planning. Specifically, it will propose a methodology for capturing, modeling, managing and exploiting various interoperability dependencies. At first it will describe methods for modeling tasks and their dependencies which can have conjunctive or disjunctive nature. Then it will elaborate on the reasoning services required and it will investigate technologies that can be used for realizing them. Since there may exist several methods to fill an intelligibility gap (if there are disjunctive dependencies), we will investigate which reasoning techniques can be exploited.

This deliverable is related to the following task, also defined in the DoW of the APARSEN project:

Task 2520: Intelligibility Modeling and Reasoning

Each interoperability objective/challenge, like those that will be collected in T2510, is a kind of demand for the performability of a particular task (or tasks). In this task (T2520) we will identify such tasks, we will reflect on their dependencies and on how these can be modeled. The objective is to propose a modeling approach that enables the desired reasoning, e.g. task performability checking, which in turn could greatly reduce the human effort required for periodically checking or monitoring whether a task on an archived digital object or collections performable, and consequently whether an interoperability objective is achievable. Such services could also assist preservation planning, especially if converters and emulators can be modeled and exploited by the dependency services. Finally, we will propose technologies for implementing the proposed modeling approach and we will report results and recommendations.

1.2 METHODOLOGY

The outcomes were based on both new insights and progress within the APARSEN project as well as independent research of partner FORTH-ICS. This deliverable was produced by the following main steps (not in linear order).

No	Activities	Dependencies
1.	Provision of contact persons from the participating partners	-
2.	Monthly plan decided in the monthly teleconferences	D25.1
3.	Independent research by FORTH-ICS (on the required knowledge management techniques for tackling the requirements of this task)	-
4.	Exploitation of past deliverables of APARSEN (and other sources)	-
5.	Coordination with the activities related to D25.1 (since both belong, to the same work package), this is explained in the next subsection	D25.1
6.	Collection of objectives and ideas from the participating partners	D25.1
7.	Preparation of the first draft of the deliverable	3
8.	Feedback from the partners	7
9.	Design and implementation of the prototype system	3

10.	Feedback from selected partners from using the system	8
11.	Preparation of the revised version of the deliverable	9
12.	Request feedback from organizations that do not participate to APARSEN	10
13.	Preparation of the final version of the deliverable.	11

Table 1 Steps for deriving the current deliverable

As regards step 12 (feedback from persons who do not participate to APARSEN), we have so far received feedback from:

- Jinsongdi Yu (Jacobs University, Bremen, Germany), member of the SCIDIP-ES project
- Katrin Molch (DLR), member of the SCIDIP-ES project

The general methodology of the work related to WP25 is shown in Figure 1.

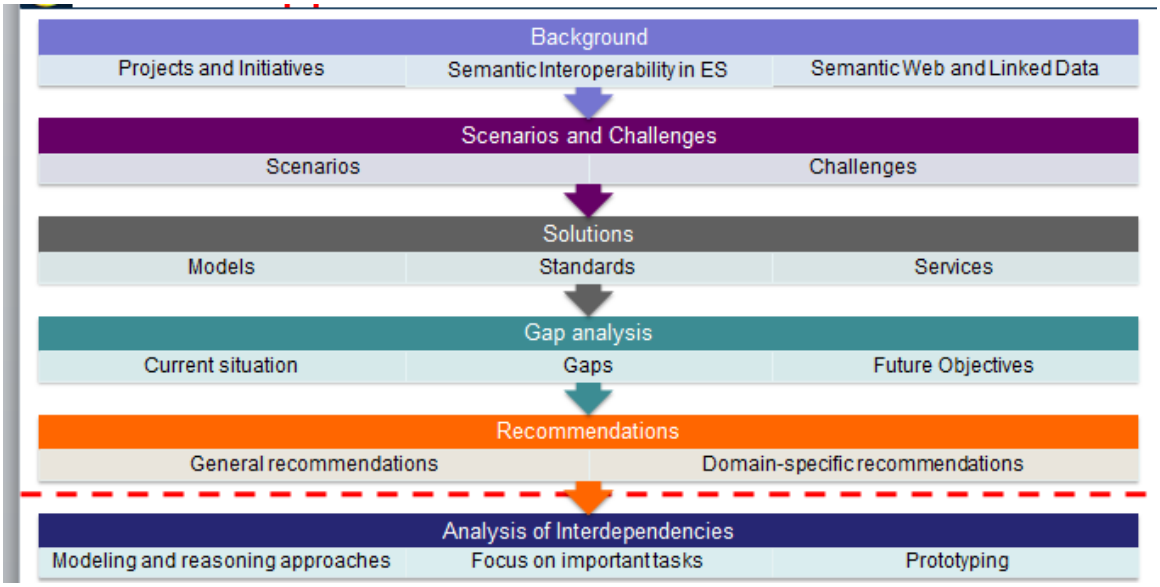


Figure 1 Methodology of work on Interoperability and Intelligibility (APARSEN WP25)

Task 2520 and this deliverable (D25.2) correspond to the step "Analysis of Interdependencies" of Figure 1, while the 5 phases before the dotted line have been addressed by Tasks 2510 an 2530 as reported in D25.1, the results of which feed into this research. The connection between the current deliverable (D25.2) and the completed D25.1 can be summarized as:

Each interoperability objective/challenge, like those described in D25.1, can be conceived as a kind of demand for the performability of a particular task (or tasks). In the current deliverable, we attempt to identify such tasks and to reflect on their dependencies. Then we show how these dependencies can be modeled and managed for enabling services which can reduce the human effort required for periodically checking (monitoring) whether a task on a digital object is performable.

1.3 DOCUMENT OUTLINE

The rest of this report is organized as follows:

Section 2 provides an introduction to interoperability and summarizes the results of the APARSEN deliverable D25.1 (interoperability objectives and approaches) which has been delivered at the beginning of 2013.

Section 3 introduces two main interoperability strategies.

Section 4 describes how interoperability is related to dependency management.

Section 0 discusses the requirements concerning automated reasoning.

Section 0 describes the methodology for applying the dependency management approach.

Section 0 focuses on the required conceptual modeling (this section is technical).

Section 0 discusses which tasks are worth modeling, how they can be organized, and discusses related activities (e.g. Linked Data).

Section 9 discusses technologies that could be used for implementing the approach (this section is technical).

Section 0 describes datasets that we have used for testing.

Section 0 elaborates on more refined gaps.

Section 12 describes the prototype that we have designed and implemented for proving the technical feasibility of the proposed approach.

Section 0 describes particular use cases from partners and discusses the applicability of the proposed approach.

Section 14 concludes the deliverable.

Auxiliary material is given in the Appendix.

Note: *Some sections are technical (mainly Section 0 and Section 9).*

2 INTEROPERABILITY

2.1 INTRODUCTION

According to the IEEE[1] definition interoperability refers to "the ability of two or more systems or components to exchange information and to use the information that has been exchanged". Various aspects or layers of interoperability have been identified, which occur at different levels of abstraction, mainly:

Syntactic interoperability

If two or more systems are capable of communicating and exchanging data, they are exhibiting syntactic interoperability, which is required for any attempts of further interoperability. While syntactic interoperability is usually associated with data formats, the term technical interoperability is often used to refer to the level of interoperability enabled by communication protocols and the infrastructure needed for those protocols to operate. According to this distinction, technical interoperability only guarantees the correct transmission of data between interoperating systems, while syntactic interoperability provides more structuring of the content through transfer syntaxes.

For instance, XML or SQL standards provide syntactic interoperability. This is also true for lower-level data formats, such as ensuring that alphabetical characters are stored in ASCII format in both of the communicating systems.

Semantic interoperability

Beyond the ability of two or more computer systems to exchange information, semantic interoperability is the ability to automatically interpret the information exchanged meaningfully and accurately in order to produce useful results as defined by the end users of both systems. To achieve semantic interoperability, both sides must defer to a common information exchange reference model.

The content of the information exchange requests should be unambiguously defined: what is sent is the same as what is understood.

Besides technical, syntactic and semantic interoperability, other higher levels of interoperability have been proposed including organizational, legal and cultural aspects. The European Interoperability Framework[2], for example, identifies 3 levels of interoperability: technical, semantic and organizational. Organizational interoperability is concerned with organizational aspects (e.g. business goals and processes) that are in play when different organizations, which may have different internal structures and processes, exchange information and work together. Organizational interoperability is considered a key step to move from isolated information systems toward a global information digital space and also addresses the requirements of users by making services available, accessible and easy to use.

Another aspect of interoperability (i.e. legal interoperability), that deals with legal requirements and implications involved in making resources more widely available. This includes, for example, access restrictions, rights management and licenses. Apart from legal issues, which may arise when different organizations and systems cooperate in a common information space, there are also issues regarding community and disciplinary boundaries across which resources (and particularly scientific resources) need to be shared. These issues pertain to the layer of interoperability, which has been called by Miller (Miller, 2000) Inter-community interoperability.

A more detailed discussion about the layers of interoperability proposed in literature can be found in D25.1.(Section 2). For the purpose of the current document, it is sufficient to convey the message that interoperability is a very complex and multifaceted concept encompassing not only technical aspects but also political, economic and social dimensions which introduce a lot of interoperability dependencies to be managed.

[1] Institute of Electrical and Electronics Engineers (www.ieee.org)

[2] http://ec.europa.eu/idabc/servlets/Docd552.pdf?id=19529

The first deliverable of WP25, namely D25.1[3] "Interoperability Objectives and Approaches" (March 2013) provided:

1. An *analysis of interoperability issues* in digital preservation.
2. An *overview* of **64 projects** and **initiatives** in different areas of digital preservation, including specific domains like Earth Science and Linked Data.
3. A discussion of *global semantic Interoperability* enabled by ***Semantic Web initiative*** *and* ***Linked Data*** (strengths and weaknesses).
4. An evaluation of **13 interoperability scenarios** and related challenges, encountered by partners and other stakeholders in their daily life activity.
5. A design of a matrix of models, standards and services for interoperability (**58 rows**) by Digital Preservation (DP) areas, interoperability objects and challenges.
6. A *gap analysis* to diagnose the landscape and identify future actions and goals.
7. A set of *recommendations* and *guidelines* for decision makers and practitioners.

The following paragraphs give some brief recap of these outcomes. Please see D25.1 for a complete overview.

The methodology followed for deriving D25.1 which focused on interoperability **approaches** and **solutions** is shown in Figure 2. The figure highlights that an all-encompassing perspective was taken, including technical, social, political, organizational and many other factors which have an impact on different areas of DP, to provide a comprehensive picture of this complex multi layered landscape and enhance the understanding of its faces and orienting strategies for finding specific solutions. The Figure shows also 1) the state of the art analysis of on-going and past projects and initiatives on interoperability in DP, 2) the collection of scenarios and challenges by partners and finally 3) the analysis of interoperability models, standards and services, served to perform the gap analysis and derive the definition of the main interoperability objectives and guidelines to fill the identified gaps. The work described in the deliverable in D25.1 aimed to provide the context for the activities within Task 2520, by identifying a number of interoperability dependencies, which should be modelled and addressed by the methodology proposed in this document.

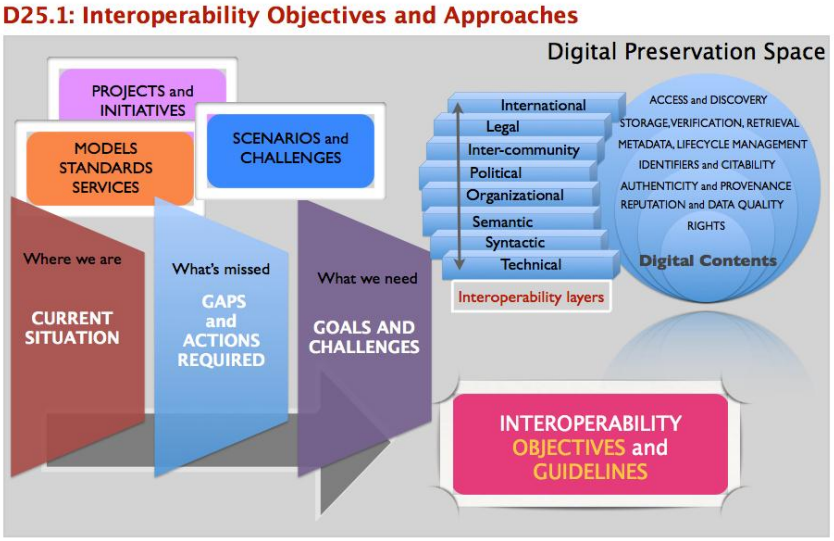


Figure 2 Methodology used for deriving D25.1

[3] http://www.alliancepermanentaccess.org/wp-content/uploads/downloads/2013/03/APARSEN-REP-D25_1-01-1_7.pdf

2.1.1 Map of Projects and Initiatives

To understand what already has been undertaken to improve interoperability between different systems and concepts, an overview of all ongoing and past projects was created covering interoperability issues related to (or relevant for) digital preservation. Information from 64 projects and initiatives were clustered around eight areas:

1) Digital Preservation Conceptual Models and Interoperability Frameworks
2) Data Infrastructures for e-Science
3) Digital Libraries
4) Open Repositories
5) Persistent Identifiers
6) Semantic Interoperability and Linked Data
7) Semantic Access to Earth Sciences Resources and
8) Other

Each project or initiative has been described by name, domain (i.e. the area to which the project or initiative belongs), timescale (i.e. the duration of the project or initiative), general description (including objectives and issues addressed), specific interoperability objectives and external references for further information.

The gathered interoperability related **projects** and **initiatives** are shown in Figure 3.

2.1.2 Map of Solutions

Having analysed the existing projects and initiatives, the interoperability issues were identified. The final aim was to describe (a) which are the critical interoperability aspects pertaining a certain area of digital preservation, (b) which main layers of interoperability are mainly involved, (c) which are the interoperability objects that are implicated and finally which concrete solutions (e.g. models, standards) have been adopted to address these issues.

The result of the analysis led to define a matrix, which combines different layers of interoperability (e.g. syntactic, semantic, organizational) with the areas of digital preservation (e.g. persistent identifiers, metadata, provenance) and the related interoperability objects and models, providing an interoperability conceptual framework for digital preservation that can be used as a starting point to facilitate practical interoperability solutions and design concrete interoperability services for long-term preservation. The proposed framework includes the following categories:

1) Digital Preservation area: indicates the area of digital preservation where interoperability takes place. Examples are preservation services, persistent identifiers, authenticity and provenance.
2) Interoperability issue/challenge: a problem of interoperability which hinders a certain task or process in an interoperability context.
3) Interoperability objects: the entities that actually need to be processed in interoperability scenarios. They can include, for example, the full content of digital resources or mere representations of such resources (i.e. metadata, identifiers).
4) Adopted solutions/models/standards: those approaches, which are adopted to address specific interoperability issues/challenges at different levels.

Based on this framework we collected information about 58 interoperability solutions shown in Figure 4.
These solutions have been clustered around eight categories identified by colours in Figure 4:

1) Persistent Identifiers,
2) Provenance,
3) Data Quality,
4) Metadata,
5) Metadata Harvesting and Information Exchange,

6) Authentication, Authorization, Rights,
7) Preservation Models and Services,
8) Research data deposit, discovery, access, reuse and citation.

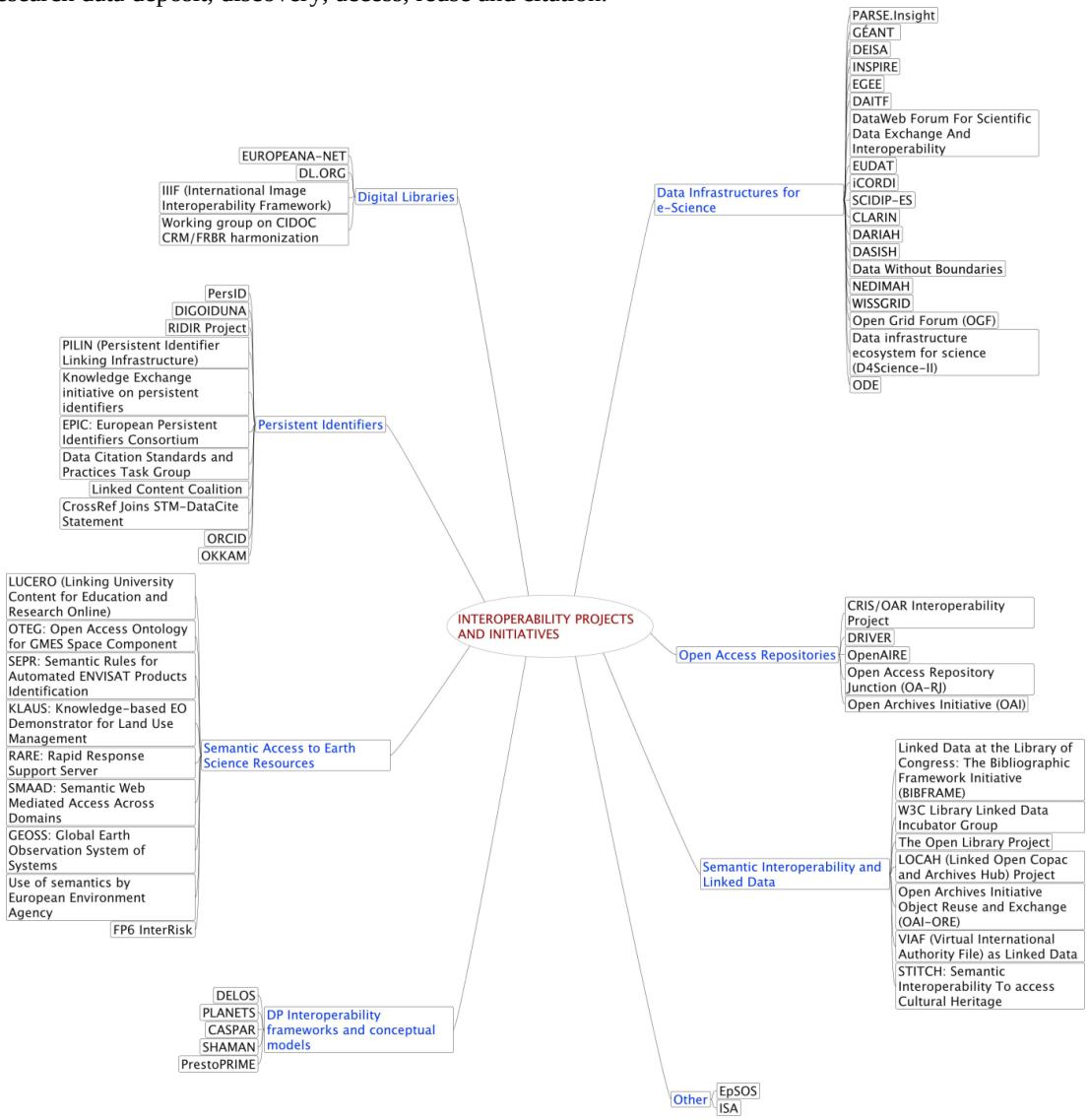


Figure 3 Projects and Initiatives

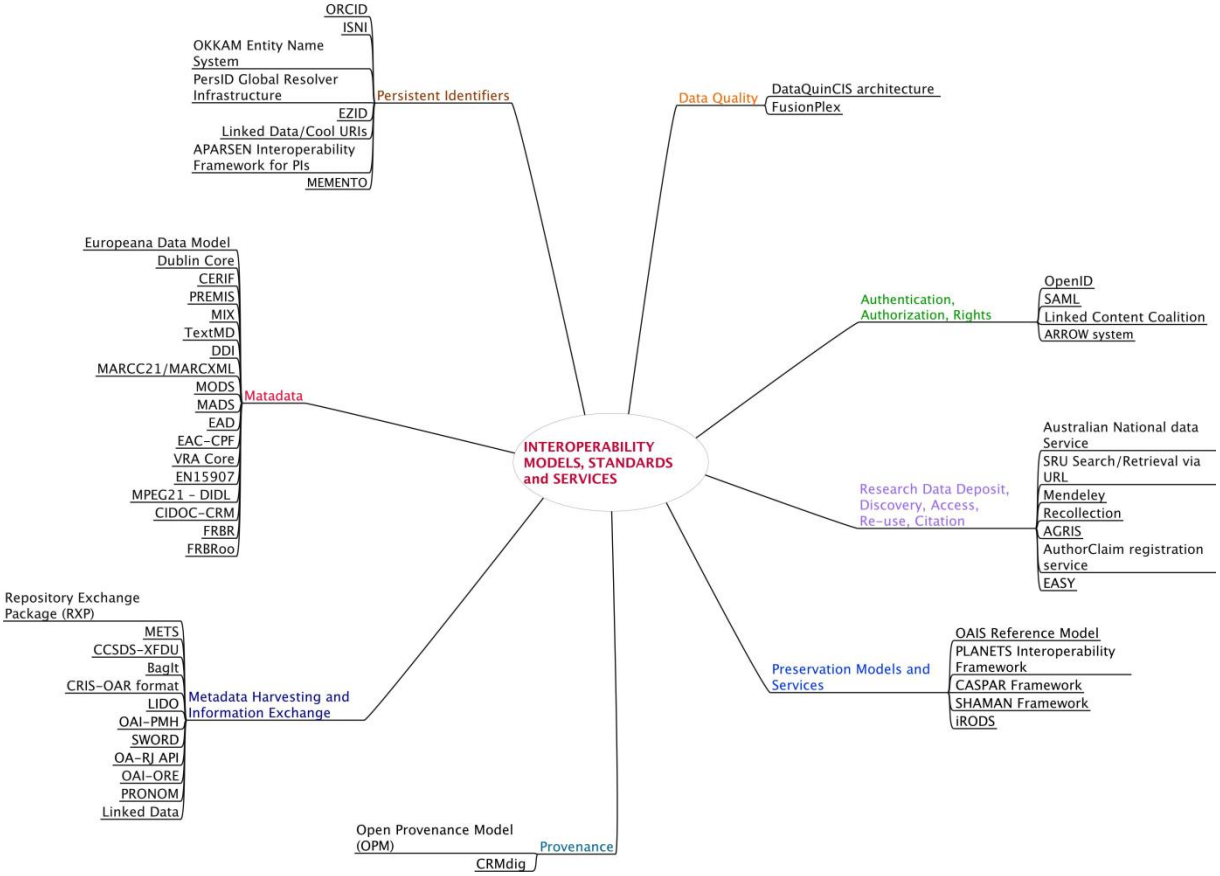


Figure 4 Interoperability solutions

Actually, the areas shown in Figure 4 are not independent or orthogonal, but connected in several ways. Their main relationships are depicted in Figure 5 indicating that a solution that has been associated to a certain area (e.g. PIDs) could be indirectly associated to another area (e.g. Provenance) due to the dependencies between them.

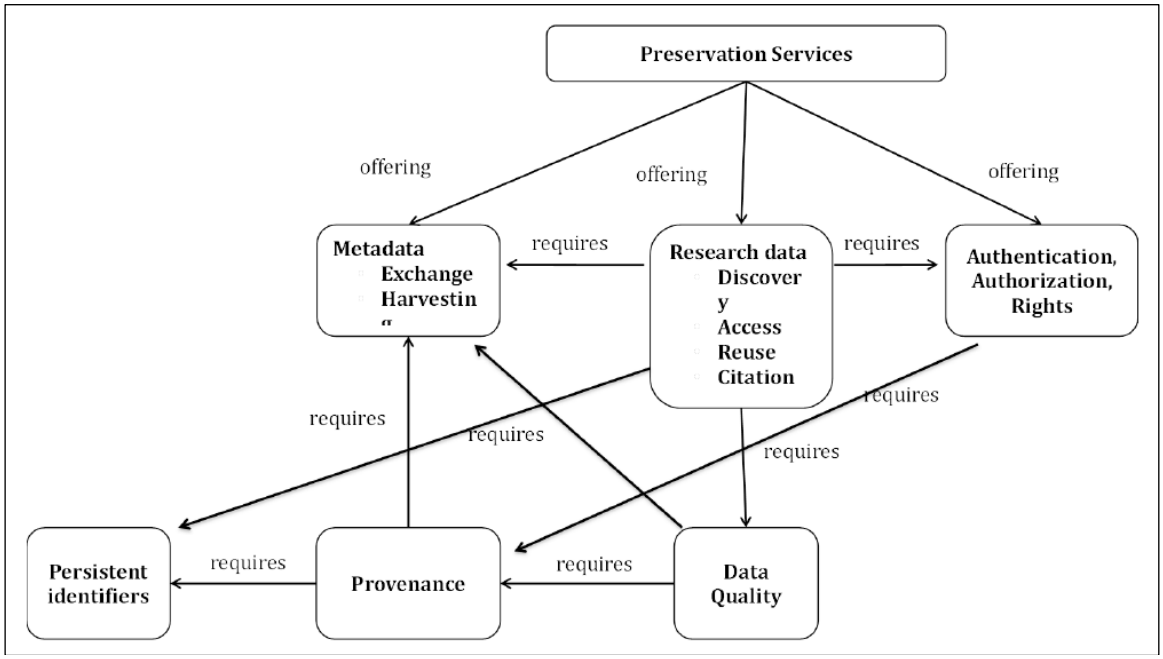


Figure 5 Main relationships between the DP areas

2.1.3 Gaps of Interoperability

In D25.1 a number of gaps between the current situation (where we are) and the desirable interoperability objectives for the future (where we want to be) were identified and possible approaches for tackling these gaps (how to fill the gaps) have been proposed.
We briefly discuss here some of the gaps identified by this analysis, referring to D25.1 for a complete description.
A first group of identified gaps deals with aspects concerning identification solutions for digital objects, authors and datasets. As an example, there is a gap between the current fragmentation of the PID landscape of solutions and the need of a unique entry point to make different PIDs interoperable allowing the long-term access to distributed sources and integrated information. In Figure 6 other gaps in the use of identification solutions are shown, including possible strategies to fill these gaps.
Another set of gaps concerns Linked Data in the domain of libraries. Currently there are many issues which hinder the full potential of Library Linked Data and the development of related valuable services, such as the heterogeneity of metadata standards, the lack of mapping across vocabularies, the lack of cross-lingual mechanisms to link resources across libraries, the lack of end-users services based on Linked Data.
Other gaps have been identified in the use of metadata to support representation and management of digital resources. In this context, the diversity of metadata standards, the existence of local schemas and the heterogeneity in metadata usage and implementation have significant implications for the development of services to access information resources shared across system and organization boundaries.

The lack of coordination in the definition of a common preservation metadata framework and the development of local solutions for metadata standards supporting preservation strategies represent another gap in the domain of metadata for DP.
Apart from preservation metadata, a further gap which hinders the development of effective strategies and solutions for DP is the lack of cross-boundary preservation infrastructures and large-scale preservation frameworks which allow local preservation systems to interoperate across organizational, national and system boundaries.

Gaps have also been identified at the level of ontological representation, vocabularies and categorization of digital resources. Among these gaps we can mention the lack of automatic mapping between different library classification systems and ontologies in specific domains (e.g. earth Science), the lack of controlled vocabularies to discover information and the lack of solutions for mapping application domain thesauri/ontologies and resources.

E-Science infrastructures represent another domain where several gaps have been diagnosed. We identified, for example, a strong need of defining a common international framework for data infrastructure for e-Science in order to overcome the current fragmentation and the lack of global coordination. Another gap is between the development of research information services on a national scale to access national scholarly information and the need of a unique entry point at international level to the research data managed by these systems.

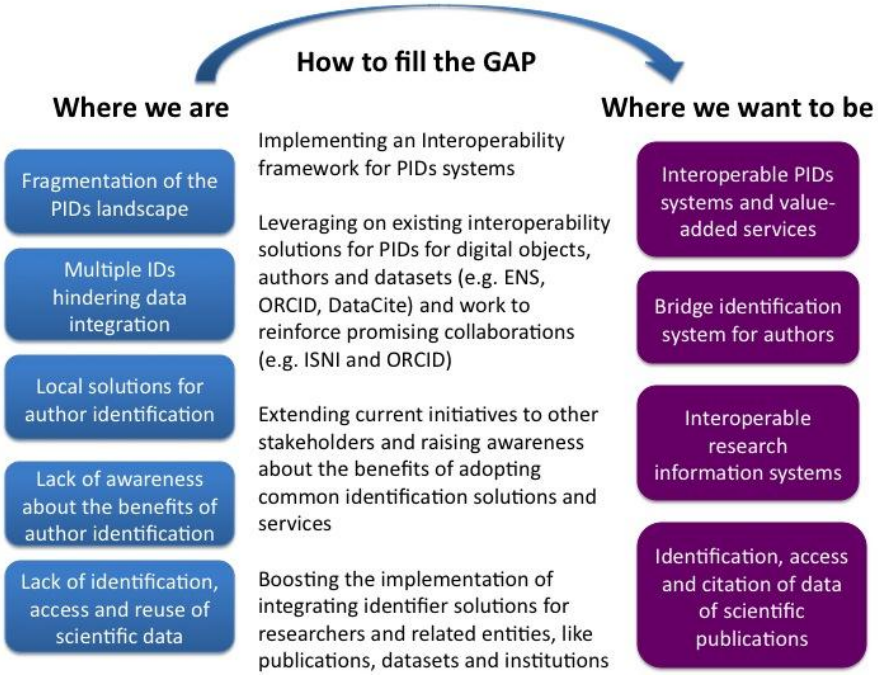


Figure 6 An extract from the gap analysis results in the domain of PIDs

2.1.4 Recommendations

In D25.1 a number of general, as well as domain-specific, recommendations have been proposed.
General recommendations are applicable to all the categories of stakeholders in DP and aim at:

- Fostering the broad adoption of common standards and specifications reducing dependencies, facilitating the interoperation between systems for the entire digital object lifecycle management process and enabling higher-level services on top of standard compliant systems.

- Promoting the use of appropriate identification systems (and their interoperability) for digital and non-digital objects and the adoption of methods for unique identification, which are independent from the kind of carrier or specific encoding of the information object.
- Promoting the convergence toward common policies and governance models, which favour the adoption of interoperability solutions and trust on them.
- Ensuring the necessary long-term financial support and the efficient use of economic resources and appropriate incentives to foster good interoperability practises and solutions.
- Raising awareness and securing trust among the involved stakeholder communities, encouraging coordination and collaboration among them and aligning interests and responsibilities into sustainable strategies and business models.

Domain-specific recommendations include two sets of recommendations:

1) Recommendations to support interoperability for Persistent Identifiers aiming at:
   - Promoting the definition of a common set of objectives and requirements among key stakeholders toward the design and implementation of an interoperability infrastructure for Persistent Identifiers.
   - Defining and sharing a high-level framework for interoperability between PID systems.
   - Building a community behind the interoperability infrastructure to agree on a shared governance model, which devolves responsibilities among the involved parties, and define common trust criteria.
   - Promoting awareness and skills development to enable different stakeholders to participate effectively on PIDs initiatives and infrastructures.
   - Promoting cross-fertilization between public and private sectors to co-operate in the implementation of added value services on top of interoperability solutions.
   - Fostering interoperability based on consolidation of trusted and established PIDs systems, like DOI, URN, ARK, instead of promoting the proliferation of ad-hoc local systems.
   - Building sustainable business models to guarantee the long-term sustainability of interoperability solutions.
   - Considering new types of interaction with and between structured data offered by the emerging Linked Data approach and investigate whether PIDs and Linked Data can be integrated to create a new class of persistent interoperable cool URIs.

2) Recommendations to support semantic interoperability in the domain of Earth Sciences which aim at:
   - Involving the end-user community in the definition of the ontology and mediators for enabling the friendly discovery of Earth Science resources, including:
     - Definition of dictionaries, starting from the state-of-the-art and real needs of such a heterogeneous community
     - Definition of thesauri, which provide the semantically linked terms, collected within the appropriate dictionary
     - Definition of ontology mediator, as needed, to semantically link available thesauri
     - Support in the standardisation activity concerning semantic representation language (e.g.: OWL, SKOS, etc…)
   - Making the Earth Observation (EO) resource provider community to lead the definition and implementation of the catalogue technology facilities, including standardisation activities needed to:
     - Seamlessly federate EO resources, sharing common Identity Management Services, Discovery Services, Invoke Services and Online Data.
     - Semantically annotate EO resources metadata, permitting to easily link ontologies to resource catalogue metadata.

3 INTEROPERABILITY STRATEGIES

Based on the analysis done previously, two main strategies on interoperability can be identified:

S1: Reliance to Standards. One general approach to tackle the interoperability problem is standardization, i.e. to achieve one interoperability objective; one strategy is to use standards appropriately for that. Standards grouped by areas were given in D25.1 (see also other related work[4]).

S2: Live without Standards. A rising question is whether we could tackle the interoperability problem without having to rely on several and possibly discrepant standards. Can we come up with processes that can aid solving the interoperability problem in a flexible manner? What kind of model/services could support that? One method to approach this question is elaborated in the current deliverable.

A very short example of S2, i.e. of the possibility of living without standards, follows:

Situation: "*One user wants to run an Amstrad 464 program (written in 1986) on his/her android smart phone which runs a 2012 version of Android OS*.

Approach: We do not necessarily need dedicated standards for realizing the above scenario. A series of *conversions* and *emulations* could make feasible the execution of the 1986 software on a 2012 platform. However, the process of checking whether this is feasible or not could be too complex for a human. This is the point where advanced modeling and automatic reasoning services could contribute.

Below we attempt to pass the main message by discussing in more detail this example. Consider a user, who would like to run on his mobile, software source code written before many years. E.g. software code written in Pascal programming language and stored in a file named game.pas

Questions:

- What can he do? (to achieve his objective)
- What should we (as community) do?
  - Do we have to develop a Pascal compiler for Android OS?
  - Do we have to standardize programming languages?
  - Do we have to standardize operating systems, virtual machines, etc?
  - .. and so on.

Direction and Answer (according to Task 2520 and the current deliverable)

- It is worth investigating if it is already possible to run it on android by "combining" existing software!
  - How? By applying a series of transformations and emulations

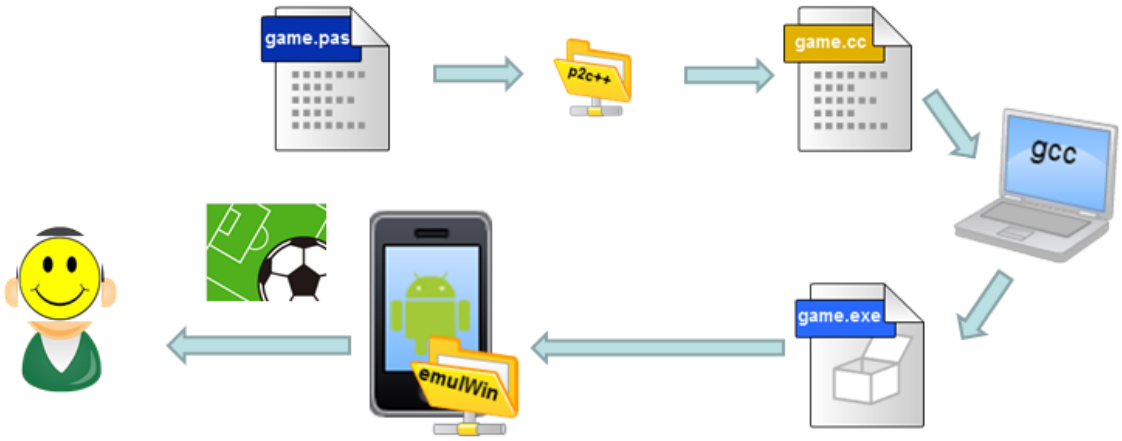
To continue this example, suppose that we have in our disposal only the following:

[4] APARSEN WP13 systematically collected and entered this kind of information into a database, see D13.1 for details). Moreover we should mention that the ongoing WP35 focuses on policies which again are related to interoperability.

- a *converter* from **Pascal** to **C++** (say **p2c++**),
- a **C++ compiler** (**gcc**) for WindowsOS,
- an *emulator* of WinOS executables over Android OS (say **emulWin).**

Someone could think: *Well, it seems that we could run **game.pas** on his mobile phone by first converting the Pascal code to C++ code, then compiling the C++ code to produce executable code, and finally by running over the emulator the executable yielded by the compilation.*

Indeed, the following series of transformation/emulations could achieve our objective:



One might argue that this is very complex for humans. Indeed this is true. We believe that such reasoning should be done by computers, not humans. The work that we present in the current deliverable shows how we can model our information in a way that enables this kind of automated reasoning.

The above scenario concerns *software*. We should however clarify that the proposed approach is not confined to software. Various interoperability objectives that concern *documents* and *datasets* can also be captured.

- For example for the case where a user wants to render a MSOffice *document* on his smart phone, the reasoning approach can infer that this is possible through various ways (e.g. by running the SuiteOffice on his smart phone, or by running MicrosoftOfficeWord.exe over an emulator, or by converting the document to PDF, etc).
- For the case of *datasets*, consider that we want to preserve datasets containing experimental results and would like to preserve their provenance. Suppose that for us provenance means ability to answer questions of the form: who derived the dataset, when this dataset was derived, how it was derived? We can model provenance as a task (that has dependencies) and we can use the dependency reasoning approach for checking for which datasets we have provenance and for which we have not. We could also exploit the reasoning services in order to discover provenance information that was not evident (e.g. result of tools that extract embedded metadata).

4 DEPENDENCY MANAGEMENT AND INTEROPERABILITY

4.1 HOW INTEROPERABILITY IS RELATED TO DEPENDENCY MANAGEMENT

The crux of the interoperability problem is that digital objects and services have various **dependencies** (syntactic, semantic, etc). We cannot achieve interoperability when the involved parties are not aware of the dependencies of the exchanged artifacts. One general approach to tackle this problem is *standardization*. From the dependency point of view, standardization essentially reduces the dependencies or makes them widely known (it does not vanish dependencies). Apart from developing standards, it is worth investigating more flexible methods for tackling the interoperability problem. A rising question is whether we could tackle the interoperability problem without having to rely on several and possibly discrepant standards. It is worth investigating whether we could establish a protocol for aiding interoperability on the basis of **explicit dependency management**.

4.2 INTRODUCTION TO DEPENDENCY MANAGEMENT FOR DIGITAL PRESERVATION

In digital preservation there is a need for services that help archivists in checking whether the archived digital artifacts remain *intelligible* and *functional*, and in identifying the consequences of probable losses (obsolescence risks). To tackle the aforementioned requirements, (Tzitzikas, 2007) showed how the needed services can be reduced to *dependency management* services, and how a semantic registry (compatible with OAIS) can be used for offering a plethora of curation services[5]. Subsequently, (Tzitzikas & Flouris, 2007) extended that model with *disjunctive dependencies*. The key notions of these works are the notion of *module*, *dependency* and *profile*. In a nutshell, a *module* can be a software/hardware component or even a Knowledge Base expressed either formally or informally, explicitly or tacitly, that we want to preserve. A module may require the availability of other modules in order to function, be understood or managed. We can denote such *dependency relationships* as $t > t'$, meaning that module t depends on module t'. A *profile* is the set of modules that are assumed to be known (available or intelligible) by a user (or community of users), and this notion allows controlling the number of dependencies that have to be recorded formally (or packaged in the context of an *encapsulation preservation strategy*). Subsequently, and since there is no objective method to specify exactly which are the dependencies of a particular digital object, (Marketakis & Tzitzikas, 2009) extended the model with *task-based* dependencies where the notion of task is used for determining the dependencies of an object.

As **tasks** we define *actions* that can be applied on a digital object (e.g. edit, render or run a digital object).

That work actually introduced an extensible *object-oriented* modeling of dependency graphs expressed in Semantic Web (SW) languages (RDF/S). Based on that model, a number of services have been defined for checking whether a module is *intelligible* by a community (or for computing the corresponding *intelligibility gap*), or for checking the *performability of a task*. These dependency management services were realized over the available SW query languages. For instance, GapMgr and PreScan (Marketakis, Tzanakis & Tzitzikas, 2009) are two systems that have been developed based on this model, and have been applied successfully in the context of the EU project CASPAR. Subsequently, (Tzitzikas, Marketakis & Antoniou, 2010) introduced a *rule-based* model which also supports task-based dependencies, and (a) simplifies the disjunctive dependencies of (Tzitzikas & Flouris, 2007), and (b) is more expressive and flexible than (Marketakis & Tzitzikas, 2009), as it allows expressing the various properties of dependencies (e.g. transitivity, symmetry) straightforwardly. That work actually reduced the problem of dependency management to *Datalog*-based modeling and query answering.

However, the aforementioned works did not capture *converters* and *emulators*. Since conversion (or migration)[6] and emulation[7] are quite important preservation strategies, a dependency management approach should allow

[5] For more about preservation services, the reader can see the deliverables of APARSEN WP21 (Preservation Services).

modeling explicitly converters and emulators (and analyze them from a dependency point of view, since they have to be preserved too), and exploit them during the offered preservation services (that means, according to a dependency management approach, exploiting them to identify intelligibility gaps and reducing these gaps). For example, a sequence of conversions can be enough to fill an intelligibility gap, or for allowing performing a task. Since there is a plethora of emulation and migration approaches that concern various layers of a computer system (from hardware to software) or various source/target formats (e.g. see (Giaretta, 2010) for an overview), it is beneficial to use advanced knowledge management techniques for aiding the exploitation of all possibilities that the existing and emerging emulators/converters enable, and assist *preservation planning* (e.g. Becker & Rauber, 2011). This is crucial since the scale and complexity of information assets and systems evolve towards overwhelming the capability of human archivists and curators (either system administrators, programmers and designers).

In a nutshell, the main contributions of our work are: (a) we extend the rule-based approach of (Tzitzikas, Marketakis & Antoniou, 2010) for modeling explicitly converters and emulators, (b) we demonstrate how this modeling apart from capturing the preservability of converters and emulators, enables the desired reasoning regarding intelligibility gaps, task performability, risk detection etc, (c) we introduce an algorithm for visualizing the intelligibility gaps and thus assisting their treatment, and (d) we show how the approach can be implemented using recently emerged Semantic Web tools.

4.3 MIGRATION AND EMULATION

Migration (according to Wikipedia) is a set of organized tasks designed to achieve the periodic transfer of digital materials from one hardware/software configuration to another, or from one generation of computer technology to a subsequent generation. This may include, for example, conversion of resources from one file format to another (e.g., conversion of Microsoft Word to PDF or OpenDocument) or from one operating system to another (e.g., Windows to GNU/Linux) such that the resource remains fully accessible and functional.

The purpose of migration is to preserve the integrity of digital objects and to retain the ability for clients to retrieve, display, and otherwise use them in the face of constantly changing technology.

Emulation (according to Wikipedia) combines software and hardware to reproduce in all essential characteristics the performance look and feel and behavioral aspects of another computer of a different design (i.e. future generations computer), allowing programs or media designed for a particular environment to operate in a different, usually newer environment. It is important to notice that emulation, differently from migration, does not focus on the digital object, but on the hardware and software environment in which the object is rendered. Emulation aims at recreating the environment in which the digital object was originally created and leaves the digital object untouched. Emulation requires an emulator, a program that translates code and instructions from one computing environment so it can be properly executed in another. Popular examples of emulators include QEMU (Bellard, 2005), Dioscuri (Van der Hoeven, Lohman & Verdegem, 2008), etc. There is currently a rising interest on emulators for the needs of digital preservation (Lohman, Kiers, Michel & van der Hoeven, 2011). Indicatively, (Suchodoletz, et al., 2010) overviews the emulation strategies for digital preservation and discusses related issues, while several recent projects have focused on the development of emulators for the needs of digital preservation (e.g. see (Van der Hoeven, Lohman & Verdegem, 2008) and (Rechert, Suchodoletz & Welte, 2010)). Also (Van der Hoeven, Lohman & Verdegem, 2008) compares applications running on Dioscuri with the same applications executed directly on the host machine. We should refer at this point to the KEEP[8] (Keeping Emulation Environments Portable) project, which aims at developing emulation services to enable accurate rendering of both static and dynamic digital objects. The overall aim of the project is to facilitate universal access to cultural heritage resources. KEEP has created an Emulation Framework[9] (EF) which provides additional services aiming at building a more solid ground for the emulation preservation strategy. KEEP is

[8] http://www.keep-project.eu/ezpub2/index.php?/eng
[9] http://emuframework.sourceforge.net/

depending on existing and future emulators, and has not created an emulator itself. Another related concept is that of the Universal Virtual Computer (UVC) that was introduced in (Lorie, 2001) (more recent work includes Van der Hoeven, van Diessen & van der Meer, 2005). It is a special form of emulation where a hardware and software independent platform is implemented, files are migrated to an UVC internal representation format and the whole platform can be easily emulated on newer computer systems[10]. It is like an intermediate language for supporting emulation

In brief, and from a dependency perspective, we could say that the *migration* process *changes the dependencies* (e.g. the original digital object depends on an old format, while the migrated digital object now depends on a newer format). Regarding *emulation* we could say that the emulation process does not change the dependencies of digital objects (but it can create new dependencies). An emulator essentially makes available the behavior of an old module (actually by emulating its behavior). It follows that the availability of an emulator can "satisfy" the dependencies of some digital objects, but we should note that the emulator itself has its own dependencies that have to be preserved. The same also holds for converters.

Running Example

James has a laptop where he has installed the NotePad text editor, the javac 1.6 compiler for compiling Java programs and JRE1.5 for running Java programs (bytecodes). He is learning to program in Java and C++ and to this end, and through NotePad he has created two files, HelloWorld.java and HelloWorld.cc, the first being the source code of a program in java, the second of one in C++. Consider another user, say Helen, who has installed in her laptop the Vi editor and JRE1.5. Suppose that we want to preserve these files, i.e. to ensure that in future James and Helen will be able to edit, compile and run these files. In general, to edit a file we need an editor, to compile a program we need a compiler, and to run the bytecodes of a Java program we need a Java Virtual Machine. To ensure preservation we should be able to express the above, that is we should be able to express the performability of a task through the explicit representations of dependencies. For instance, the task of compiling depends on the availability of a compiler, the task of editing depends on the availability of an editor and so on.
To this end we could use facts and rules. For example, we could state: *A file is editable if it is TextFile and a TextEditor is available*. Since James has two text files (HelloWorld.java, HelloWorld.cc) and a text editor (NotePad), we can conclude that these files are editable by him. By a rule of the form: *If a file is Editable then it is Readable too*, we can also infer that these two files are also readable. We can define more rules in a similar manner to express more task-based dependencies, such as compilability, runability etc. For our running example we could use the following facts and rules:

[10] For more see http://en.wikipedia.org/wiki/UVC-based_preservation

Table 2 An intuitive example of Facts and Rules

Facts	James	Hellen
NotePadis a TextEditor	x	
VI is a TextEditor		x
HelloWorld.java is a JavaFile	x	
HelloWorld.cc is a C++File	x	
javac1.6 is a JavaCompiler	x	
JRE1.5 is a JVM	x	x
gcc is a C++Compiler	x	
Rules		
A file is Editable if it is a TextFile and a TextEditor is available		
A file is JavaCombilable if it is a JavaFile and aJavaCompiler is available		
A file is C++Combilable if it is a C++File and aC++Compiler is available		
A file is Compilable if it is JavaCompilableorC++Compilable		
A file is a TextFile if it is JavaFile or C++File		
If a file is Editable then it is Readable		

The last two columns indicate which facts are valid for James and which for Helen. From these we can infer that James is able to compile the file HelloWorld.java and that if James sends his TextFiles to Helen then she can only edit them but not compile them since she has no facts about Compilers.

Let us now extend our example with *converters* and *emulators*. Suppose James has also an old source file in Pascal PL, say game.pas, and he has found a *converter* from Pascal to C++, say p2c++. Further suppose that he has just bought a smart phone running Android OS and he has found an *emulator* of WinOS over Android OS. It should follow that James can run game.pas on his mobile phone (by first converting it in C++, then compiling the outcome, and finally by running over the emulator the executable yielded by the compilation).

5 REQUIREMENTS ON REASONING SERVICES

Regarding curation services, we have identified the following key requirements:

Task-Performability Checking
To perform a task we have to perform other subtasks and to fulfill associated requirements for carrying out these tasks. Therefore, we need to be able to decide whether a task can be performed by examining all the necessary subtasks. For example, we might want to ensure that a file is runnable, editable or compilable. This should also exploit the possibilities offered by the availability of converters. For example, the availability of a converter from Pascal to C++, a compiler of C++ over Windows OS and an emulator of Windows OS over Android OS should allow to infer that the particular Pascal file is runnable over Android OS.

Risk Detection
The loss or removal of a software module could also affect the performability of other tasks that depend on it and thus break a chain of task-based dependencies. Therefore, we need to be able to identify which tasks are affected by such removals.

Identification of missing resources to perform a task
When a task cannot be carried out it is desirable to be able to compute the resources that are missing. For example, if Helen wants to compile the file HelloWorld.cc, her system cannot perform this task since there is not any C++Compiler. Helen should be informed that she should install a compiler for C++ to perform this task.

Support of Task Hierarchies
It is desirable to be able to define task-type hierarchies for gaining flexibility and reducing the number of rules that have to be defined.

Properties of Dependencies
Some dependencies are transitive, some are not. Therefore, we should be able to define the properties of each kind of dependency.

6 GENERAL METHODOLOGY FOR APPLYING THE DEPENDENCY MANAGEMENT APPROACH

Below we adapt the methodology introduced in (Marketakis & Tzitzikas, 2009) for the case of the rule-based dependency management approach.

Step 1) Identify desired **tasks** and objectives.

Step 2) Model the identified **tasks** and their **dependency types**. If tasks can be hierarchically organized, then this should be done.

Step 3) Specialize the **Rule-based modeling** according to the results of the previous step.

Step 4) **Capture the dependencies** of the digital objects of the archive. This can be done manually, automatically or semi-automatically. Tools like *PreScan* can aid this task. Various possible granularities: object-level, type-level, collection-level.

Step 5) Customize, use and **exploit the dependency services** according to the needs. For instance, the intelligibility-related services can be articulated with *monitoring* and *notification* services.

Step 6) **Evaluate** the services in **real tasks** and **curate** accordingly the repository (return to Step 1).

Figure 7 The main steps of the methodology

7 MODELING TASKS AND THEIR DEPENDENCIES

At first we provide some background information about the modeling and reasoning framework that we will use.

Note: This Section contains technical material, appropriate for readers having a computer science/engineering background. A reader without such background could skip this Section.

In brief this Section explains how the information has to be modeled by a system for making feasible the intended reasoning services.

7.1 RULES AND DATALOG

We will model our approach in Datalog (Ceri, Gottlob, & Tanca, 1989) which is a query and rule language for deductive databases (syntactically subset of Prolog).

In brief, a Datalog program consists of *facts*, e.g. JavaFile (myfile.java), and *rules*. An example of a rule having a *head* with a predicate of two variables and a *body* with two monadic predicates is: Compilable(X,Y) :- JavaFile(X), JavaCompiler(Y), which is read as follows: if we have a java file f1, and a java compiler f2, then we can infer that f1 is compilable by f2.

In Datalog, the set of predicates is partitioned into two disjoint sets, EPred and IPred. The elements of EPred denote extensionally defined predicates, i.e. predicates whose extensions are given by the facts of the Datalog programs (i.e. tuples of database tables), while the elements of IPred denote intentionally defined predicates, where the extension is defined by means of the rules of the Datalog program.

7.2 MODELING APPROACH

In accordance with (Tzitzikas, Marketakis & Antoniou, 2010), digital files and profiles (as well as particular software archives or system settings) are represented by facts (i.e. database tuples), while task-based dependencies (and their properties) are represented as Datalog rules.

To assist understanding, the following Figure depicts the basic notions in the form of a rather informal concept map, in the sense that a rule-based approach cannot be illustrated with a graph in a manner both intuitive and precise. A rough description of the Figure follows:

A task can have dependencies (task dependencies), meaning that for applying this task over a module, other modules should be available, and/or the performability of other tasks may be required (over the same module or different modules). Each module has a module type, and module types can be hierarchically organized. Now converters and emulators are special types of modules each having "source" and "destination" module types (broadly speaking). Analogously, a special case of task performability is conversion and emulator performability.

Below we will detail the modeling and realization of this model.

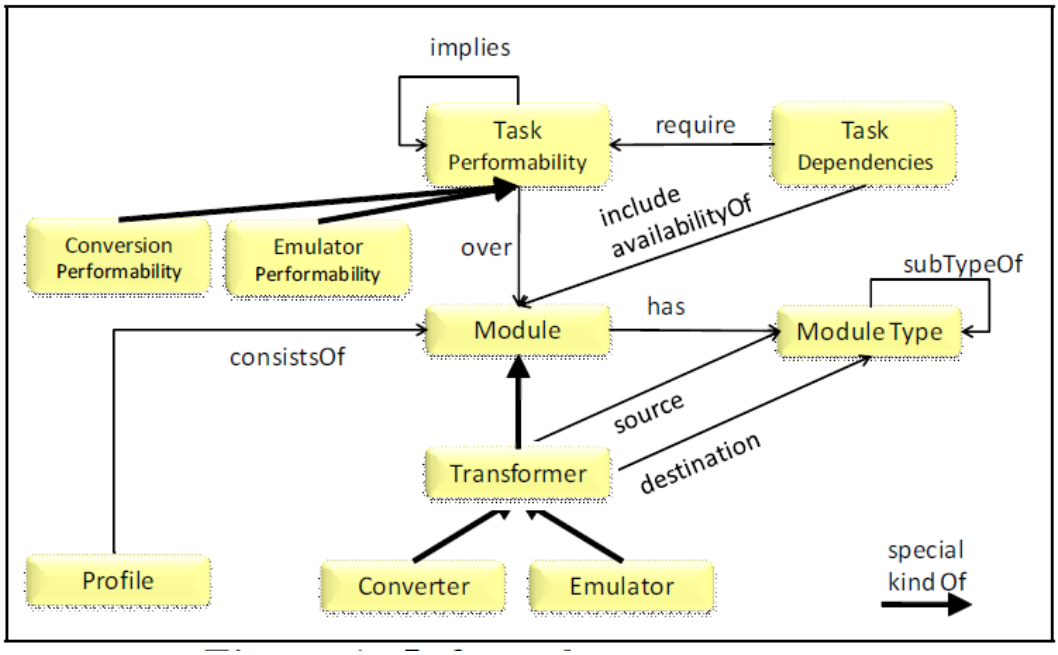


Figure 8 Informal concept map

Digital Files, Type Hierarchies, and Profiles
Digital files and their types are represented as facts using predicates that denote their types, e.g. for the three files of our running example we can have the facts shown in the left column of the following Figure. Software components are described analogously (e.g. see right column).

Facts	
for digital files	for software components
`JavaFile(HelloWorld.java).`	`TextEditor(vi).`
`C++File(HelloWorld.cc).`	`JVM(jre1.5win)`
`PascalFile(game.pas).`	`JVM(jre1.6linux)`

Figure 9 Example of facts that model digital files and software components

Each file can be associated with more than one type. In general, we could capture several features of the files (apart from types) using predicates (not necessarily unary), e.g. LastModifDate (HelloWorld.java, 2008-10-18).

The types of the digital files can be organized *hierarchically*, and such taxonomies can be represented with rules, e.g. to define that every JavaFile is also a UTF8File we must add the rule UTF8File(X) :- JavaFile(X). A *profile* is a set of facts, describing the modules available (or assumed to be known) to a user (or community). For example, the profiles of James and Helen are the ticked facts in the corresponding columns of Table 2.

Task-Dependencies and Task Hierarchies
We will also use (IPred) predicates to model tasks and their dependencies (i.e. rules). Specifically, for each real world task we define *two* intentional predicates: one (which is usually unary) to denote the (performability of the) task, and another one (with arity greater than one) for denoting the dependencies of the task. For instance, `Compile(HelloWorld.java)` will denote the compilability of HelloWorld.java. Since its compilability depends on the availability of a compiler (specifically a compiler for the Java language), we can express this

dependency using a rule of the form: `Compile(X) :- Compilable(X,Y)` where the binary predicate `Compilable(X,Y)` is used for expressing the appropriateness of a Y for compiling a X. For example, `Compilable(HelloWorld.java, javac 1.6)` expresses that HelloWorld.java is compilable by javac 1.6. It is beneficial to express such relationships at the class level (not at the level of individuals), specifically over the types (and other properties) of the digital objects and software components, i.e. with rules of the form:

```
Compilable(X,Y) :- JavaFile(X), JavaCompiler(Y).
Compilable(X,Y) :- C++File(X), C++Compiler(Y).
Runable(X,Y) :- JavaClassFile(X), JVM(Y).
Editable(X,Y) :- JavaFile(X), TextEditor(Y).
```

Relations of higher arity can be employed based on the requirements, e.g.:

```
Run(X) :- Runnable(X,Y,Z)
Runnable(X,Y,Z) :- JavaFile(X), Compilable(X,Y), JVM(Z)
```

We can express *hierarchies of tasks* as we did for file type hierarchies, for enabling deductions of the form: "if we can do task A then certainly we can do task B", e.g. "if we can edit something then certainly we can read it too" expressed as: `Read(X) :- Edit(X).`

We can also express *general properties* of task dependencies, like *transitivity*. For example, from `Runnable(a.class,JVM)` and `Runnable(JVM, Windows)` we might want to infer that Runnable(a.class, Windows). Such inferences can be specified by a rule of the form:
`Runable(X,Y) :- Runnable(X,Z), Runnable(Z,Y).`
As another example, `IntelligibleBy(X,Y) :-IntelligibleBy(X,Z), IntelligibleBy(Z,Y).`
This means that if X is intelligible by Z and Z is intelligible by Y, then X is intelligible by Y. This captures the assumptions of the dependency model described in (Tzitzikas, 2007) (i.e. the transitivity of dependencies).

Modeling Converters

Conversions are special kinds of tasks and are modeled differently. In brief, to model a converter and a corresponding conversion we have to introduce one unary predicate for modeling the converter (as we did for the types of digital files) and one rule for each conversion that is possible with that converter (specifically one for each supported type-to-type conversion). In our running example, consider the file game.pas (which contains source code in Pascal PL), and the converter p2c++ from Pascal to C++. Recall that James has a compiler for C++. It follows that James can compile game.pas since he can first convert it in C++ (using the converter), then compile it and finally run it. To capture the above scenario it is enough to introduce a predicate for modeling the converters from Pascal to C++, say ConverterPascal2C++, and adding the following rule:
`C++File(X) :- PascalFile(X), ConverterPascal2C++(Y).`
Since the profile of James will contain the facts PascalFile(game.pas) and ConverterPascal2C++(p2c++), we will infer C++File(game.pas), and subsequently that this file is compilable and runnable. Finally, we should not forget that a converter is itself a module with its own dependencies, and for performing the intended task the converter has to be runnable. Therefore, we
have to update the rule as follows:
`C++File(X) :- PascalFile(X), ConverterPascal2C++(Y), Run(Y).`

Modeling Emulators

Emulation is again a special kind of task and is modeled differently. Essentially we want to express the following:
(i) If we have a module X which is runnable over Y,
(ii) and an emulator E of Y over Z (hosting system=Z, target system=Y),
(iii) and we have Z and E,
(iv) then X is runnable over Z.
For example, consider the case where:

X=a.exe (a file which is executable in Windows operating system),
Y=WinOS (the Windows operating system),
Z=AndroidOS (the Android operating system), and
E=W4A (i.e. an emulator of WinOS over AndoidOS).

In brief, for each available emulator (between a pair of systems) we can introduce a unary predicate for modeling the emulator (as we did for the types of digital files, as well as for the converters), and writing one rule for the emulation. For example, suppose we have a file named a.exe which is executable over WinOS. For this case we would have written:

```
Run(X) :- Runnable(X,Y)
Runnable(X,Y) :- WinExecutable(X), WinOS(Y)
```

and the profile of a user that has this file and runs WinOS would contain the facts WinExecutable(a.exe) and WinOS(mycomputer), and by putting them together it follows that Run(a.exe) holds. Now consider a different user who has the file a.exe but runs AndroidOS. However, suppose that he has the emulator W4A (i.e. an emulator of WinOS over AndoidOS). The profile of that user would contain:

```
WinExecutable(a.exe)
AndroidOS(mycomputer) // instead of WinOS(mycomputer)
EmulatorWinAndroid(W4A)
```

To achieve our goal (i.e. to infer that a.exe is runnable), we have to add one rule for the emulation. We can follow two approaches. The first is to write a rule that concerns the runnable predicate, while the second is to write a rule for classifying the system that is equipped with the emulator to the type of the emulated system:

A. Additional rule for Runnable
This relies on adding the following rule:

```
Runnable(X,Y,Z):- WinExecutable(X), EmulatorWinAndroid(Y), AndroidOS(Z)
```

Note that since the profile of the user contains the fact EmulatorWinAndroid(W4A) the body of the rule is satisfied (for X=a.exe, Y=W4A, Z=myComputer), i.e. the rule will yield the desired inferred tuple Runnable(a.exe,W4A,mycomputer). Note that here we added a rule for the runnable which has 3 variables signifying the ternary relationship between executable, emulator and hosting environment.

B. Additional type rule (w.r.t. the emulated Behavior)
An alternative modeling approach is to consider that if a system is equipped with one emulator then it can also operate as the emulated system. In our example this can be expressed by the following rule:
`WinOS(X):- AndroidOS(X), EmulatorWinAndroid(Y).`
It follows that if the profile of the user has an emulator of type EmulatorWinAndroid (here W4A) and mycomputer is of type AndroidOS, then that rule will infer WinOS(mycomputer), implying that the file a.exe will be inferred to be runnable due to the basic rule of runnable which is independent of emulators (i.e. due to the rule `Runnable(X,Y) :- WinExecutable(X), WinOS(Y)`).
Both approaches (A and B) require the introduction of a new unary predicate about the corresponding pair of systems, here EmulatorWinAndroid. Approach (A) requires introducing a rule for making the predicate runnable "emulatoraware", while approach (B) requires a rule for classifying the system to the type of the emulated system. Since emulators are modules that can have their own dependencies, they should be runnable in the hosting system. To require their runnability during an emulation we have to update the above rules as follows (notice that last atom in the bodies of the rules):

```
A':Runnable(X,Y,Z):-                    |     B': WinOS(X):-
WinExecutable(X),                       |     AndroidOS(X),
EmulatorWinAndroid(Y),                  |     EmulatorWinAndroid(Y),
```

```
AndroidOS(Z),                          |      Runnable(Y,X)
Runnable(Y,Z)                          |
```

Modeling Important Parameters

Sometimes it is important to model the required (important) parameters for the performability of a task. For example, an emulator may need a particular parameter for emulating a particular system. In this case it is beneficial to model this explicitly. Methodologically, it is not suggested to model all parameters, e.g. those of minor importance, but only the crucial ones, those for enabling the required reasoning. For example, consider the following rule:

```
WinOS(X):- AndroidOS(X), EmulatorWinAndroid(Y),Runnable(Y,X)
```

and suppose that this emulator needs one particular parameter for emulating windows, say a file winImg.dat. One way to capture this, is to extend the above rule as:

```
WinOS(X):-AndroidOS(X),EmulatorWinAndroid(Y),Runnable(Y,X),Module(winImg.dat)
```

where Module is the top class of the module type hierarchy. This rule will fire only if the winImg.dat is recorded in the system.

7.3 CHECKING TASK PERFORMABILITY USING THE PROPOSED MODELING APPROACH

The task performability service aims at answering if a task can be performed by a user/system. It relies on query answering over the Profiles of the user. E.g. to check if HelloWorld.cc is compilable we have to check if HelloWorld.cc is in the answer of the query Compile(X). As we described earlier, converters and emulators will be taken into account, meaning that a positive answer may be based on a complex sequence of conversions and emulations. This is the essential benefit from the proposed modeling. For example, let us check the performability of the running example, described in Section 4.3, for the user James. The goal is to check if James can run the game.pas file on his mobile phone. Indeed the fact Runnable(game.pas,smartPone) can be derived as shown in proof tree shown in Figure 10. In that Figure each fact is represented by a rectangle, while colored rectangles indicate the applicable rules.

The used facts in this example are:

```
PascalFile(game.pas),
ConverterPascal2C++(p2c++),
WinOS(mycomputer),
AndroidOS(smartPhone),
C++Compiler(gcc),
EmulatorW4A(W4A)
```

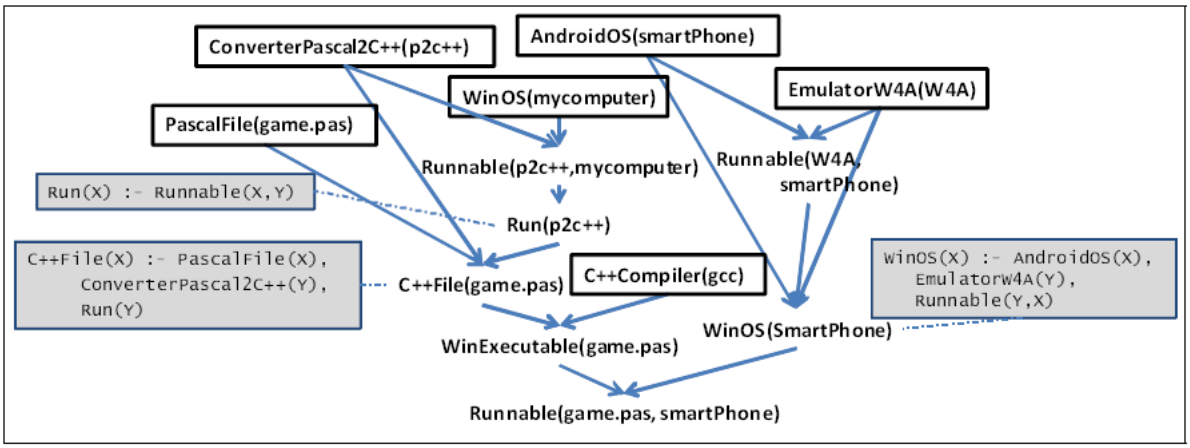


Figure 10 The proof Tree

7.4 METHODOLOGY (FOR USING THE RULE-BASED APPROACH)

Here we analyze the Step 3 of the methodology illustrated in Figure 7. Specifically, for each real world task we define two intentional predicates: one (which is usually unary) to denote the performability of the task, and another one (which is usually binary) for denoting the dependencies of the task (e.g. Read(X) and Readable(X,Y)).

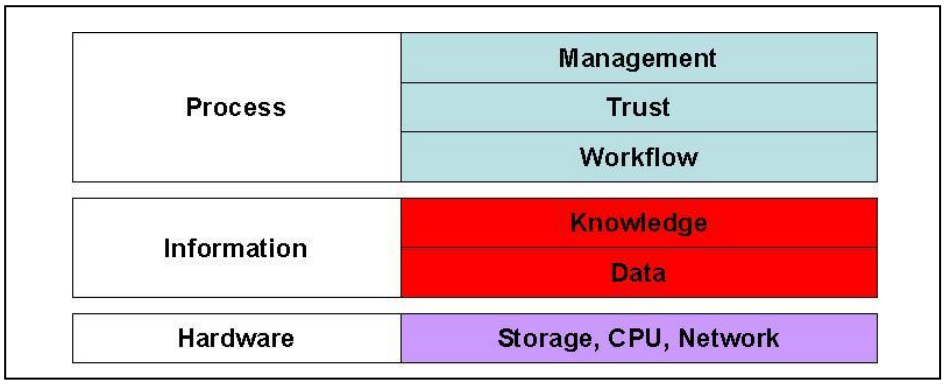
To model a *converter* and a corresponding conversion we have to introduce one unary predicate for modeling the converter (as we did for the types of digital files) and one rule for each conversion that is possible with that converter (specifically one for each supported type-to-type conversion). To model an *emulator* (between a pair of systems) we introduce a unary predicate for modeling the emulator and writing one rule for the emulation. Regarding the latter we can either write a rule that concerns the runnable predicate, or write a rule for classifying the system that is equipped with the emulator to the type of the emulated system. Finally, and since converters and emulators are themselves modules, they have their own dependencies, and thus their performability and dependencies (actually their runnability) should be modeled, too (as in ordinary tasks).

8 WHICH TASKS TO MODEL, HIERARCHY OF TASKS

The rising question is: ***Which tasks are the more useful for the needs of digital preservation?***

8.1 ORGANIZING TASKS HIERARCHICALLY

We could organize such tasks hierarchically. This increases the flexibility of the process and reduces possible redundancies. This is quite natural, and we have seen that the community has tried to provide a kind of layering. For instance, as already described in D25.1, the Warwick Workshop, Digital Curation and Preservation: "Defining the research agenda for the next decade", held in November 2005, noted that virtualization is an underlying theme, with a layering model illustrated as follows:



The common research issues that were identified at that point were:

Automation and Virtualization	• Develop language to **describe data policy** demands and processes together with associated support systems.
	• Develop **collection oriented description** and transfer techniques.
	• Develop data description tools and associated generic migration applications to **facilitate automation**.
	• Develop **standardized intermediate forms** with sets of coder/decoder pairs to and from specific common formats.
	• Develop **code generation tools** for automatically creating software for format migration.
	• Develop techniques to allow **data virtualization of common science objects** with at least some discipline specific extensions.
	• **Management and policy specifications** will be need to be formalized and virtualized.
	• Further **virtualization of knowledge** – including developments of interoperable and maintainable ontologies.
	• Develop **automatic processes for metadata extraction.**

Specific research topics included:

Virtualization	• Continuing work on ways of **describing information all the way from the bits upwards**, in standardized ways – "virtualization". Work is needed on each of the identified layers in section A1.2.
	• **Knowledge virtualization** involving Ontologies and other Semantic Web developments are required to enable the characterization of the applicability of a set of relationships across a set of semantic terms.
	• Develop use of **data format description languages** to characterize the structures present within a digital record, independently of the original creation application.
	• It is important to make significant progress on dealing with **dynamic data including databases**, and object behaviour.
	• **Representation Information tools**, probably via layers of virtualization to allow appropriate normalization, including mature tools for dealing with dynamic data including databases.
	• Additional work on **preservation strategies and support tools** from emulation to virtualization.
	• Develop increasingly powerful virtualization **tools** and techniques with a particular emphasis on knowledge technologies.
Automation	• Develop protocols and information management exchange mechanisms, including synchronization techniques for indices etc., to **support federations.**
	• **Standardized APIs** for applications and data integration techniques
	• Fuller development of **workflow systems and process definition** and control.
Support	• Develop simple semantic **descriptions of Designated Communities**.
	• **Standardize Registry/Repositories for Representation Information** to facilitate sharing.
	• Develop **methodologies and services for archiving personal collections** of digital materials.
Hardware	• Develop and **standardize interfaces** to allow "pluggable" storage hardware systems.
	• **Standardize archive storage API** i.e. standardized storage virtualization.
	• Develop **certification processes for storage systems**.
	• Undertake research to characterise types of read and **transmission errors** and the development of techniques which detect and potentially correct them.

Table 3 Research Topics (Warwick Workshop)

From the above we can understand that a *layering* is important for increasing the flexibility of digital preservation architectures, which can be adapted to a variety of preservation scenarios.

Also note that the modeling perspective elaborated in this deliverable allows modeling various tasks and organizing them hierarchically. Below we describe (quite generally) some tasks. In some cases, the more we go down to the list, the more complex the tasks become, i.e. some of these tasks rely on the abilit**y** of performing other tasks.

Ability to:

- **Retrieve** the bits: Ability to get a particular set of stored bits.
- **Access:** Ability to retrieve the bits starting from an identifier (e.g. a persistent identifier)
- **Render:** Given a set of bits, ability to render them using the right symbol set (e.g. as defined in Doerr, Tzitzikas, 2012) for creating the intended sensory impression.
- **Run**: Ability to run a program in a particular computer platform.
- **Search**: Ability to find a digital object. Search ability can be refined based on the type of the object (doc, structured, composite) and its searchable part (contents, structure, metadata).
- **Link:** Ability to place a digital object in context and exploit it. This may require combining data across difference sources.
- **Assert Quality:** Ability to answer questions of the form: What is its value of this digital object, is it authentic?
- **Get Provenance:** Ability to answer the corresponding questions (who, when, how).
- **Assert Authenticity:** (based on provenance, etc)
- **Reproduce:** Ability to reproduce a scientific result. This is crucial for e-Science.
- **Update**: Ability to update and evolve a digital object.
- **Upgrade/Convert/Transform:** Ability to upgrade a digital object (e.g. to a new format), or convert its form.

8.2 SOME INTICATIVE BASIC TASKS

Instead of elaborating on everything, for the needs of the dependency management approach, it is worth focusing on some specific tasks for testing and evaluating the approach.

We could start from three main kinds of digital objects: ***documents***, ***datasets*** and ***software***. For each one of the above kinds, we can attempt to identify some key tasks, whose performability are important and aligned with the objectives of digital preservation.

Documents
Mainly we want to render them. We can say that the main task is the *projection* as defined in (Doerr, Tzitzikas, 2012). Its performability is based on the format of the information carrier. This means that each digital document has a type (e.g. pdf,doc, docx) and that type determines its dependencies for this task, e.g. the projection of a docx file requires the X version of MSWord.

Software
In this document we have already provided many examples about software. We could say that the main task is Run. For achieving runnability, when the used technologies change, the task Compile is also important.

Datasets
Suppose datasets are in the form of tabular data. The main task is to understand what the stored values are, e.g. that the data in the second column are Celsius degrees. Moreover we would like to know their creation context, e.g. that the dataset contains temperatures measured in the location of Knossos using a thermometer Y, on behalf of the National Meteorology Agency. Probably for the needs of intelligibility, the provision of EAST/DESL descriptions (as discussed in (Marketakis & Tzitzikas, 2009)), are enough. So, they could be considered as the required dependencies for the task of intelligibility of datasets. As discussed in the same paper (Marketakis & Tzitzikas, 2009), instead of having data in tabular form associated with EAST descriptions, an alternative choice is to keep them stored in RDF/S. In this way their structure and semantics are more self-describing, since an ontology can be used to represent explicitly the conceptualization behind the data and descriptions based on this ontology can be used to represent instances of the ontology categories. Another positive point of this choice is that in this way the dataset is better linked with the “external” world (in the sense that it can refer to external

entities and external ontologies using their URIs). Moreover, the ontology can be extended to represent more types of data across time. This is aligned with the W3C proposals related to the Semantic Web. For instance, and according to ***5 Stars Open Data***[11], linked data are rated as follows:

★ make your stuff available on the Web (whatever format) under an open license [1]

★★ make it available as structured data (e.g., Excel instead of image scan of a table) [2]

★★★ use non-proprietary formats (e.g., CSV instead of Excel) [3]

★★★★ use URIs to denote things, so that people can point at your stuff [4]

★★★★★ link your data to other data to provide context [5]

What each rating means is also illustrated in Figure 11.

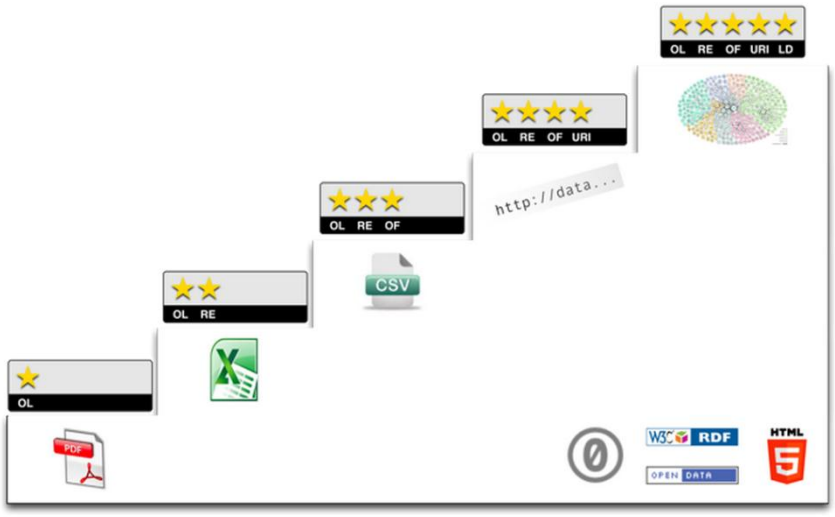


Figure 11 5 Stars Open Data rating

A rising question is how the above scale is related to the dependency perspective that is elaborated in the current deliverable. To this end, we have to investigate which task is assumed in each case. For this reason below we consider a similar example as in http://5stardata.info/, i.e. we consider "the temperature forecast for Galway, Ireland for the next 3 days":

Rating	Assumed task or tasks
1 star	Assumed Task: Get the forecast data in digital form and in a readable way. This is related to the tasks "**Retrieve" , "Render"** and "**Access**" described earlier.

[11] http://5stardata.info/

2 stars	Assumed (extra) task: Get the structure of the data, i.e. one should be able to answer queries of the form: how many rows and how many columns this dataset contains, or what is the value of the cell [i,j]. Note that if the dataset were stored as a picture, then to answer such queries we would depend on the availability of image processing software (e.g. OCR). If on the other hand, it is stored in .xls, then we depend on the availability of MS Excel. This task is also related to the task **Search** described earlier.
3 stars	Assumed task: As in 2stars, but here we can want to get information about the structure without relying on commercial software like MSOffice, but on a more general and open format, e.g. CVS, which requires having just a text editor.
4 stars	The provision of URIs allows citing the dataset, as a whole, but also a particular piece of that dataset. Assumed task: It is related to the task **Link** described earlier.
5 stars	Here the "context links" allow answering questions of the form: what, where, when, etc. This resembles *provenance* queries. Assumed task: It is related to the tasks **Link** and **Get Provenance** described, earlier.

Below we provide an example, demonstrating that higher star rating implies performability of more tasks. In addition these tasks can be performed with less, or more easily resolvable, dependencies.

Suppose that we want to process the contents of a file containing the weather forecast (i.e. air temperature, surface wind, rainfall and snowfall in mm, cloudiness) for Heraklion city for the following 3 days. The processing of the data will be made by a software agent (e.g. a specific application which produces weather statistics). We will describe the various tasks and their dependencies assuming that the original data fall into the five categories defined by 5-star-data.

: All the data are available as an image (in jpeg format and accessible through the internet) like the following:

Figure 12 A jpeg image containing the weather forecast for Heraklion city

The first obvious task for the agent is the ability to **retrieve** the data. After retrieving the file (in other terms downloading), the agent should **extract** the data from the image. This task depends on the existence of an appropriate OCR program. The latter will extract the data, even all these data are just characters and numbers for the agent. Therefore, an additional file (e.g. in pdf format) containing the specification of the data should be **retrieved** and **accessed** from the agent. To access the contents of the specification an appropriate reader for this format should be provided. To sum up, the following tasks are required for manipulating the data:

Retrieve(forecast.jpeg)	Download the data from the Internet.
Retrieve(forecastSpecification.pdf)	Download the specification from the Internet.
ExtractData(foreast.jpeg)	Extract the data using an appropriate OCR program. This means that additional tasks and dependencies are required (e.g. OCR program

	execution dependencies).
Read(forecastSpecification.pdf)	Read the specification using an appropriate application (e.g. PDF reader). This will add more tasks and dependencies.

It is obvious that apart from the tasks described above, further tasks (and dependencies) are required for executing the above applications (OCR program, PDF viewer).

: The data are available through the internet in xls format. After **retrieving** the data, the agent should **read** the contents using an appropriate application (Excel). Similarly to the previous category, the data carry no semantics; so additional information should be provided. In this case however, the specification can be added with the structured data; so the downloaded file contain also the specification that is required. To sum up, the following tasks are required for manipulating the data:

Retrieve(forecast.xls)	Download the data from the Internet.
Read(forecast.xls)	Read the contents of the downloaded file using Excel.

Apart from the above tasks, the runability of Excel requires running some other tasks (and adding some more dependencies).

: The data are available through the internet in XML format. The first task is to **retrieve** them and then **read** the contents. The dependencies for reading the contents are simpler in this case (compared to the 2-star case) because no particular commercial application is required for reading them. The agent can use any text editor for reading them (and the specification as well) which simplifies the dependencies for this task. To sum up, the following tasks are required for manipulating the data:

Retrieve(forecast.xml)	Download the data from the Internet.
Read(forecast.xml)	Read the contents of the downloaded file using a text editor.

Apart from the above tasks, the runability of a text editor might add some more dependencies, which are much simpler (compared to the runability of Excel and OCR programs).

: This case is similar to the 3-star case. The only difference is that data are referenced using URIs, which makes no difference in terms of the tasks that are applicable over the data.
However, citability is important for other applications and resources, e.g. a researcher in a scientific paper would like to cite this particular dataset, or a service that collects weather forecasts for many places in the world.

: The data are available through the Internet that RDF/XML format, and URIs are used to refer to them. The difference compared to 4-star data is semantic information about the data is not included in the same file (e.g. temperature is measured in Celsius degrees, rainfall in millimeters, etc.). This information is provided by linking the actual data with other data (or schemas) in the Web. After **retrieving** the data the agent **reads** the contents and manipulates the data. Although it is beneficial to link the data on the web, in terms of the tasks (and their dependencies), which are applicable on the data 5-star data, they do not further simplify the dependencies. To sum up, the following tasks are required.

Retrieve(forecast.rdf)	Download the data from the Internet.
Read(forecast.rdf)	Read the contents of the downloaded file using a text editor.

From the above example it is obvious that the tasks, which are applicable on 5-star data, are simpler (e.g. download and read them) compared to 1-star (or no-star) data. Furthermore, the dependencies of these tasks tend

to be less complex; the readability of the contents of an xml file (5-star data) requires the availability of a text editor, while the readability of the contents of a jpeg file (1-star data) requires their extraction using sophisticated programs (e.g. OCR programs) with much more dependencies.

9 IMPLEMENTATION TECHNOLOGIES FOR DEPENDENCY MANAGEMENT FOR DIGITAL PRESERVATION

There are several possible implementation approaches. Below we describe them in brief:

Prolog is a declarative logic programming language, where a program is a set of Horn clauses[12] describing the data and the relations between them (i.e. facts and rules). The proposed approach can be best straightforwardly expressed in Prolog. Furthermore, and regarding abduction there are several approaches that either extend Prolog (Christiansen, Dahl, 2004) or augment it (Christiansen, Dahl, 2005) and propose a new Programming Language.

The **Semantic Web Rule Language (SWRL) (**Horrocks, et al., 2004**)** is a combination of OWL DL and OWL Lite (McGuinness & Harmelen, 2004) with the Unary/Binary Datalog RuleML7. SWRL provides the ability to write Horn-like rules expressed in terms of OWL concepts to infer new knowledge from existing OWL Knowledge Bases. For instance, each type predicate can be expressed as a class. Each profile can be expressed as an OWL class whose instances are the modules available to that profile (we exploit the multiple classification of SW languages). Module type hierarchies can be expressed through *subclassOf* relationships between the corresponding classes. All rules regarding performability and the hierarchical organization of tasks can be expressed as SWRL rules. An alternative approach, on which we focus on this paper, is to use triple-stores and exploit its query-based inference capabilities (more in the next section). In a **DBMS**-approach all facts can be stored in a relational database, while *Recursive SQL* can be used for expressing the rules. Specifically, each type predicate can be expressed as a relational table with tuples the modules of that type. Each profile can be expressed as an additional relational table, whose tuples will be the modules known by that profile. All rules regarding task performability, hierarchical organosis of tasks, and the module type hierarchies, can be expressed as datalog queries. Note that there are many commercial SQL servers that support the SQL:1999 syntax regarding recursive SQL (e.g. Microsoft SQL Server 2005, Oracle 9i, IBM DB2). Indicatively, Table 4 synopsizes the various implementation approaches.

What	DB-approach	Semantic Web-approach
ModuleTypepredicates	relationaltable	class
Facts regarding Module (and their types)	tuples	classinstances
DC Profile	relationaltable	class
DC ProfilesContents	tuples	classinstances
Taskpredicates	IDBpredicates	predicatesappearinginrules
TaskTypeHierarchy	datalog rules, or isa if an ORDBMS is used	subclassOf
Performability	datalogqueries (recursive SQL)	rules

Table 4 Implementation Approaches

[12] A Horn clause is a clause (i.e. a disjunction of literals) with at most one positive literal.

9.1 AN RDF/S IMPLEMENTATION

Note: This Section contains technical material, appropriate for readers having a computer science/engineering background. A reader without such background, could skip this Section.

In brief this Section details how the proposed method can be implemented using Semantic Web technologies.

Here we describe one Semantic Web-based implementation using RDF/S and *OpenLink Virtuoso*, which is a general purpose RDF triple store with extensive SPARQL and RDF support (Erling, Mikhailov, 2007). Its internal storage method is relational, i.e. RDF triples are stored in tables in the form of quads (*g*,*s*,*p*,*o*) where *g* represents the graph, *s* the subject, *p* the predicate and *o* the object. We decided to use this system because of its inference capabilities, namely *backward chaining* reasoning, meaning that it does not materialize all inferred facts, but computes them at query level. Its reasoned covers the related entailment rules of rdfs:subClassOf and rdfs:subPropertyOf, while *user defined custom inference rules* can be expressed using *rule sets*. Practically this means that transitive relations (i.e. *subClassof*, *subPropertyOf*, etc.) are not physically stored in the Knowledge Base, but they are added to the result set at query answering. *Transitivity* is also supported in two different ways. Given a RDF schema and a rule associated with that schema, the predicates rdfs:subClassOf and rdfs:subPropertyOf are recognized and the inferred triples are derived when needed. In case of another predicate, the option for transitivity has to be declared in the query. For our case, we have to "translate" our facts and rules to quads of the form (*g, s, p, o*) which are actually RDF triples contained in a graph *g*. The support of different graphs is very useful for the cases of profiles; we can use a different graph for each profile. We will start by showing how facts can be "translated" to RDF quads and later we will show how inference rules can be expressed using ASK and CONSTRUCT or INSERT SPARQL queries. Note that if we use INSERT instead of CONSTRUCT then the new inferred triples will be stored in the triple store (materialization of inferred triples). Hereafter, we will use only CONSTRUCT. For better readability of the SPARQL statements below we omit namespace declarations.

Modules: Module types are modeled using RDF classes while the actual modules are instances of these classes. Module type hierarchies can be defined using the rdfs:subclassof relationship. For example, the fact JavaFile('HelloWorld.java') and the rule for defining the module type hierarchy TextFile(X) :- JavaFile(X) will be expressed using the following quads:

```
g, <JavaFile>, rdf:type, rdfs:Class
g, <TextFile>, rdf:type, rdfs:Class
g, <JavaFile>, rdfs:subclassof, <TextFile>
g, <HelloWorld.java>, rdf:type, <JavaFile>
```

Profiles: We exploit the availability of graphs to model different profiles, e.g. we can model the profiles of James and Helen (including only some indicative modules), as follows:

```
<jGrph>, <NotePad>, rdf:type, <TextEditor>
<jGrph>, <HelloWorld.java>, rdf:type, <JavaFile>
<jGrph>, <javac_1_6>, rdf:type, <JavaCompiler>
<hGrph>, <VI>, rdf:type, <TextEditor>
<hGrph>, <jre_1_5>, rdf:type, <JavaVirtualMachine>
```

Dependencies: The rules regarding the performability of tasks and their dependencies are transformed to appropriate SPARQL CONSTRUCT statements which produce the required inferred triples. For example, the rule about the compilability of Java files (Compilable(X,Y) :- JavaFile(X),JavaCompiler(Y)) is expressed as:

```
CONSTRUCT{?x <compilable> ?y}
```

```
WHERE{?x rdf:type <JavaFile>.
       ?y rdf:type <JavaCompiler>}
```

To capture the compilability of other kinds of source files (i.e. C++, pascal etc.) we extend the previous statement using the UNION keyword (this is in accordance with the Datalog-based rules; multiple rules with the same head have union semantics). For example the case of Java and C++ is captured by:

```
CONSTRUCT{?x <compilable> ?y}
WHERE{
       {      ?x rdf:type <JavaFile>.
              ?y rdf:type <JavaCompiler>}
       UNION
       {      ?x rdf:type <C++File>.
              ?y rdf:type <C++Compiler>}
}
```

Finally, the unary predicate for the performability of task, here Compile, is expressed as:

```
CONSTRUCT{?x rdf:type <Compile>}
WHERE{ {?x <compilable> ?y} }
```

Converters: The rules regarding conversion are modeled analogously, e.g. for the case of a converter from Pascal to C++ we produce:

```
CONSTRUCT{?x rdf:type <C++File>}
WHERE{ ?x rdf:type <PascalFile>.
       ?y rdf:type <ConverterPascal2C++>.
       ?y rdf:type <Run>}
```

Note that the last condition refers to an inferred type triple (Runnable). If there are more than one converters that change modules to a specific module type then the construct statement is extended using several WHERE clauses separated by UNIONs, as shown previously.

Emulators: Consider the scenario described in Section 3, i.e. a user wanting to run a.exe upon his Android operating system. The approach B (which does not require expressing any predicate with three variables), can be expressed by:

```
CONSTRUCT{?x rdf:type <WindowsOS>}
WHERE{ ?x rdf:type <AndroidOS>.
          ?y rdf:type <EmulatorWin4Android>.
          ?y <runnable> ?x}
```

If the emulator needs a particular parameter, as for example the module winImg.dat which we have described on Section 3 we have to add an extra triple on the previous query for this module, so we model the emulator as :

```
CONSTRUCT{?x rdf:type<WindowsOS>}
WHERE{ ?x rdf:type <AndroidOS>.
       ?y rdf:type <EmulatorWin4Android>.
       ?y <runnable> ?x.
       <winImg.dat>rdf:type <Module>}
```

Services: To realize the reasoning services (e.g. task performability, risk detection, etc), we rely on SPARQL queries. For example, to answer if the file HelloWorld.java can be compiled, we can send the INSERT query about the compilability of the files (as shown previously) and then perform the following ASK query on the entailed triples: `ASK{<HelloWorld.java><compilable> ?y}.` If this query returns true then there is at least one appropriate module for compiling the file. The risk-detection service requires SELECT and DELETE SPARQL queries (as discussed at section 4). For example, to find those modules whose *editability* will be affected when we remove the module Notepad, we have to perform :

```
SELECT ?x
WHERE {?xrdf:type <Edit>}
DELETE <Notepad>rdf:type <TextEditor>
```

From the select query, we get a set *A* containing all modules which are editable. Then, we remove the triple about Notepad and perform again the select query getting a new set *B*. The set difference *A\B* will reveal the modules that will be affected. If empty, this means that there will be no risk in deleting the Notepad.

10 PRELIMINARY TESTING

We have created and collected various datasets and we have designed and developed a preliminary prototype system for testing purposes.

10.1 PROOF OF CONCEPT DATASET AND REPOSITORY

The objective of this dataset is to allow checking the correctness of the method, and for this reason it contains all the examples that are described in this document. For instance, we have created and loaded a N-triple file that contains the triples that define the schema (classes and properties) and the facts for Jame's profile. All explicit triples are entered in one graph space, say *j_graph* (James' graph). We adopt an additional graph space, say *j_graph_compl* ("compl" from completed) that stores all explicit plus all inferred triples. The inferred triples are produced by the INSERT statements that correspond to the rules. All queries (and reasoning services) are based on *j_graph_compl.*

Moreover, all facts and rules (including the examples of real converters and emulators which are described in Section 0) have been stored in a prototype repository, accessible through a SPARQL endpoint http://62.217.127.222:8890/sparql. Specifically, the graph *j_graph_compl* contains all the produced triples from the aforementioned examples. Any user or service can connect for executing the desired SPARQL queries. We used it for validating that the implementation behaves as specified by the theory.

10.2 REAL DATASET

This Section explains how some real and well-known converters and emulators can be modeled using the Semantic Web-based implementation just described. The objective is to evaluate the expressiveness of the proposed method.

Texi2HTML converter: Texi2HTML[13] is a Perl script, which converts Texinfo source files to HTML output. Texinfo is the official documentation format of the GNU project. To model this scenario we must introduce classes for the various module types, i.e. for texi files, for perl scripts, for perl interpreters, and for the particular converter (from texi to HTML). For instance, consider a user who has a myfile.texifile, the strawberry-perl.exe perl interpreter, and the Texi2htmlScript.pl converter (from texi to HTML). The profile of this user will contain the facts:

```
PerlScript(Texi2htmlScript.pl)
PerlInterpreter(strawberry-perl.exe)
TexinfoFile(myfile.texi)
Texi2HTMLConverter(Texi2htmlScript.pl)
```

Note that Texi2htmlScript.pl (as any perl script) requires the availability of a Perl interpreter to run, therefore we should add the rule:

```
Runnable(X,Y) :- PerlScript(X), PerlInterpreter(Y)
```

As stated in Section 0, we also have to declare a rule for the conversion, in our case the rule:

```
HTML(X) :- TexinfoFile(X), Texi2HTMLConverter(Y),Run(Y)
```

Dioscuri emulator: Dioscuri[14] is a component-based x86computer hardware emulator written in Java. Each hardware component is emulated by a software surrogate called a module. By combining several modules the user can configure any computer system, as long as these modules are compatible. For example, consider a user having dioscuri emulator version0.7.0 (which requires a JVM to run) and suppose he wants to run Chess.exe, a

[13] http://www.nongnu.org/texi2html/
[14] http://dioscuri.sourceforge.net/

16-bit DOS Application on his computer with the WindowsXp Operating System (jre1.5wininstalled). Declaring again the appropriate classes, the profile of this user will contain the facts :

```
DOSExecutable(Chess.exe)
WindowsXPOS(mycomputer)
DioscuriEmulator(dioscuri-0.7.0.jar)
JavaByteCode(dioscuri-0.7.0.jar)
```

The execution of a Java ByteCode requires a JVM so:

```
Runnable(X,Y) :- JavaByteCode(X), JVM(Y)
```

From the above, we can now write the rule for the emulation:

```
DOSOS(X) :- WindowsXPOS(X), DioscuriEmulator(Y),Runnable(Y,X)
```

QEMU emulator: QEMU[15] is a generic open source machine emulator and virtualizer that can run an unmodified target operating system. To emulate another machine one needs to have the process emulator (QEMU) and an ISO image of the machine he wants to emulate. For instance, consider a user having the QEMU1.1 emulator, and an ISO file of the Windows XP, say WinXP.iso, who wants to emulate the WindowsXP OS on his Linux machine. His profile will contain the facts:

```
LinuxOS(mycomputer)
QEMUEmulator(QEMU1.1)
ISOFile(WinXP.iso)
```

Now we can write the rule :

```
WindowsXPOS(X) :- LinuxOS(X), QEMUEmulator(Y), Module(WinXP.iso)
```

As we have stated at Section 0, the emulator must be runnable in the hosting system (here mycomputer), therefore, we have to add a Runnable rule to extend the above rule and reach the following:

```
Runnable(X,Y) :- QEMUEmulator(X), LinuxOS(Y)
WindowsXPOS(X) :- LinuxOS(X), QEMUEmulator(Y), Module(WinXP.iso), Runnable(Y,X)
```

Notice that the user in his profile has the fact ISOFile(WinXP.iso), but the above rule uses the atom Module(WinXP.iso). The rule will fire because Module is the top class of the module type hierarchy (i.e. if something belongs to the class ISOFile then it also belongs to the class Module).

[15] http://wiki.qemu.org/Main_Page

11 MORE REFINED GAPS

11.1 INTRODUCTION

In the dependency management approach that we have just described, the notion of module is treated as an *atom* i.e. as an undivided element. However, in many cases, a module could have an internal structure and this structure could be known and formally expressed. In such cases, we could refine the notion of gaps, and instead of saying "module mx is missing", the internal parts of mx that are missing could be computed and provided.

An example of modules that could fall in this category are models that formally express parts of the community knowledge". Note that community knowledge is increasingly coded in a structured way. For instance, classification schemes, taxonomies, thesauri, are expressed in SKOS (Simple Knowledge Organization System, http://www.w3.org/2004/02/skos/). Ontologies are used to define the concepts of particular domains and their relationships. Methods and tools that extract and publish structured knowledge from text are also evidenced. Some typical examples are (a) the DBpedia, that publishes structured knowledge extracted from Wikipedia, (b) YAGO2, that extracts knowledge from Wikipedia, WordNet and Geonames, and (c) Freebase, that extracts data from sources such as Wikipedia, ChefMoz, NNDB and MusicBrainz. As a transition step the official recommendations also support RDF annotation within XHTML, like RDFa and Microdata. RDFa recommendation supports the addition of a set of attribute-level extensions to HTML, XHTML and various XML-based document types for embedding rich RDF structures information within Web documents. Indicatively, Facebook uses RDFa through the Open Graph Protocol to integrate web pages into the Facebook social graph. Overall, we can say that RDF/S is currently the **lingua franca** for expressing these models.

Moreover, we should mention that RDF/S has been proposed as a data structure for software engineering, specifically for expressing software structure and dependencies. For example, there are tools that scan Java bytecode for method calls and create a description of the dependencies between classes and the package/archive encoded in RDF. Other tools transform Maven POM (Project Object Model) files into RDF.

However, as world evolves, these models evolve (sometimes weekly) and there is a need for help in identifying the changes in these models for understanding evolution.

For this reason below we discuss other kinds of gaps, including gaps that concern descriptive metadata, gaps in the form of change operations, and gaps of finer granularity. This perspective is complementary to the dependency management approach described earlier.

Note that general purpose differential functions for RDF/S Knowledge Bases were described in (Zeginis, Tzitzikas & Christophides, 2011). A recent advancement (that was achieved in the context of APARSEN) that exploits blank node name anonymity for reducing the delta is described in (Tzitzikas, Lantzaki & Zeginis, 2012) and is discussed below.

11.2 MORE REFINED GAPS: APPROACHES

Recall that according to OAIS an object can have various descriptive metadata. Let's assume that all these metadata are represented with respect to ontologies expressed in RDF/S. In particular, consider two objects o_1 and o_2 where the metadata of o_1 are expressed with respect to an ontology A, while those of o_2 are expressed with respect to an ontology C. Now consider a particular community p_1 that is not familiar with any of the ontologies used for expressing metadata, and a community p_2 that is familiar with ontology B and suppose that C is a specialization of B (i.e. it reuses and extends elements of B), as shown in the next Figure. Familiarity with an ontology means familiarity with the domain of the ontology and the conceptualization of that ontology.

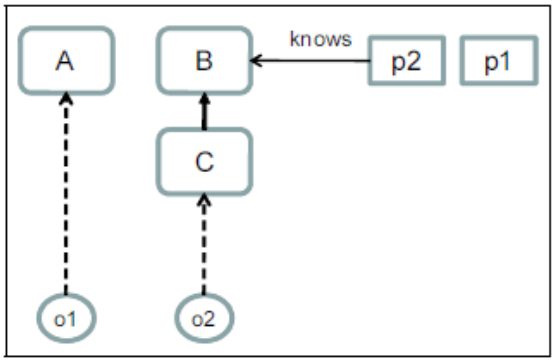


Figure 13 Example of dependencies to ontologies

We could define the *descriptive gap* between an object o and a community profile p, denoted by dgap(o, p), as the set of ontologies that a member of the p community needs to understand in order to understand the metadata of o. In our case, this would mean that:

$$dgap(o_1, p_1) = A$$
$$dgap(o_2, p_1) = C \cup B$$
$$dgap(o_1, p_2) = A$$
$$dgap(o_2, p_2) = C$$

Furthermore, we can refine the granularity of modules: instead of considering ontologies as modules, we can consider the elements of these ontologies as modules. To this end, we could exploit *comparison operators*, else called *diff* or *delta* (Δ) operators, like those proposed in (Noy & Musen, 2002), (Zeginis, Tzitzikas & Christophides, 2007). For example, consider the case illustrated in the following Figure.

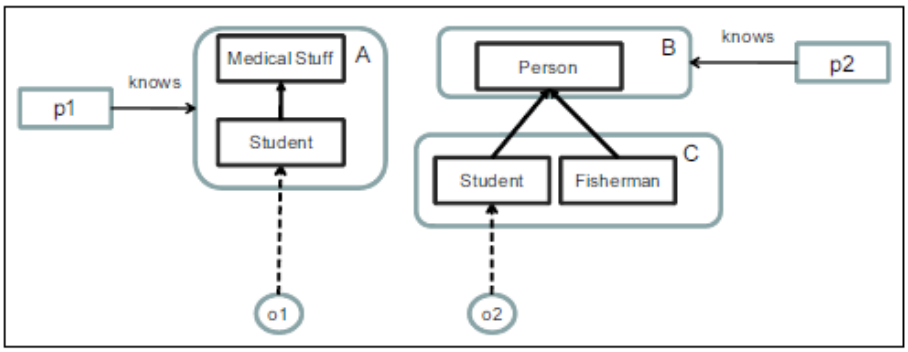


Figure 14 Example with more refined gaps

If we assume that equality of concept names implies equality of concepts, we can define:

$$dgap(o_1, p_1) = \Delta(A \rightarrow A) = \emptyset$$
$$dgap(o_2, p_1) = \Delta(A \rightarrow C \cup B)$$
$$dgap(o_1, p_2) = \Delta(B \rightarrow A)$$
$$dgap(o_2, p_2) = \Delta(B \rightarrow C \cup B)$$

According to this view, a gap comprises change operations. A detailed treatment of such cases is described at (Zeginis, Tzitzikas & Christophides, 2011).

An alternative approach to defining fine grained gaps that consist of modules (not change operations) is also possible. The key observation is that *instanceOf* and *isA* relations are special kinds of dependencies; actually, they carry more meaning than a plain dependency relation. This means that an *isA* hierarchy could be construed as a dependency graph in our framework (where each subclass depends on its superclasses). In the example of the previous Figure, this means that we have the dependencies o_2 < {instanceOf} Student < {isA} Person. Under this perspective, Gap(o_2, p_2) = {Student}. It follows that according to this view, profiles, as well as intelligibility gaps, can contain all kinds of RDF elements.

So far we have considered gaps that comprise sets of modules. The dependency types that participate to the computation of gap could also be returned, as they convey extra meaning which could be exploited and recorded (e.g. in the manifest file of XFDU). For instance, we can define gaps as sets of paths where a path is a sequence of (depType, module) pairs, or RDF triples of the form (subject, predicate, object), indicating the specializations of the ontology that are required. In the example of Figure 14, where

$$Gap(o_2, p_2) = \{\texttt{Student}\}$$

A more informative gap would be:

$$IGap(o_2, p_2) = \{\texttt{instOf Student subclassOf Person}\}$$

In a triple form we could write:

$$IGap(o_2, p_2) = \{\{\texttt{o}_2\ \texttt{instOf Student}\},\ \{\texttt{Student subclassOf Person}\}\}$$

11.3 BNODE ANONYMITY AND GAPS

A rather peculiar but quite flexible feature of RDF is that it allows the representation of *unnamed* nodes, else called *blank nodes* (for short *bnodes*), a feature that is convenient for representing complex attributes (e.g. an attribute address as shown in the next Figure) without having to name explicitly the auxiliary node that is used for connecting together the values that constitute the complex value (i.e. the particular street, number and postal code values).

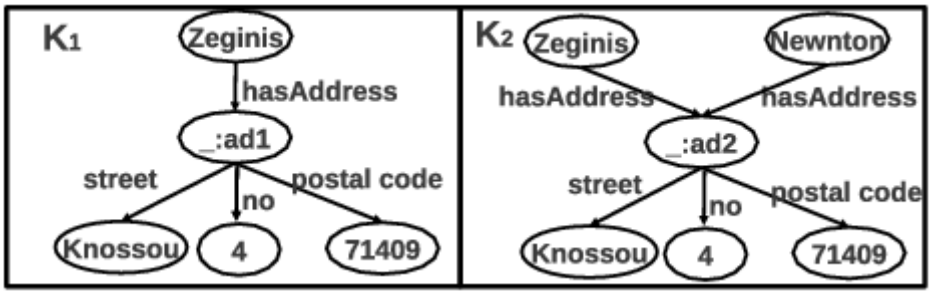


Figure 15 Two Knowledge Bases with bnodes

A recent paper (Mallea, Arenas, Hogan, Polleres, 2011) that surveys the treatment of bnodes in RDF data, proves that blank nodes is an inevitable reality. Indicatively, and according to their results, the data fetched from the "hi5.com" domain consist of 87.5% of blank nodes, while those from the "opencalais.com" domain, which is part of LOD (Linked Open Data) cloud, has 44.9% bnodes. The authors of that paper also state that the inability to match bnodes increases the delta size and does not assist in detecting the changes between subsequent versions of a Knowledge Base.

This aspect is important also for digital preservation. For instance, if the recording of provenance is done in RDF/S (e.g. as described in WP24), then the usage of bnodes is beneficial (no need to invent names for naming the nodes that correspond to event or sub activities).

The ability to exploit bnode anonymity for reducing the delta is therefore crucial for aiding agents (human or artificial) to understand the gap (difference) between two different models. In (Tzitzikas, Lantzaki & Zeginis, 2012) we showed how we can exploit bnode anonymity to reduce the delta size when comparing RDF/S Knowledge Bases . We proved that finding the optimal mapping between the bnodes of two Knowledge Bases , i.e. the one that returns the smallest in size delta regarding the unnamed part of these Knowledge Bases, is NP-Hard in the general case, and polynomial in case there are not directly connected bnodes. To cope with the general case we presented polynomial algorithms returning approximate solutions.

The experimental evaluation showed that in real datasets with no directly connected bnodes, a signature-based algorithm was two orders of magnitude faster than an algorithm based on the Hungarian algorithm (less than one second for Knowledge Bases with 6,390 bnodes), but yielded up to 0.34 times (or 34%) bigger deltas than the Hungarian, i.e. than the optimal mapping. For checking the behavior of the algorithms in Knowledge Bases with directly connected bnodes, we created synthetic datasets, over which we compared the two algorithms. The signature-based algorithm requires only 10.5 seconds to match 153,600 bnodes. The following Figure summarizes the main results.

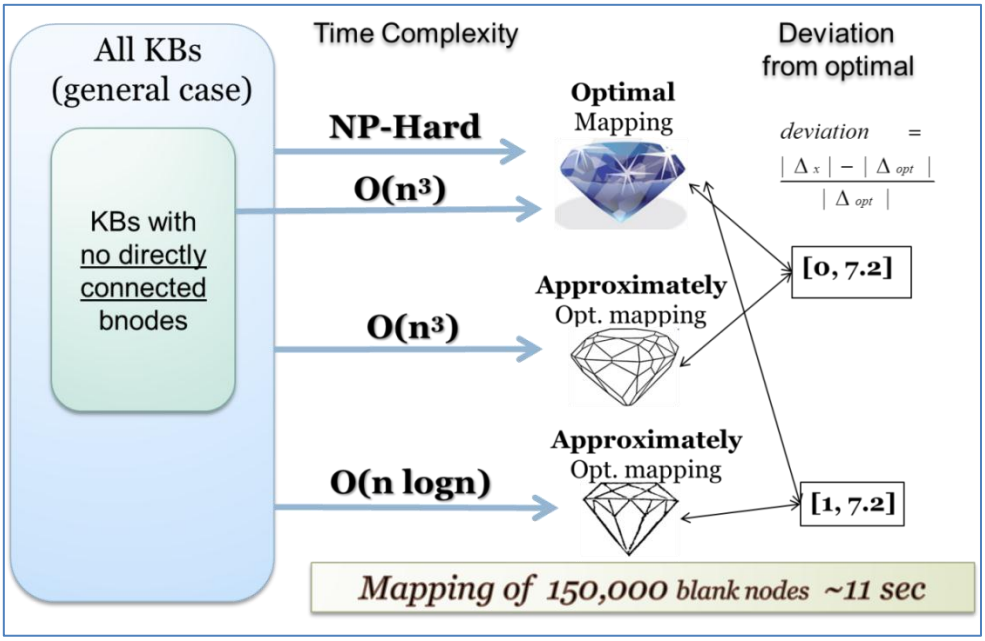


Figure 16 BNodeDelta: Synopsis of the theoretical and experimental Results

More information is available in the following papers:

Yannis Tzitzikas, Christina Lantzaki, Dimitris Zeginis: Blank Node Matching and RDF/S Comparison Functions. International Semantic Web Conference (1) 2012: 591-607
Christina Lantzaki, Yannis Tzitzikas, Dimitris Zeginis: Demonstrating Blank Node Matching and RDF/S Comparison Functions. International Semantic Web Conference (Posters & Demos) 2012

Furthermore, we have developed a web system that allows users to compare their knowledge models. Some screendumps are in order. The web system is accessible from the URL: http://www.ics.forth.gr/isl/BNodeDelta.

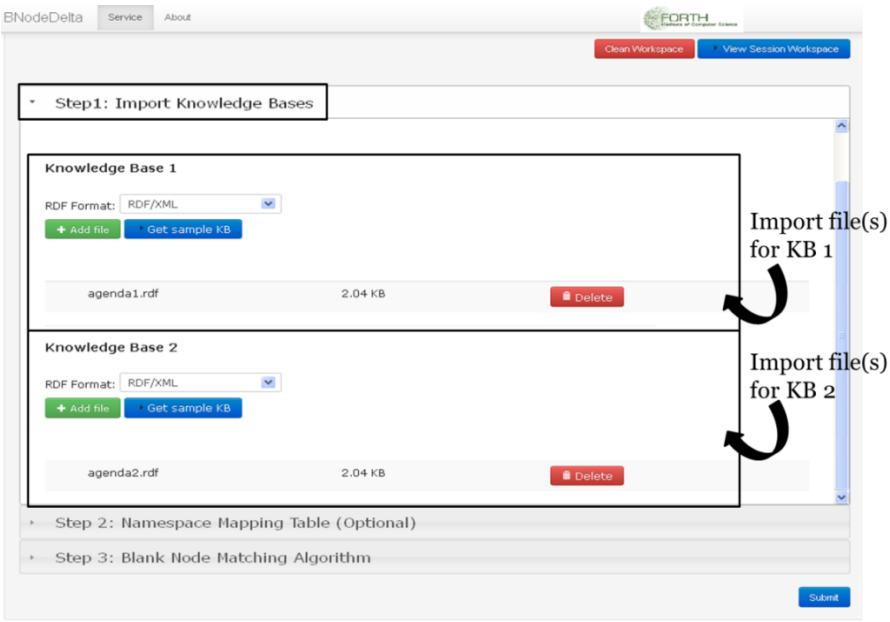


Figure 17 BNodeDelta: Selecting the Knowledge Bases to be compared

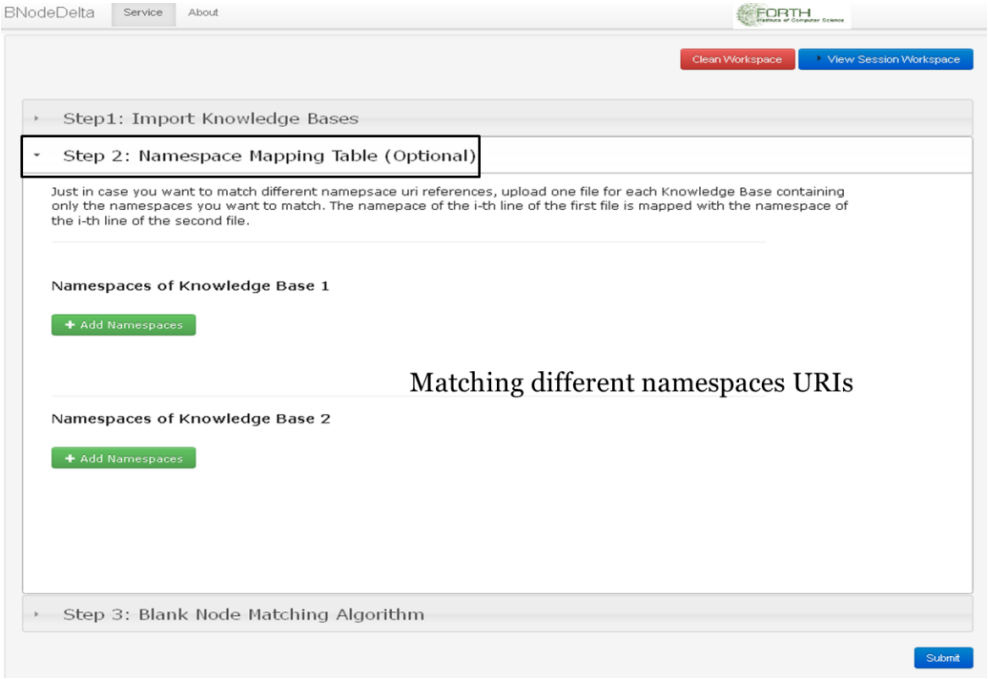


Figure 18 BNodeDelta: Defining the Mapping of the Namespaces

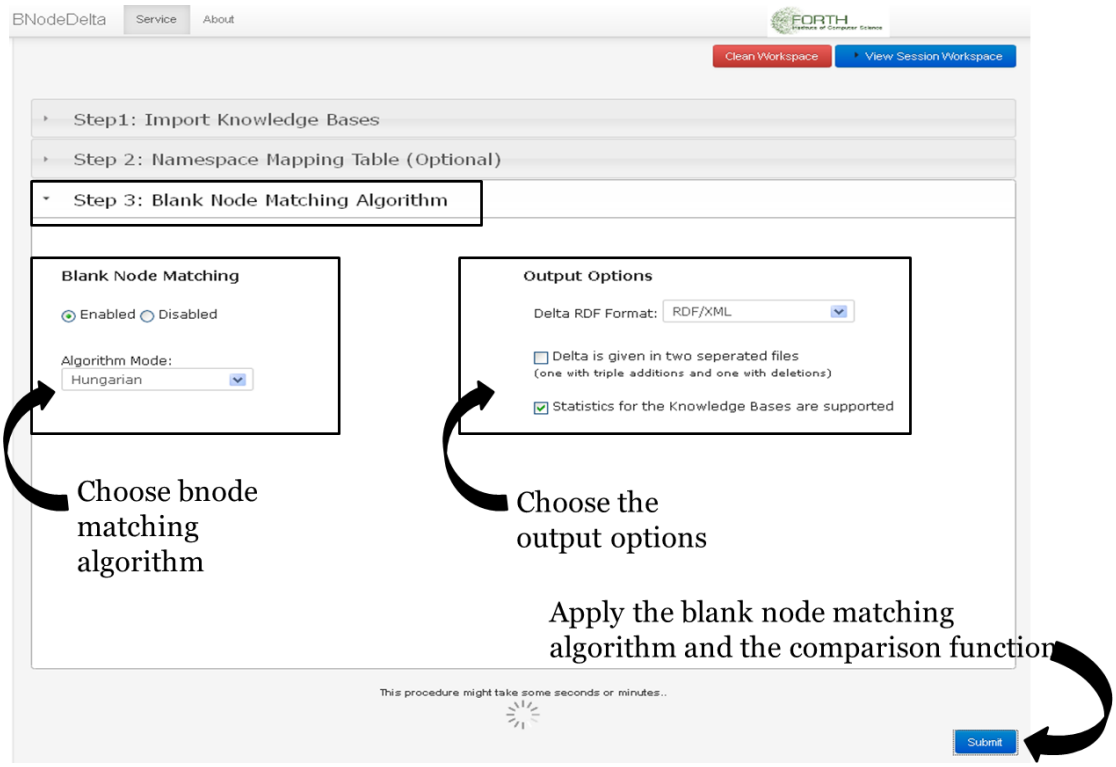


Figure 19 BNodeDelta: Selection of the bnode matching algorithm

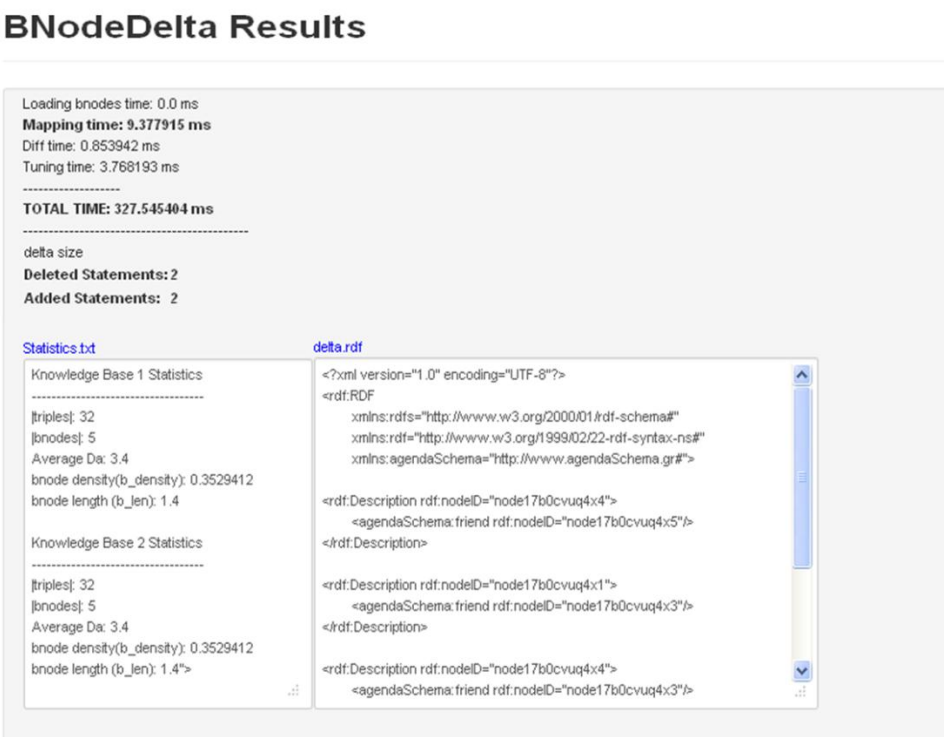


Figure 20 Results of BNodeDelta

12 THE PROTOTYPE SYSTEM (EPIMENIDES) FOR DEPENDENCY MANAGEMENT

For demonstration purposes we have implemented (and keep improving) a web application named Epimenides[16] for end users, based on the RDF/S implementation approach which was described in Section 9.1.

The Use Case diagram that provides an overview of the supported functionality is given in Figure 21. The application can be used by several users (e.g. end-user, curator), and each one can build and maintain his/her own profile. To be flexible, a gradual method for the definition of profiles is supported. The main scenario is described in the sequel.

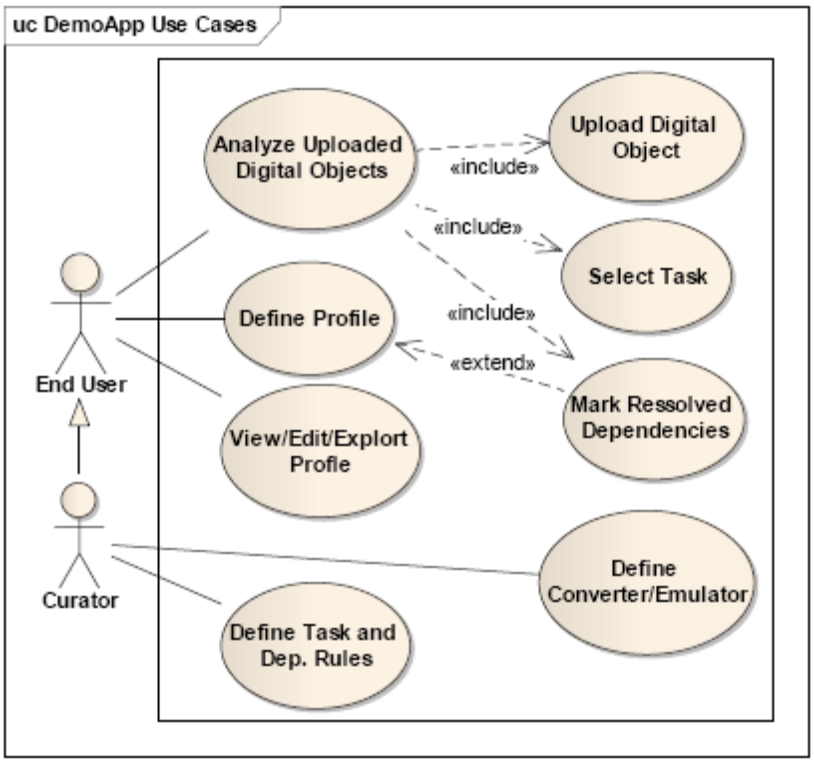


Figure 21 Use Case Diagram of Epimenides

After login, the user can upload a digital object and select the task whose performability wants to check. The system then checks the dependencies and computes the corresponding gap. The curator can define new tasks to the system. To identify the dependencies of the uploaded objects, the system exploits the extension of the object (like .pdf, .doc,.docx), and its Knowledge Base already stores the dependencies of some widely used file types. The identified dependencies are then shown to the user. The user can then add those that (s)he already has, and this is actually the method for defining the profile gradually. In this way (s)he does not have to define his profile in one shot. The system stores the profiles of each user (those modules marked as ”I have them”) to the RDF storage. The profiles are in different graph spaces for each user/profile.

[16] Epimenides of Knossos (Crete) (Greek: Επιμενίδης) was a semi-mythical 7th or 6th century BC Greek seer and philosopher-poet.

12.1 THE KNOWLEDGE BASE OF THE PROTOTYPE AND USAGE EXAMPLES

Figure 22 shows the architecture of the system's Knowledge Base. For the representation of the modules the Knowledge Base contains all the MIME[17] media types expressed as a *subClassOf* hierarchy (this hierarchy is shown in the left of Figure 23). The dependency rules are also stored in the Knowledge Base as strings of SPARQL queries. Finally, the Knowledge Base also contains information about tasks.

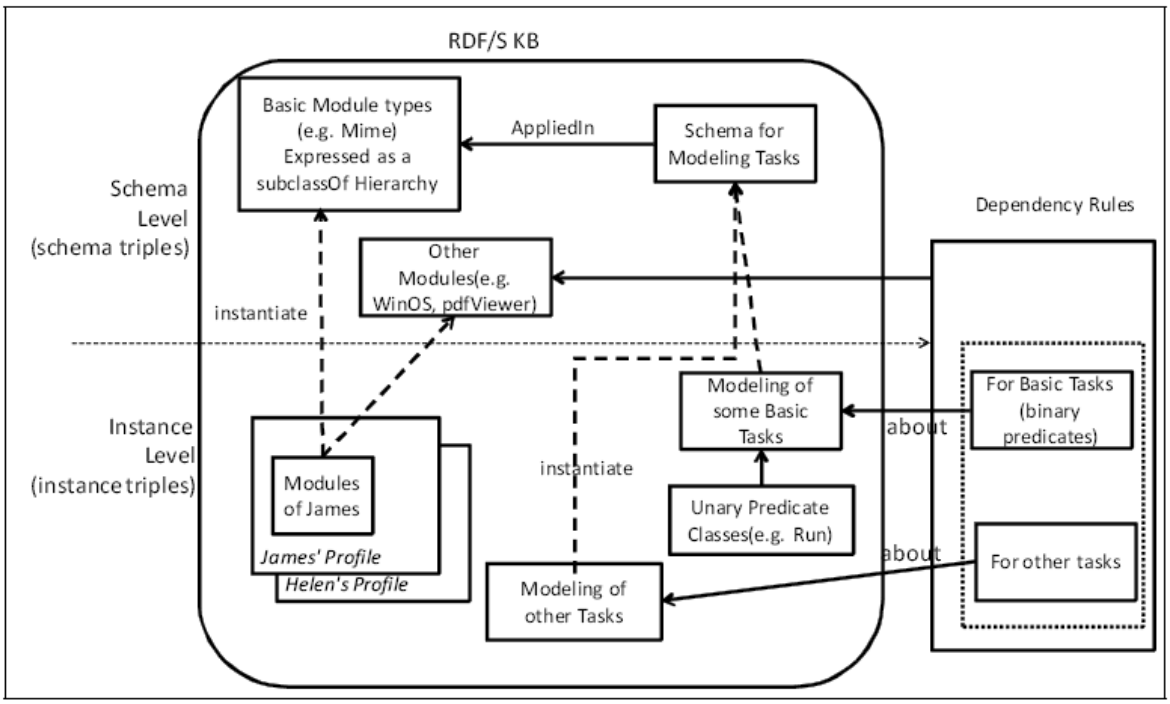


Figure 22 Architecture of the Knowledge Base

To explain the structure of the Knowledge Base we shall use an example that is illustrated in Figure 23. Suppose a user that his/her profile contains only the module *WinOS* and he uses the application for first time. The user uploads a file, say *f*, and the system by its file type extension (suppose .exe), or by analyzing the contents (e.g. by using tools like Jhove or JMimeMagic library), can realize that the uploaded file is an executable file, and that belongs to the "*application/octet-stream*" MIME type and consequently to the octet-stream class of the Knowledge Base (as shown in Figure 23). To achieve this for the first case (using the file extension) any MIME type class in the Knowledge Base has the property *hasExtension*. In this way the Knowledge Base contains the triple (<octet-stream>, <hasExtension>, ".exe").

By knowing the class that models the type of *f*, the system can find the tasks that usually make sense to apply to the uploaded file by the property *appliedIn* of the Knowledge Base, which has domain a task and range a MIME type.

[17]Multipurpose Internet Mail Extensions (MIME) is an Internet standard that extends the format of email

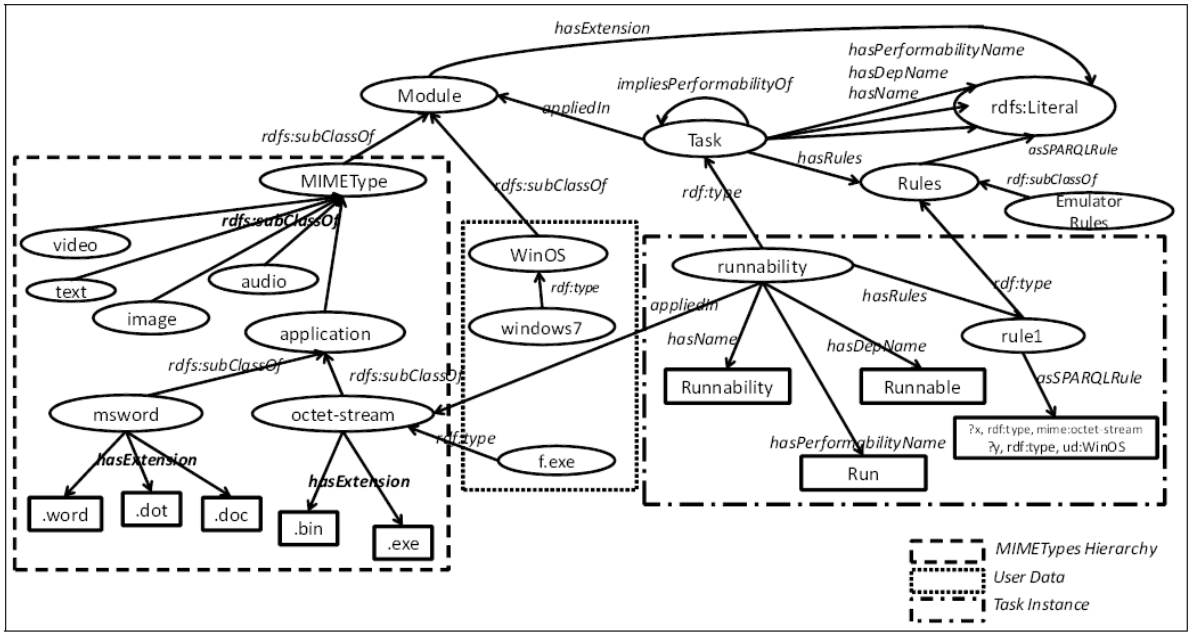


Figure 23 The contents of the RDF/S Knowledge Base of the Prototype

The user can select one of the retrieved tasks and the next step for the system is to check if this task can be performed. This can be done by the task-performability service (as we have described in Section 0).

To complete the performability service, the system produces the Operational Knowledge Base (for short OKB) in which new classes and new properties are created and populated. The name of new classes and properties are determined by the properties of Knowledge Base *hasPerformabilityName* and *hasDepName*respectively. The OKB is a superset of the Knowledge Base. Specifically, it contains the results of the application of all rules that the Knowledge Base contains. Whenever the Knowledge Base changes (e.g. user uploads a new file) the OKB is updated, and all rules applied again. In this way, query answer can indeed support the desired services for task performability, taking also into account the emulators and the converters. In Figure 24 you can see the OKB for our example. Its right side shows how the User Data are changed when the OKB is produced (with bold are represented the new produced resources).

In our case suppose that the user has selected the task with name "Runnability". The class Run and the property Runnable have already been created in the OKB, as Figure 24 shows. Now the system using the property *hasPerformabilityName*, issues the query Run(f) in the profile of the user (in OKB). Obviously, the answer of the query in our example is true, as you can see in Figure 24; therefore the system informs the user that this task can be finally performed. In case the selected task could not be performed, the system should inform the user for the missing dependencies.

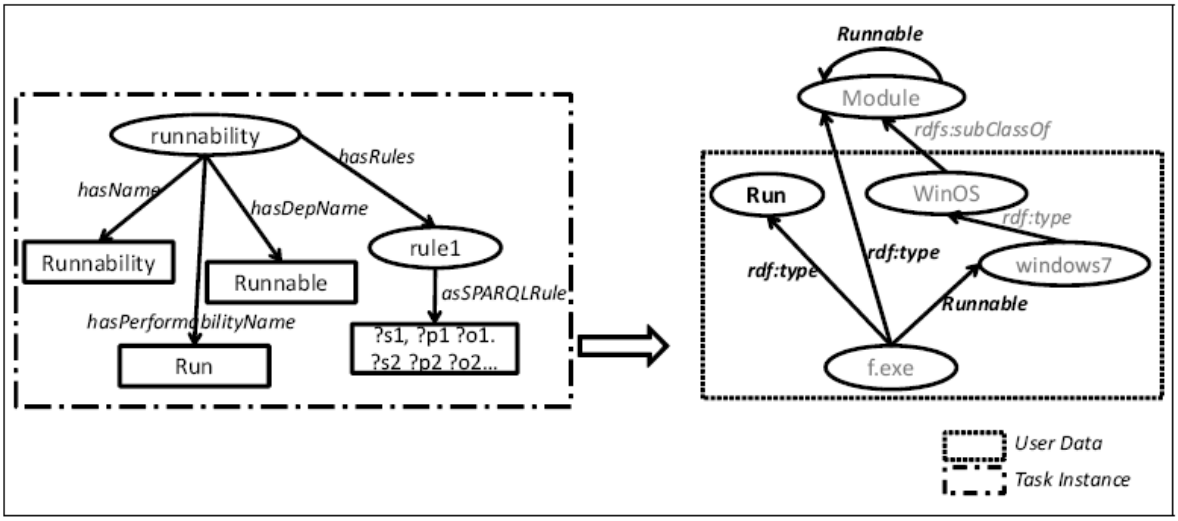


Figure 24 The notion of "Operational Knowledge Base" (OKB) of the prototype

To determine missing dependencies that are required for performing the selected task, the system uses the Dependency Rules that are stored in the Knowledge Base. Specifically, at first it retrieves from the property asSPARQLRule of each one rule of the selected task the direct dependencies, which actually is a set of atoms. These atoms are shown to the user and (s)he can either select that (s)he already has a corresponding fact, or (s)he can ask the system to show how an atom can be satisfied. In the second case the system explores the Knowledge Base for rules (including rules for emulators/converters) that have as head the selected atom. The above procedure is repeated for the new rules. In this way, a gradual expansion is created, as the user gradually explores the possible paths.

The prototype is accessible from the URL: http://www.ics.forth.gr/isl/epimenides[18] and anyone can use it to load a demo profile.

Currently the Knowledge Base that we have created for the Demo User contains 657 Module Types, including 647 MIME Type Modules. It also contains information about three tasks (readability, runnability, rendering). Furthermore, for each of the 647 Mime Types the Knowledge Base contains extra information (e.g. the extension of a mime type). As regards the dependencies of the tasks, 24 rules have been specified. In total, the Knowledge Base contains 2,225 RDF triples, while after the application of the rules they become 2,235. Note, as we have already mentioned, that a user can enrich the Knowledge Base by adding his/her own Module Types, Tasks and Rules using the prototype system.

12.2 AIDING THE INGESTION OF TASKS

Above we have described the use cases for end users. The job of the curator could also be assisted by providing a simple method for adding tasks and modeling the corresponding dependencies. This is related to the use case named "Define Task and Dep. Rules" in Figure 21. The curator can enrich the system to support extra tasks and rules. He should provide some input and the application produces the required rules. Specifically, the curator should provide:

- The unary predicate that denotes the performability of the task (e.g. Edit),
- the MIME type/s that can be applied in this task (e.g. text/plain),
- the module type/s that is/are required for the selected MIME Type (e.g. Text Editor),
- and optionally the task, whose performability can be implemented by the new task (e.g. Readability).

[18] Also accessible from : http://139.91.183.63:8080/epimenides/

12.3 MENU OPTIONS

The menu of the application, as you can see in the next Figure, is separated in 3 sections. The first contains the main option of the application: "Upload Digital Object".

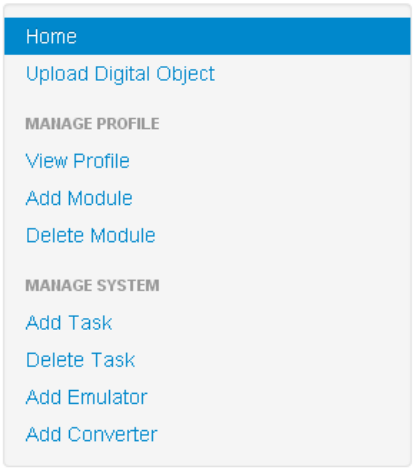


Figure 25 Menu Options

The "MANAGE PROFILE" section contains options available to any user (simple users), that may belong to an organization or a group. The user can add/delete modules to his profile, where a profile is a list of modules.

The "MANAGE SYSTEM" section contains options for a curator user. Such a user has also the ability to define Tasks, Emulators and Converters for the users that belong to the same organization/group with him/her. To properly add a Tasks/Emulator/Converter one has to provide extra information from which the application will produce the required rules. A simple user can add to his/her profile an emulator X, only if it has been properly defined from a curator user (and consequently the application has produced the required rules).

12.4 SCREENS FROM THE PROTOTYPE

Figure 26 shows the first screen of the system, where you can make a login to load your personal profile or you can load some demo profiles.

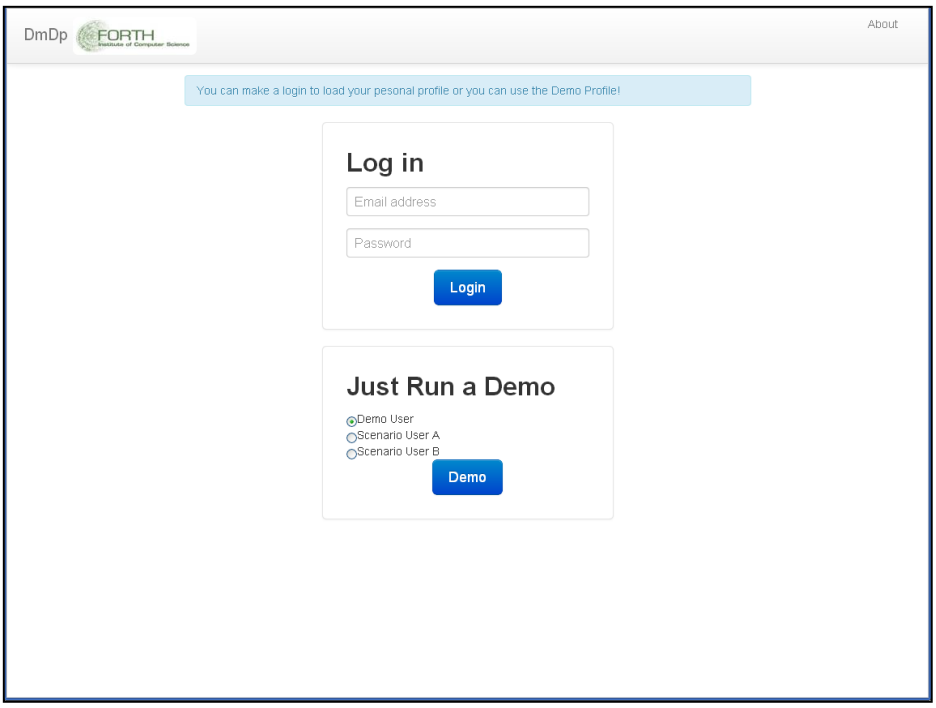


Figure 26 Load your personal profile or use a demo profile

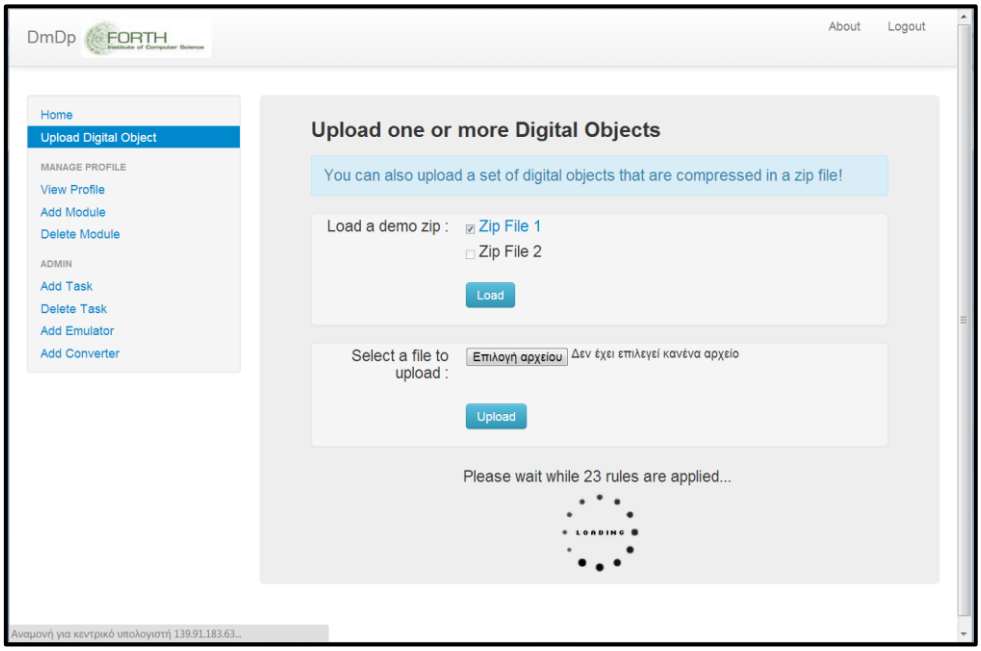


Figure 27 Upload digital objects to check the performability of them

Figure 27 shows the first screen that allows the user to upload a file (atomic or a zipped collection of files).

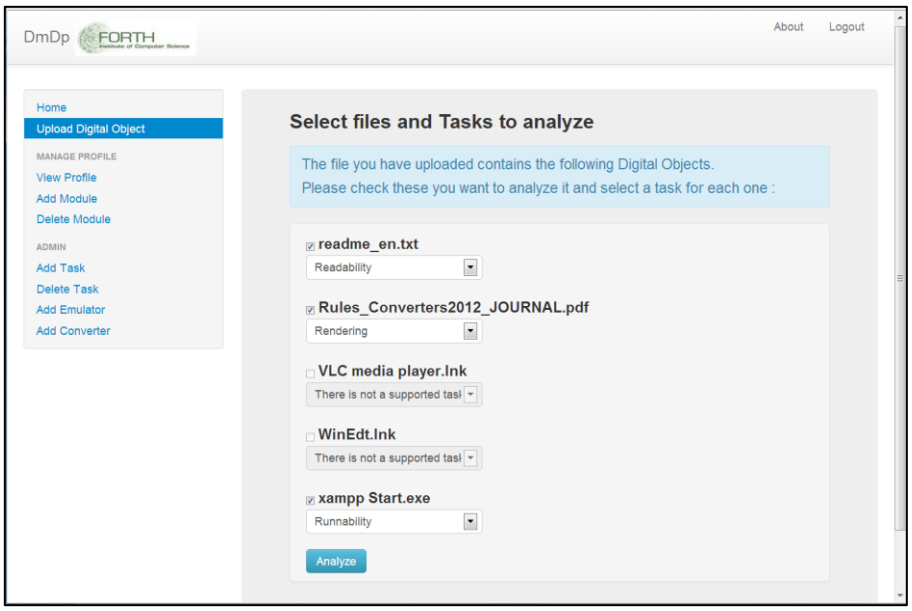


Figure 28 System finds the tasks that usually make sense to apply to the uploaded digital objects

The system analyzes the contents of the zip file and for each of the included files it suggests a task. This is shown in Figure 28.

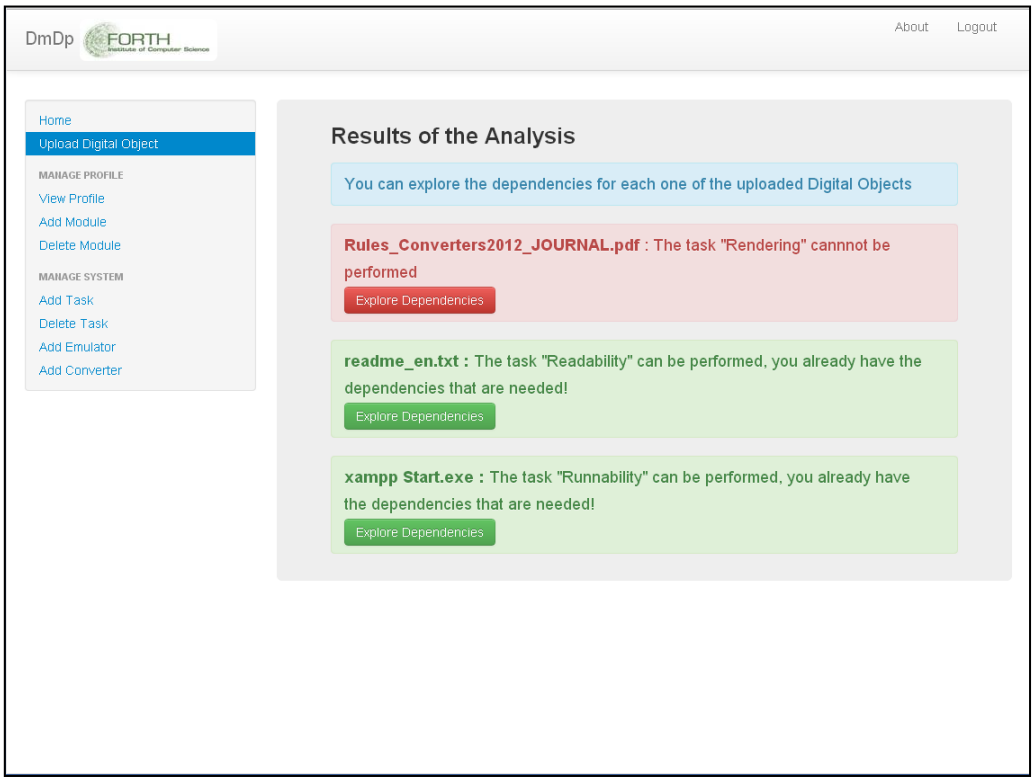


Figure 29 Results of the analysis.

Figure 29 shows the results of this analysis. We can see that the first file is in red because the selected task, i.e. Rendering, cannot be performed over that file. In contrast, the selected tasks for the other two files can be performed, and for this reason they are marked with green.

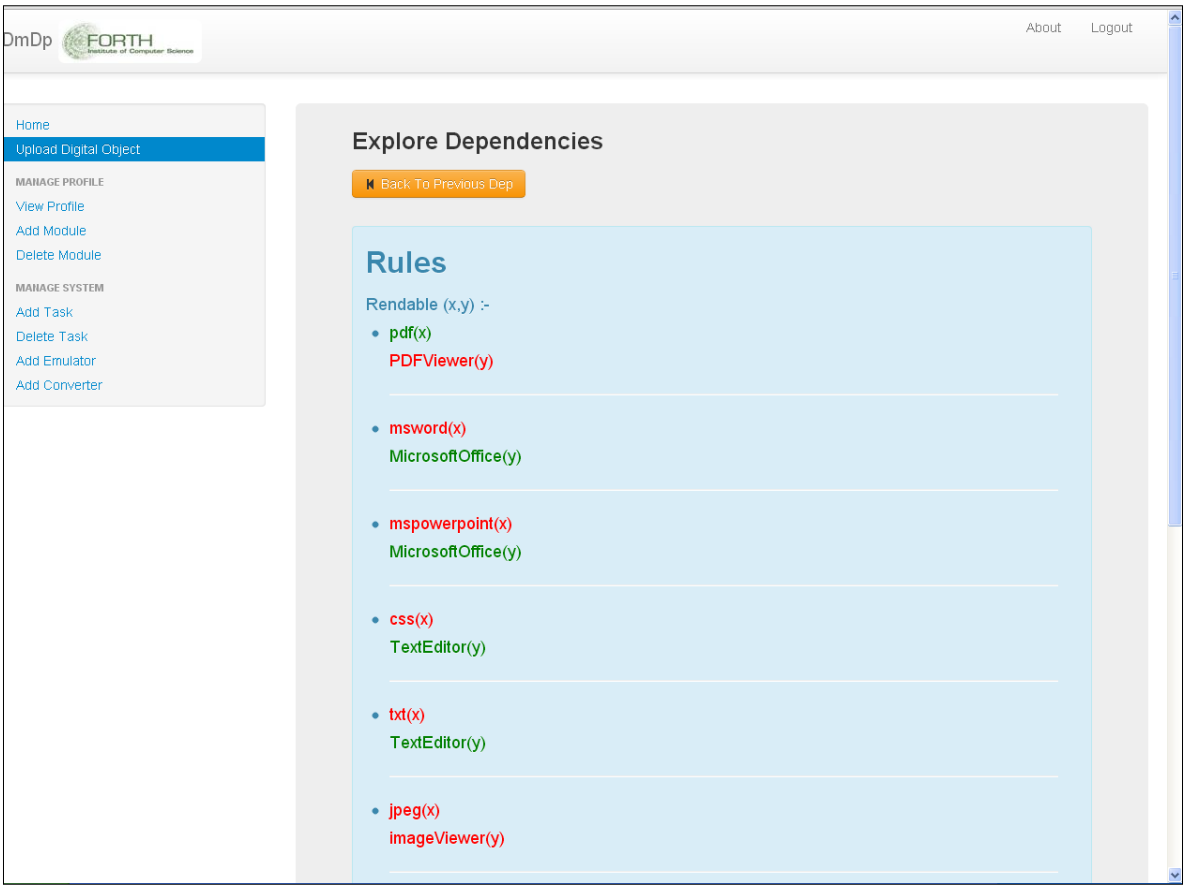


Figure 30 Explore Dependencies of a Task

The user can explore the dependencies for each one of the digital objects. For example, Figure 30 shows what happens if the user clicks to explore the dependencies of the "Rendering" task. We can see all the rules of the selected task that are available in the system. The atoms of each rule are green or red. Green atoms are available in the profile of the user, while the red are not.
Moreover, the user can click on an atom to explore the dependencies of this atom, so (s)he can see the rules or the facts of this atom.

12.5 TESTING THE PROTOTYPE

Various aspects of testing are described below.

12.5.1 Correctness Testing

The objective was to check that the implementation of the system behaves as expected by the theory.

Action(s):

We inserted the data and the rules used in the examples of this report, and we checked the results of the services.

Status and Results:

The tests were successful. The system behaves as expected.

12.5.2 Usability Testing

The objective was to check the usability of the system, specifically how easy it is for a user to understand the main concepts of the approach by using the system, and how the system per se is usable.

Action(s):
[a1] We prepared a short tutorial for the system. It is accessible from the wiki page of APARSEN[19].
[a2] We prepared some scenarios that we asked users to carry out using the system. It is accessible from the wiki page of APARSEN[20].
[a3] We formed a list of persons that could participate to this evaluation. The list included the participants of this WP plus persons from organization that are not members of APARSEN.

Participants from APARSEN

Person	Organization	Email
Rene van Horik	DANS	rene.van.horik@dans.knaw.nl
Barbara Bazzanella	University of Trento, Italy	barbara.bazzanella@unitn.it
Jinsongdi Yu (external user; members of the SCIDIP-ES project)	Jacobs University, Bremen, Germany	j.yu@jacobs-university.de
Emanuele Bellini	FRD	Bellini@rinascimento-digitale.it

[a4] We prepared a small questionnaire that the users had to answer after using the system. Specifically the users had to fill the following form: https://docs.google.com/forms/d/10DcuUV3uDI6nQ0QbAPfx4yYd-AqU-_enAGACnjmj4Ow/viewform
The questionnaire is also available in the appendix (Section 16.3) of this document

The results (so far) are positive. A short analysis of the feedback received so far is given in the Appendix. We plan to continue improving the system and getting feedback; therefore we do not report here detailed results.

[19] http://aparsen.digitalpreservation.eu/pub/Main/ApanWp25/demoUsersGuide.docx
[20] http://aparsen.digitalpreservation.eu/pub/Main/ApanWp25/UserExperience_short.docx

13 USE CASES FROM APARSEN PARTICIPANTS AND APPLICABILITY

Apart from the above testing, we decided to investigate the applicability of the methodology and the approach over the practices and systems of some partners, as a means to get feedback and also for communicating and explaining to them how this approach could bring benefits to them (this is described in the next Section).

Of course, this was just indicative. In an operational setting, the approach and the tool have to be adapted to the particular practices and tools that are in use by the organization at hand. Some comments about other tools that can be used together are described in Section 13.2.

13.1 SCENARIOS FOR DANS

DANS (Data Archiving and Networked Services, NL) aims at promoting sustained access to digital research data. For this purpose, DANS encourages researchers to archive and reuse data in a sustained manner, e.g. through the online archiving system EASY (http://easy.dans.knaw.nl). DANS also provides access, via NARCIS (http://www.narcis.nl), to scientific datasets, e-publications and other research information in the Netherlands. Apart from these, the institute provides training and advice, and performs research into sustained access to digital information.

In collaboration with DANS, we defined a number of scenarios that indicate where and how the dependency management approach could be used. The analysis yielded five scenarios, whose description follows. Then, we consolidate them, and describe them using the steps of the methodology that was introduced in Section 0.

Partner	DANS
Scenario Id	1
Scenario Title	**Checking File Format Compatibility (compliance or migratability) with Acceptable/Preferred File Formats during Ingestion**
Description	For a number of data types (tables, text, images, etc.), specific file formats are considered as durable at least into the near future. DANS maintains a list of *acceptable* and *preferred* formats. These lists are the basis for file format migration activities. The list that DANS currently uses[21] follows:

[21] Taken from http://www.dans.knaw.nl/sites/default/files/file/EASY/DANS%20preferred%20formats%20UK%20DEF.pdf

Type of data	Preferred format(s)	Acceptable format(s)
Text documents	• PDF/A (.pdf)	• OpenDocument Text (.odt) • MS Word (.doc, .docx) • Rich Text File (.rtf) • PDF (.pdf)
Plain text	• Unicode TXT (.txt, ...)	• Non-Unicode TXT (.txt, ...)
Spreadsheets	• PDF/A (.pdf) • Comma Separated Values (.csv)	• OpenDocument Spreadsheet (.ods) • MS Excel (.xls, .xlsx)
Databases	• ANSI SQL (.sql, ...) • Comma Separated Values (.csv)	• MS Access (.mdb, .accdb) • dBase III or IV (.dbf)
Statistical data	• SPSS Portable (.por) • SAS transport (.sas) • STATA (.dta)	• R (*)
Pictures (raster)	• JPEG (.jpg, .jpeg) • TIFF (.tif, .tiff)	
Pictures (vector)	• PDF/A (.pdf) • Scalable Vector Graphics (.svg)	• Adobe Illustrator (.ai) • PostScript (.eps) • PDF (.pdf)
Video	• MPEG-2 (.mpg, .mpeg, ...) • MPEG-4 H264 (.mp4) • Lossless AVI (.avi) • QuickTime (.mov)	
Audio	• WAVE (.wav)	• MP3 AAC (.mp3) (**)
Computer Aided Design	• AutoCAD DXF version R12 (.dxf)	• AutoCAD other versions (.dwg, .dxf)
Geographical Information	• MapInfo Interchange Format (.mif/.mid)	• ESRI Shapefiles (.shp and accompanying files) • MapInfo (.tab and accompanying files) • Geographic Markup Language (.gml)

(*) under investigation
(**) please contact DANS before depositing MP3 audio files

Applicability

If the converters (or emulators) that are in use by DANS for carrying out the migration activities, are registered in a system like the one that we propose, then the system can be exploited not only for checking whether a newly ingested file is in an *acceptable/preferred* format, but also for checking whether it is *migratable* to one preferred or acceptable format using the migration/emulation software that DANS uses and has registered.

Implementation

To realize this scenario, one has to define a profile (say profile_DANS) that consists of:

[a] The list containing the software that DANS uses for managing a file having an acceptable/preferred file format (e.g. AcrobatReader for rending PDF files, VLC for playing mpg/mpeg/mp4/avi/mov files). At least one software per format is required.

[b] For each file type in the list of acceptable/preferred list, a task has to be associated (the one usually applicable on such file types) and the dependencies for that task have to be delivered in a way so that that they are satisfied by the list of software described in [a] (e.g. *Render(X) :-*

	pdfFile(X), pdfViewer(Y)). [c] The list of tools that DANS uses for migration/conversion purposes (e.g. *docxToPdfConverter(doc2pdf)*).
Use Case	After having done the steps that were just described, the end user (or archivist) could just use the system. Whenever he uploads a file, the system prompts the applicable task and directly informs the user if it is in an acceptable format or migratable to an acceptable format using the software that DANS has.
Without this approach	It is difficult for a curator to determine that (a) an archived dataset is formatted in a durable format and (b) to have an overview of the applicable file format migration procedures that can be carried out to convert a file into a preferred file format (given the fact that the list of preferred file formats will change over time as file formats might become obsolete).

Partner	DANS
Scenario Id	2
Scenario Title	**Updating the List of Preferred/Acceptable Formats and Detecting the Consequences of Obsolete Formats**
Description	As the usability and durability of file formats tend to change over time, for DANS it is important to periodically monitor and assess the applicability of the list of preferred formats and if it is necessary to replace a file format that became obsolete with a new one. Also new preferred formats can be introduced in the list. Specifically, say every year, the specifications on the list of preferred file formats have to be assessed based on a number of criteria (e.g. discussions in literature, consensus of organizations that provide guidelines in this field, etc.)
Applicability & Implementation	[a] To add a new format in the list of acceptable/preferred file formats, the archivist can just register it to the repository (see scenario 1, steps [a] and [b]). The check performed at ingestion time will then function as expected (i.e. in accordance with the revised list of acceptable formats). [b] Before deleting a file format (or managing software) from the list of acceptable/preferred file formats (or available software respectively), the archivist can check the impact of that deletion, i.e. the impact that this deletion will have on the performability of tasks over the archived files. Recall the discussion on Section 9.1 about the Risk Detection. [c] To delete a file format (or managing software) from the list of acceptable/preferred file formats (or available software respectively), the archivist can just delete the corresponding entries from the system. After doing so, the checking at ingestion time will function as expected, i.e. in accordance with the revised list of acceptable formats.
Without this approach	• It is difficult to identify all the consequences of file format's obsolescence. • It is also difficult to identify what will happen if managing software[22] is lost or will become obsolete

Partner	DANS
Scenario Id	3
Scenario Title	**Assistance in Planning and Performing Migration to Acceptable/Preferred File Formats**

[22] Software that is able to convert to/from a preferred file format

Description	Research datasets are formatted in a number of formats as submitted to the data archive by the depositors. The data archive stores and manages these datasets in the format as submitted by executing so-called "bit-preservation" (more about bit preservation in a next scenario). The data archive archives all formats but only commits itself to the long-term usability of file format that are formatted according to so-called preferred formats, described in the previous scenarios. In two situations a file format migration is required: (1) as part of the *ingest procedure* files not formatted according to the preferred file format are migrated to a suitable preferred file format. (2) in case in the future a *preferred file format becomes obsolete* the files have to be migrated to this new format. The migration process requires tools. Quality features of these tools are: speed, accuracy, level of completeness, and usability of the tool.
Applicability & Implementation	The dependency management approach can show to the archivist whether a file format migration is possible using the software that DANS has (recall Scenario 1). Also since a migration can be performed with different tools (or execution plans in general), the proposed system can assist the archivist by showing to him/her, the possible actions/tools and this can be achieved by exploring the dependencies that the system offers (recall the screens of the system that offer exploration services).
Without this approach	It is difficult for a human to identify all possible migration plans .

Partner	DANS
Scenario Id	4
Scenario Title	**Assistance in Planning and Performing Migration to Acceptable/Preferred File Formats**
Description	Despite the fact that research data archives are aimed at the durable access of datasets, there are cases where specific software is required to be able to use the datasets. For such cases, activities have to be undertaken to guarantee that this software is usable over time. Software preservation involves much more dependencies, than research data preservation (e.g. changing operating systems, proprietary source code, etc.). Research data archives currently have no general accepted software preservation strategy.
Applicability & Implementation	The examples of the current deliverable have demonstrated this with various examples (recall the task of runability and compilability).
Without this approach	It is difficult and time consuming to plan software migration.

Partner	DANS
Scenario Id	5
Scenario Title	**Bit Preservation (ability to test corruption)**
Description	The bit preservation scenario involves activities to guarantee that digital objects do not become corrupted. This means not one bit is changed over time. Thus the integrity of the data objects is guaranteed. This can be achieved by creating checksums on the occasion where the digital objects are ingested in the data archive and periodically check whether the checksum is still valid. Dependencies in the scenario are the strength of the checksum procedures and the time interval the checksum is checked as part of the bit preservation activities.
Applicability &	If checksums are supposed to be used for ensuring that the data have not been corrupted, then an

Implementation	archive can model as task the computation of checksums for being sure that in the future the archiving organization will be able to recomputed them and compare them with the stored ones. Note that there are several tools for computing checksums[23]. We can say that this is a special case of scenario 4.
Without this approach	It is difficult and time consuming to plan software migration

Consolidation of the Scenarios

Here we consolidate the key points of the above scenarios and Table 5 describes them using as gnomon the steps of the methodology introduced in Section 0.

General Step	Specialization for the case of DANS
1. Identify the desired tasks and objectives	The desired tasks are: **[a]** those related to the list of the acceptable/preferred formats, e.g. render (for pdf, txt, pictures), play (for video, aurio), getTheRelationalModel (for spreadsheets, databases), etc. **[b]** those related to the runability of DANS software (including computability of checksums).
2. Model them and their dependencies (check hierarchy)	**[a]** Using the list of software described in Scenario 1[a]. Moreover the dependencies of the runability of the tools that DANS uses for migration have to be modeled. **[b]** Model the software dependencies that are required for running the software that DANS uses. In general the modeling required is quite simple, analogous to the examples given in the deliverable.
3. Specialize the rule-based approach	It seems that there is not need for any particular specialization.
4. Identify Ways to capture dependencies (manual, auto, …)	The file types are detected automatically (when one uses the upload feature of the web application). For applying this approach in a big collections of files, various tools could be used for automating this process l (more in Section 13.2). Surely, in an operational setting the proposed functionality could extend or complement the functionality of the ingestion procedures of the systems that DANS currently uses.
5. Customize use and exploit the dependency services	For demonstration purposes this can be done using the web application, i.e. no need for customization or integration with the other systems of DANS. However, in an operational setting the processes and systems of DANS should be considered. Applicability is discussed in more detail in Section 13.2.
6. Evaluate	For the needs and the capacity of APARSEN, this can be done using the web application.

Table 5 Application of the Methodology for the case of DANS

[23] http://en.wikipedia.org/wiki/Checksum#Checksum_tools.

13.2 RELATED TOOLS AND APPLICABILITY

In an operational setting, the approach and the tool have to be adapted to the particular practices and tools that are in use by the organization at hand. Roughly, the approach could be applied:

- By **using, configuring and extending** the **web prototype system**. Moreover the RDF comparison tool (described in Section 11.3) could be plugged in order to offer more refined gaps in case where this is possible.
- By using the functionality of the prototype system **through an API**, which is used for extending the systems that an organization has already in place.
- By **extending an existing repository management system**. For instance, Fedora[24] is a widely used repository management system for digital objects that provides tools and interfaces for the creation, ingest, management, and dissemination of content. The key abstraction, is the Fedora Digital Object (FDO). An FDO has an identifier (PID), Dublin Core metadata , and Datastreams (the actual content). A Datastream can be of any MIME-type and it can be managed locally (in the Fedora repository), or by external data sources (in that case it referenced by its URL). FDOs can be connected through relationships forming a network of digital objects, and these relationships are stored as metadata in digital objects within special Datastreams. The Fedora repository service automatically indexes all the relationships creating a graph of all the objects in the repository and their relationships to each other. The user can then make queries (e.g. SPARQL queries) to this graph and take results of the repository content. Fedora has also the ability to associate the data in a FDO with Web services to produce dynamic disseminations, where a dissemination is a view of an object produced by a service operation (i.e. a method invocation) that takes as input one or more datastreams of the object. Our approach could be implemented by extending the Fedora repository. This could be quite straightforward since the Fedora stores metadata using RDF/S and the set of relations that can be used for connecting objects is not limited. One way could be to extend the Fedora with a service that takes as input the MIME-type of the contents (datastreams) from the FDOs. This service using those MIME-type and having a basic mapping between the MIME-types and the tasks (e.g the MIME-type application/msword must be checked for the task *render*) can automatically define the required dependencies. The Knowledge Base that stores these could be the same with that of Fedora, or an external one. Surely the administrator of the repository could define various other tasks and dependencies using the approach that we have described.
- By **a provider of cloud services** who apart from offering storage services, offers various virtualization services and uses the methodology and techniques described in this document for realizing them. We could say that the long term vision is the *virtualization* of the basic preservation tasks. Just like the virtualization of *storage* that is currently offered by the cloud have made the life easier for the organizations that have to keep stored content, the virtualization of *rendering* and *software execution* would be an important contribution to digital preservation, and significant relief for the responsible organizations. To realize this virtualization, and preserve the performability of these tasks as operating systems, protocols, format change, the provider of such services needs a repository and services like those that we have described in this deliverable. This could be done either by the community itself collaboratively, or provided (and charged) by the private sector.

Some other tools and datasets which can contribute to an operational application of the approach are described below.

[24] http://fedora-commons.org/

13.2.1 PreScan (for aiding the ingestion of embedded metadata of files)

PreScan (Marketakis, Tzanakis & Tzitzikas, 2009) is a tool developed in the context of the EU project CASPAR. It can aid the ingestion of metadata. Figure 31 sketched the process that it carries out. In brief, this tool can scan the file system, extract the embedded metadata from the files, and transform them to RDF using the desired RDF schema. In the sequel, the resulting metadata could feed the Knowledge Base of the prototype system.

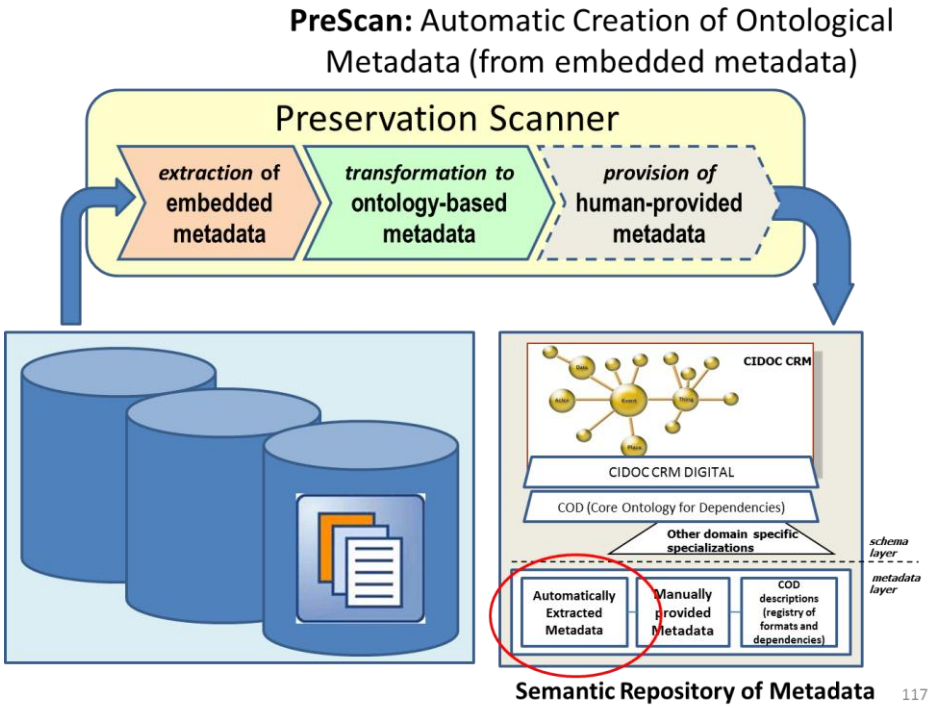


Figure 31 The system PreScan

13.2.2 The PRONOM Registry and its Contents

PRONOM[25] is an on-line information system about data file formats and their supporting software products. Originally developed to support the accession and long-term preservation of electronic records held by the National Archives. PRONOM holds information about software products, and the file formats which each product can read and write.

Linked Data PRONOM Lab[26] plans to make the registry data available in a Linked Open Data format. They created an RDF triplestore and a SPARQL Endpoint[27] is available. Also a draft vocabulary specification and accompanying documentation in RDF are available[28]. The used RDF properties are:

http://www.w3.org/1999/02/22-rdf-syntax-ns#type

[25] http://www.nationalarchives.gov.uk/PRONOM/Default.aspx
[26] http://labs.nationalarchives.gov.uk/wordpress/index.php/2011/01/linked-data-and-pronom
[27] http://test.linkeddatapronom.nationalarchives.gov.uk/sparql/endpoint.php
[28] http://test.linkeddatapronom.nationalarchives.gov.uk/vocabulary/pronom-vocabulary.htm

http://www.w3.org/2000/01/rdf-schema#label
http://reference.data.gov.uk/technical-registry/version
http://www.w3.org/2004/02/skos/core#altLabel
http://reference.data.gov.uk/technical-registry/formatType
http://purl.org/dc/elements/1.1/description
http://reference.data.gov.uk/technical-registry/byteOrder
http://reference.data.gov.uk/technical-registry/releaseDate
http://reference.data.gov.uk/technical-registry/withdrawnDate
http://reference.data.gov.uk/technical-registry/MIMETYPE
http://reference.data.gov.uk/technical-registry/PUID
http://reference.data.gov.uk/technical-registry/extension
http://reference.data.gov.uk/technical-registry/internalSignature
http://reference.data.gov.uk/technical-registry/byteSequence
http://reference.data.gov.uk/technical-registry/byteSequencePosition
http://reference.data.gov.uk/technical-registry/byteSequenceOffset
http://reference.data.gov.uk/technical-registry/byteString
http://reference.data.gov.uk/technical-registry/UTI
http://reference.data.gov.uk/technical-registry/developedBy
http://reference.data.gov.uk/technical-registry/maxByteSequenceOffset
http://reference.data.gov.uk/technical-registry/supportedBy
http://reference.data.gov.uk/technical-registry/XPUID
http://www.w3.org/2004/02/skos/core#note
http://reference.data.gov.uk/technical-registry/lossiness
http://reference.data.gov.uk/technical-registry/WAVE_Format_GUID
http://reference.data.gov.uk/technical-registry/compressionDocumentation
http://reference.data.gov.uk/technical-registry/mediaFormat

Table 6 The RDF properties of PRONOM

In comparison to our approach PRONOM is less powerful. PRONOM does not model the notion of *task*. Moreover, the notion of converter and emulator is not covered.

However, we could exploit some information from the PRONOM registry in order to enrich the Knowledge Base of our prototype. Specifically, as we have seen in the previous table, there are some common properties (i.e. extension and mime Type). For example from the extension of a file we can retrieve extra information from PRONOM registry by running the following SPARQL query:

```
select ?puid ?mime ?description ?developer where {
       ?s <http://reference.data.gov.uk/technical-registry/extension> "doc".
       ?s <http://reference.data.gov.uk/technical-registry/PUID> ?puid .
       ?s <http://reference.data.gov.uk/technical-registry/MIMETYPE> ?mime.
       ?s <http://purl.org/dc/elements/1.1/description> ?description.
       ?s <http://reference.data.gov.uk/technical-registry/developedBy> ?developer
}
```

The Persistent Unique Identifier (PUID) shown in the previous query is an extensible scheme for providing persistent, unique and unambiguous identifiers for records in the PRONOM registry. A unique PUID is assigned to each registry entry of the PRONOM.

However, the PRONOM registry contains only 101 mime types, while the Knowledge Base of the prototype already contains 647 different mime types, therefore the value of PRONOM is limited.

Moreover, there is a RDF repository (P2-Registry (Tarrant & Carr, 2009)) that links the PRONOM registry data with the DBpedia data. The P2-Registry is available at: http://p2-registry.ecs.soton.ac.uk/ . This registry provides open access to all the data contained within, as well as services including a SPARQL endpoint[29] and RESTful

[29] http://p2-registry.ecs.soton.ac.uk/SPARQL/

HTTP services. Data is currently available in XML and RDF formats, an HTML interface is not currently proposed other than to offer an explanation of the services available.

14 CONCLUDING REMARKS

Each interoperability objective or challenge (like those described in APARSEN D25.1 *Interoperability Objectives and Approaches*) can be considered as a kind of demand for the *performability of a particular task* (or tasks), e.g. the task of exchanging data between two systems, the task of performing on the received data a certain operation, etc. However, each task for being performed has various prerequisites (e.g. operating system, tools, software libraries, parameters, etc). We call all these *dependencies*.

The definition and adoption of standards (for data and services), aids interoperability because it is more probable to have (now and in the future) systems and tools that support these standards, than having systems and tools that support proprietary formats. From a dependency point of view, standardization essentially reduces the dependencies and makes them more easily resolvable; it does not vanish dependencies.

In all cases (standardization or not), we cannot achieve interoperability when the involved parties are not aware of the dependencies of the exchanged artifacts. However, the ultimate objective is the ability of performing a task, not the compliance to a standard. Even if a digital object is not compliant to a standard, there may be tools and processes that enable the performance of a task on that object. As the scale and complexity of information assets and systems evolves towards overwhelming the capability of human archivists and curators (either system administrators, programmers and designers), it is important to aid this task, by offering services that can check whether it is feasible to perform a task over a digital object. For example, a series of conversions and emulations could make feasible the execution of software written in 1986 software on a 2013 platform. The process of checking whether this is feasible or not could be too complex for a human and this is where advanced reasoning services, could contribute, because such services could greatly reduce the human effort required for periodically checking (monitoring) whether a task on a digital object is performable.

Towards this vision, this report describes how we have extended past rule-based approaches for dependency management for capturing *converters* and *emulators*, and we have demonstrated that the proposed modeling enables the desired reasoning regarding task performability, which in turn could greatly reduce the human effort required for periodically checking or monitoring whether a task on an archived digital object is performable.

We have provided various examples including examples that show how real converters and emulators can be modeled. We have designed and implemented a proof of concept prototype for testing whether the proposed reasoning approach behaves as expected. The results were successful, therefore the technical objectives of this task (as described in the DoW) are fully accomplished.

Although the Knowledge Base of the prototype system (which has been implemented using W3C semantic web technologies) currently represents only some indicative tasks, it can demonstrate the benefits of the proposed approach. In addition, we used this prototype system as a means to specify a number of concrete use cases for the case of DANS.
We should also mention that since the implementation is based on W3C standards, it can be straightforwardly enriched with information coming from other external sources (i.e. SPARQL endpoints). In any case we should stress that the methodology presented is general and can be used for extending the modeled tasks, modules, converters and emulators, in order to capture the desired requirements.

For cases where the considered modules have internal and known structure, e.g. as in the case of formally expressed community knowledge (vocabularies, taxonomies, ontologies and semantically described datasets), instead of considering each such module as an atom (undivided element), the internal structure can be exploited for computing more refined gaps. If furthermore, this internal structure is represented using Semantic Web Languages (RDF/S, OWL), which currently form the *lingua franca* for structured content, then one can apply general purpose (application independent) RDF *diff* tools (tools that compute the *difference* between two RDF/S Knowledge Bases), for computing more refined gaps. To this end, in this deliverable we have reported some recent contributions that we have made on such tools that concern the management of blank nodes.

Contribution to VCoE. Overall, the methodology for capturing, modeling, managing and exploiting the various interoperability dependencies can be considered as a significant contribution to the VCoE: expertise in designing and realizing novel inference services for task-performability, risk-detection and for computing intelligibility gaps. Furthermore, the implemented system (which is already web accessible) can be used for disseminating the results of this work, as well as for investigating and planning future operational applications of this approach, either in the context of single organizations (e.g. the DANS case), or in the context of the VCoE (e.g. as an advanced semantic registry).

15 REFERENCES

Ref id	Bibliographic Reference
Di Battista, Eades, Tamassia & Tollis, 1999	G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis (1999). *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall.
Becker & Rauber, 2011	C. Becker and A. Rauber (2011). Decision criteria in digital preservation: What to measure and how. *JASIST*,62(6):1009–1028.
Bellard, 2005	F. Bellard (2005). QEMU, a fast and portable dynamic translator. In *Procs of the USENIX Annual Technical Conference, FREENIX Track*, pages 41–46.
Bergmeyer, 2011	W. Bergmeyer (2011). The KEEP Emulation Framework. In*Proceedings of the 1st International Workshop on SemanticDigital Archives (SDA 2011)*.
Bertolazzi, et al., 1994	P. Bertolazzi, R. F. Cohen, G. Di Battista, R. Tamassia,and I. G. Tollis (1994). How to draw a series-parallel digraph.*International Journal of Computational Geometry &Applications*, 4(4):385–402.
Christiansen, Dahl, 2004	H. Christiansen and V. Dahl (2004). Assumptions and abductionin Prolog. In *3rd International Workshop on Multiparadigm Constraint Programming Languages, MultiCPL*, volume .Citeseer.
Christiansen, Dahl, 2005	H. Christiansen and V. Dahl (2005). HYPROLOG: A new logic programming language with assumptions and abduction. *Lecture Notes in Computer Science*, 3668:159–173.
Doerr, Tzitzikas, 2012	Doerr and Y. Tzitzikas (2012), "Information Carriers and Identification of Information Objects: An Ontological Approach", arXiv:1201.0385v1 [cs.DL] (http://arxiv.org/abs/1201.0385)
Giaretta, 2010	David Giaretta (Editor) (2010). *Advanced Digital Preservation*.Springer
Elenius, Martin, Ford & Denker, 2009	D. Elenius, D. Martin, R. Ford, and G. Denker (2009). Reasoning about Resources and Hierarchical Tasks Using OWL andSWRL. In *Procs of the 8th International Semantic WebConference (ISWC'2009)*
Erling, Mikhailov, 2007	O. Erling and I. Mikhailov (2007). RDF Support in the VirtuosoDBMS. In *Procs of 1st Conference on Social SemanticWeb*.
Fikes, Nilsson, 1972	R.E. Fikes and N.J. Nilsson (1972). Strips: A new approach to the application of theorem proving to problem solving.*Arti_cial intelligence*, 2(3-4):189–208.
Horrocks, et al., 2004	Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, SaidTabet, Benjamin Grosof, and Mike Dean (2004). "SWRL: ASemantic Web Rule Language Combining OWL and RuleML", May 2004.(http://www.w3.org/Submission/SWRL/).
Jackson, 2011	A.N. Jackson (2011). Using automated dependency analysis to generate representation information. In *Procs of the 8th International Conference on Preservation of Digital Objects (iPres'2011)*.
Lohman, Kiers, Michel & van der Hoeven, 2011	B. Lohman, B. Kiers, D. Michel, and van der J. Hoeven (2011). Emulation as a Business Solution: the EmulationFramework. In *Procs of the 8th International Conferenceon Preservation of Digital Objects (iPres'2011)*.
Mallea, Arenas, Hogan, Polleres, 2011	Alejandro Mallea, Marcelo Arenas, Aidan Hogan, and Axel Polleres (2011): On Blank Nodes .ISWC 2011
Marketakis, Tzanakis & Tzitzikas, 2009	Y. Marketakis, M. Tzanakis, and Y. Tzitzikas (2009). PreScan:Towards Automating the Preservation of Digital Objects.In*Procs of the International Conference on Management of Emergent Digital Ecosystems MEDES'2009*, Lyon,France, October.
Marketakis &	Y. Marketakis and Y. Tzitzikas (2009). Dependency Managementfor Digital Preservation

Tzitzikas, 2009	using Semantic Web technologies.*International Journal on Digital Libraries*, 10(4).
McGuinness & Harmelen, 2004	D. L. McGuinness and F. van Harmelen (2004). "OWL WebOntology Language Overview",(http://www.w3.org/TR/owl-features/).
Rechert, Suchodoletz & Welte, 2010	K. Rechert, D. von Suchodoletz, and R. Welte (2010). Emulationbased services in digital preservation. In *Procs of the 10th annual joint conference on Digital libraries*, pages 365–368.ACM.
Tarrant, Tarrant & Carr, 2009	D. Tarrant, S. Hitchcock, and L. Carr (2009). Where the SemanticWeb and Web 2.0 meet format risk management: P2registry. In *In Procs of the 6th Intern. Conf. on Preservation of Digital Objects (iPres 2009)*
Theodoridou, Tzitzikas, Doerr, & Marketa, 2010	M. Theodoridou, Y. Tzitzikas, M. Doerr, Y. Marketakis,and V. Melessanakis (2010). Modeling and Querying Provenanceby Extending CIDOC CRM. *J. Distributed and ParallelDatabases (Special Issue: Provenance in Scientic Databases).*
Tzitzikas, 2007	Y. Tzitzikas (2007). "Dependency Management for the Preservation of Digital Information". In *Procs of the 18th Intern. Conf. on Database and Expert Systems Applications, DEXA'2007*, Regensburg, Germany,September 2007.
Tzitzikas & Flouris, 2007	Y. Tzitzikas and G. Flouris (2007). "Mind the (Intelligibily) Gap".In*Procs of the 11th European Conference on Research andAdvanced Technology for Digital Libraries, ECDL'07*,Budapest, Hungary, September 2007. Springer-Verlag.
Tzitzikas, Marketakis & Antoniou, 2010	Y. Tzitzikas, Y. Marketakis, and G. Antoniou (2010). Task-basedDependency Management for the Preservation of DigitalObjects using Rules. In *Procs of 6th Hellenic Conf. on Artificial Intelligence, SETN-2010*, Athens, Greece
Van der Hoeven, Lohman & Verdegem, 2008	J. Van der Hoeven, B. Lohman, and R. Verdegem (2008). Emulation for digital preservation in practice: The results. *International Journal of Digital Curation*, 2(2).
Suchodoletz, et al., 2010	D. von Suchodoletz, K. Rechert, J. van der Hoeven, and J. Schroder (2010). Seven steps for reliable emulation strategies–solved problems and open issues. In *7th Intern.Conf. on Preservation of Digital Objects (iPRES2010)*,pages 19–24.
Tzitzikas, Analyti & Kampouraki, 2012	Y. Tzitzikas, A. Analyti and M. Kampouraki (2012) Curating the Specificity of Metadata while World Models Evolve, iPres2012 (http://www.ics.forth.gr/~tzitzik/publications/Tzitzikas_2012_iPres_Specificity.pdf)
Tzitzikas, Marketakis & Kargakis, 2012	Y. Tzitzikas, Y. Marketakis and Y. Kargakis (2012). Conversion and Emulation-aware Dependency Reasoning for Curation Services, iPres2012 (http://www.ics.forth.gr/~tzitzik/publications/Tzitzikas_2012_iPres_DepMgmtForCovertersEmulators.pdf)
Tzitzikas, Lantzaki & Zeginis, 2012	Y. Tzitzikas, C. Lantzaki and D. Zeginis (2012). Blank Node Matching and RDF/S Comparison Functions , Proceedings of the 11th International Semantic Web Conference (ISWC'12), Nov 2012, Boston, USA (http://www.ics.forth.gr/~tzitzik/publications/Tzitzikas_2012_ISWCpaper.pdf)
Ceri, Gottlob, & Tanca, 1989	S Ceri, G. Gottlob, and L. Tanca (1989). What you always wanted to know about datalog (and never dared to ask). In *IEEE Transactions on Knowledge and Data Engineering, I(1)*.
Zeginis, Tzitzikas & Christophides, 2011	D. Zeginis, Y. Tzitzikas, and V. Christophides (2011). "On Computing Deltas of RDF/S Knowledge Bases". ACM Transactions on the Web (TWEB).
Zeginis, Tzitzikas & Christophides,	D. Zeginis, Y. Tzitzikas, and V. Christophides (2007). "On the Foundations of Computing Deltas Between RDF Models". In Procs of ISWC-07.

2007	
Noy & Musen, 2002	N. F. Noy and M. A. Musen (2002). "PromptDiff: A Fixed-point Algorithm for Comparing Ontology Versions". In Procs of AAAI-02.
APARSEN D24.1	APARSEN Project: Deliverable 24.1. Report on authenticity and plan for interoperable authenticity evaluation system. (2012)
APARSEN ID24.1	ID2401, Report on Provenance Interoperability and Mappings, (Internal APARSEN Deliverable) (2012)
Lantzaki, Tzitzikas & Zeginis, 2012	Christina Lantzaki, Yannis Tzitzikas, Dimitris Zeginis (2012). Demonstrating Blank Node Matching and RDF/S Comparison Functions. International Semantic Web Conference (Posters & Demos)
Miller, 2000	Paul Miller (2000) .ARIADNE. Tratto da ARIADNE Web Magazine for Information Professionals: http://www.ariadne.ac.uk/issue24/interoperability
Lorie, 2001	R.A. Lorie (2001). Long term preservation of digital information. Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries, pages 346–352.
Van der Hoeven, van Diessen & van der Meer, 2005	JR van der Hoeven, RJ van Diessen, and K. van der Meer (2005). Development of a Universal Virtual Computer (UVC) for long-term preservation of digital objects. *Journal of Information Science*, 31(3):196.

16 APPENDIX

16.1 ARTICULATION WITH OTHER APARSEN WPS AND TASKS, CONTRIBUTIONS TO THE VCOE AND OOUTREACH

Here we describe in brief how this work is related with other APARSEN work packages (completed, ongoing or forthcoming) of APARSEN. They are synopsized in the form of a table.

Workpackage	***What***
WP24 (Authenticity and Provenance)	The various models for representing and recording provenance and authenticity, as described in the deliverables of WP24, allow answering some basic questions, e.g. who derived that, when, using what, etc. From the perspective of the current deliverable, these questions can be considered as *tasks*, and their performability can be modeled for better ensuring that their performability will be preserved.
WP22 (Identifiers and Citability)	The *resolvability* of persistent identifiers (as described in the deliverables of WP21) can be considered as a task (whose dependencies can be modeled).
WP21 (Preservation Services)	The reasoning services described in this deliverables is a kind of preservation services.
WP11 (Common vision).	The work described in this deliverable will be part of the common vision.
WP13 (Coordination of common Standards).	Since the implementation of the approach described in this deliverable is based on W3C standards (RDF/S), there is no need for a new standard. Only a possible agreement on how to exchange information about software modules and tasks. Specifically, a common core schema is enough. For instance, that would allow directly importing the data of the PRONOM to the prototype system.

16.2 CONTRIBUTION TO THE VCOE

The methodology for capturing, modeling, managing and exploiting the various interoperability dependencies can be considered as a significant contribution to the VCoE. Consequently, the VCoE could offer to the community its expertise in designing and realizing novel inference services for task-performability, risk-detection and for computing intelligibility gaps.

Furthermore, the implemented system (which is already web accessible) can be used for disseminating the results of this work, as well as for investigating and planning future operational applications of this approach, either in the context of single organizations (e.g. the DANS case), or in the context of the VCoE (e.g. as an advanced semantic registry).

16.3 USABILITY EVALUATION OF EPIMENIDES

Two scenarios were used in this evaluation:

Scenario A: Consider a user, who has a laptop where he has installed a C++ compiler (gcc) and a smart phone running Android OS. Suppose (s)he has an old source file in Pascal programming language, say game.pas, and (s)he has found a converter from Pascal to C++, say p2cpp, and an emulator of Windows OS over Android OS, say emulWin. It is evident that (s)he cannot run directly the game.pas on his/her laptop or on his/her smart phone.

Using the application load his/her profile (Just Run a Demo - Scenario User A), which already contains all the required modules (you do not need to add anything), upload the game.pas file, and try to understand why finally (s)he can run the game.pas. You can download the game.pas file from here.

Scenario B: Now consider another user who has a smart phone running Android OS. Suppose that he received a "secret.doc" file. Assume that he has found an Android to Windows Emulator (i.e. an emulator that allows running windows applications on an Android OS). This user is wondering if he can edit the secret.doc file. Use the application, that is described in this document, and try to answer this question. You should add the modules that this user has.

After that the users had to fill the following form:

Did you successfully complete the Scenario A *
- YES
- NO

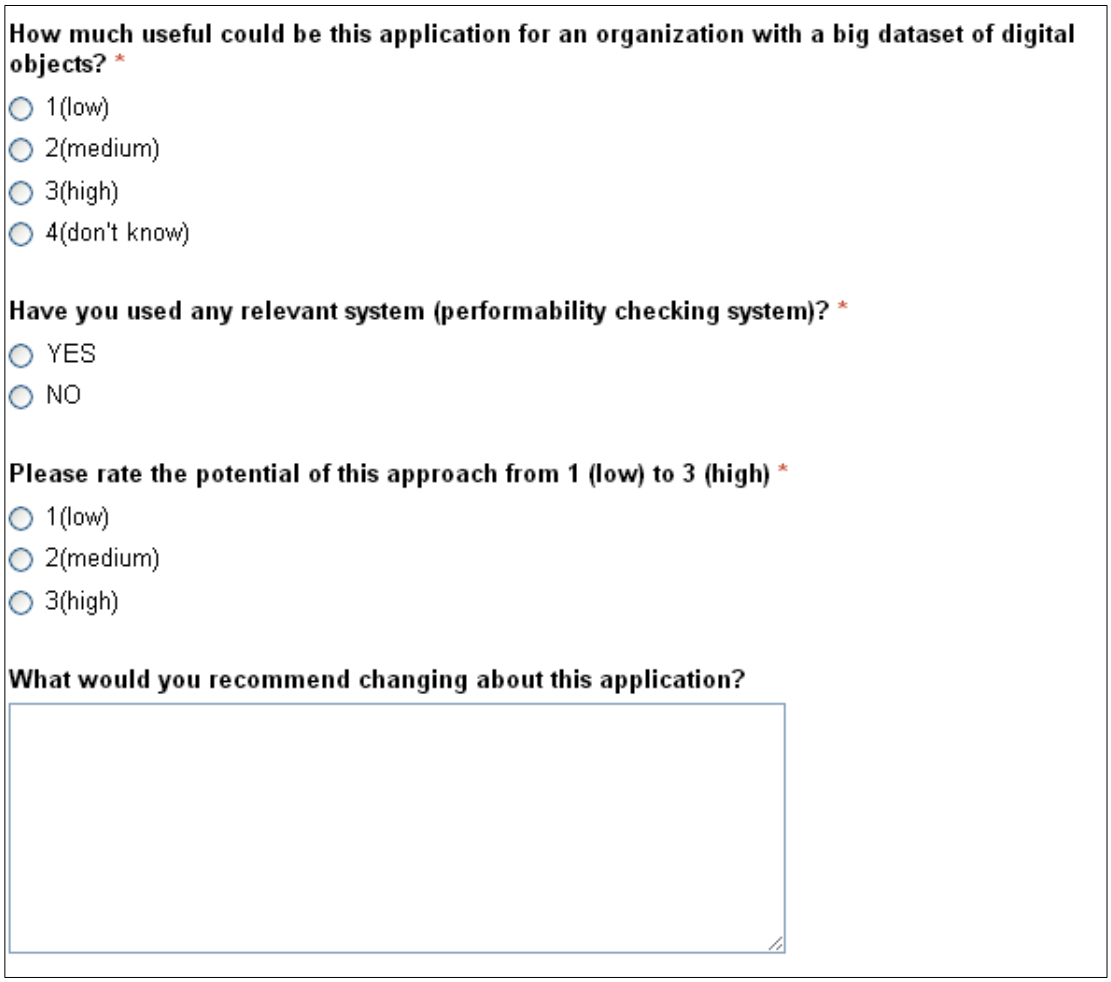
The time to complete Scenario A was :
- 1 - 3 min
- 4 - 6 min
- 7 - 9 min
- > 10 min

Did you successfully complete the Scenario B *
- YES
- NO

The time to complete Scenario B was :
- 1 - 3 min
- 4 - 6 min
- 7 - 9 min
- > 10 min

Have you a better understanding of why a task can be performed in existing and unknown profile *
(Scenario A)
- YES
- NO

Did the system assist you in checking the performability of a task? *
with the system it is easier and less time consuming to do it
- YES
- NO



How much useful could be this application for an organization with a big dataset of digital objects? *

- 1(low)
- 2(medium)
- 3(high)
- 4(don't know)

Have you used any relevant system (performability checking system)? *

- YES
- NO

Please rate the potential of this approach from 1 (low) to 3 (high) *

- 1(low)
- 2(medium)
- 3(high)

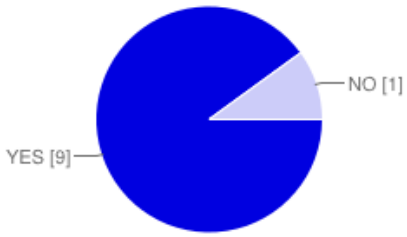
What would you recommend changing about this application?

16.3.1 Results

Ten users answered this questionnaire with ages ranging from 20 to 30. All of the participants had a computer science background (some of them had a MSc in Computer Science). We can distinguish these users in two groups: the advanced group consisting of 3 users (from APARSEN) and the regular ones consisting of 7 users (from Computer Science Department, University of Crete). The advanced users had read the deliverable before using system, while the regular ones had not. For this reason, and before starting the evaluation, we gave to each regular user a brief tutorial on using the system through examples .

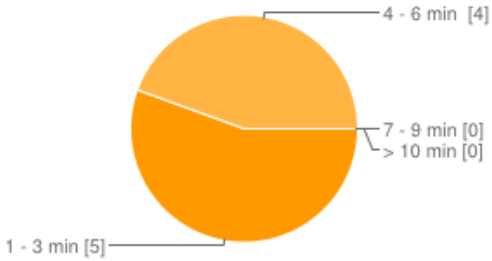
Below we summarize the results the answers of the questionnaire.

Did you successfully complete the Scenario A



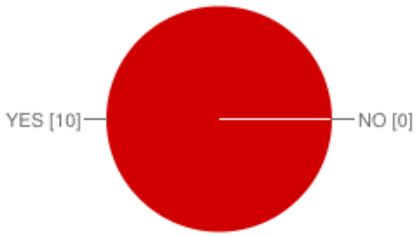
YES	**9**	90%
NO	**1**	10%

The time to complete Scenario A was :



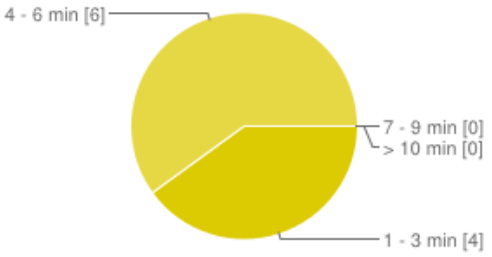
1 - 3 min	**5**	56%
4 - 6 min	**4**	44%
7 - 9 min	**0**	0%
> 10 min	**0**	0%

Did you successfully complete the Scenario B



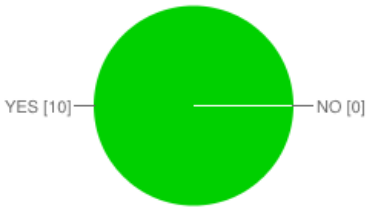
YES	**10**	100%
NO	**0**	0%

The time to complete Scenario B was :



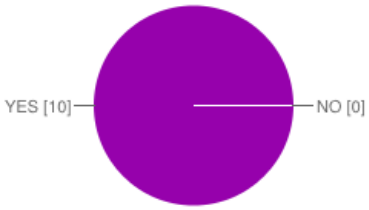
1 - 3 min	**4**	40%
4 - 6 min	**6**	60%
7 - 9 min	**0**	0%
> 10 min	**0**	0%

Have you a better understanding of why a task can be performed in existing and unknown profile



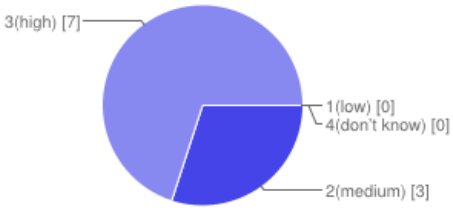
YES	**10**	100%
NO	**0**	0%

Did the system assist you in checking the performability of a task?



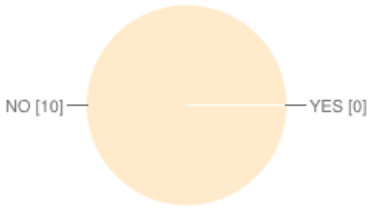
YES	**10**	100%
NO	**0**	0%

How much useful could be this application for an organization with a big dataset of digital objects?



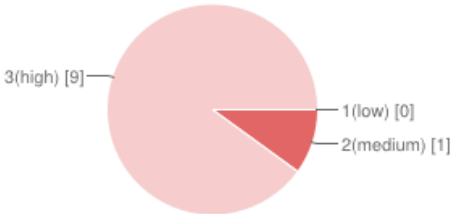
1(low)	**0**	0%
2(medium)	**3**	30%
3(high)	**7**	70%
4(don't know)	**0**	0%

Have you used any relevant system (performability checking system)?



YES	**0**	0%
NO	**10**	100%

Please rate the potential of this approach from 1 (low) to 3 (high)



1(low)	**0**	0%
2(medium)	**1**	10%
3(high)	**9**	90%

Analysis

As we can see, 90% of the participants completed the scenario A, while scenario B was completed from all users (100%). The time to complete both scenarios A and B was less than 6 minutes. From the above, we can conclude that Epimenides is understandable and easy to use.

In the questions 5 and 6, all users (100%) answered that the system assisted them in checking the performability of a task and that they better understood why a task can be performed in an existing and unknown profile. This demonstrates the value of the system.

Finally, 70% declared that this application is useful for an organization with a big dataset of digital objects. It is also worth noting that no user had ever used any relevant system. At last, a big percentage (90%) of the participants rated with 3 (high) the potential of this approach.