PreScan: Towards Automating the Preservation of Digital Objects

Yannis Marketakis
Computer Science Department, University of Crete
Institute Of Computer Science, FORTH-ICS
GREECE
marketak@ics.forth.gr

Makis Tzanakis
Computer Science Department, University of Crete
Institute Of Computer Science, FORTH-ICS
GREECE
tzanakis@ics.forth.gr

Yannis Tzitzikas
Computer Science Department, University of Crete
Institute Of Computer Science, FORTH-ICS
GREECE
tzitzik@ics.forth.gr

ABSTRACT

The preservation of digital objects is a topic of prominent importance for archives and digital libraries. However the creation and maintenance of metadata is a laborious task that does not always pay off immediately. For this reason there is a need for tools that automate as much as possible the creation and curation of preservation metadata. Such an automation is important especially for emergent systems and structures, like file systems, which are much more complex and dynamic, in comparison to the traditional digital archives. In this paper we propose a semi-automatic approach that binds in a flexible manner (a) automatically extracted embedded metadata, (b) manually provided metadata, and (c) dependency management services. In addition we elaborate on the problem of keeping the metadata repository up-to-date. Finally, we report our experiences from developing and using such a system based on Semantic Web technologies.

1. INTRODUCTION

The preservation of digital objects is a challenging task for various organizations such as libraries, cultural institutions, museums, archives, etc. Digital objects require constant maintenance because they depend on both software and hardware modules that are upgraded or replaced every few years. Several aspects of the problem are being studied including data/medium preservation strategies [10, 6, 17, 18], migration and encapsulation approaches [10, 22, 11, 8], work-flow and preservation approaches [19], theoretical attempts [5], cost-related strategies for data preservation planning [15, 27, 4, 20, 25], standards [13], and there are several preservation initiatives [7] and ongoing international projects [1, 2].

Our objective is to assist the curation of archives of digital objects. Specifically, we aim at providing services that help archivists to check whether the archived digital artifacts remain functional, to identify hazards and the consequences of probable losses or obsolescence risks. The majority of preservation approaches rely on metadata. However the creation and maintenance of metadata is a laborious task that does not pay off immediately, especially if these metadata are useful only for preservation purposes. For this reason there is a need for tools that automate as much as possible the creation and curation of preservation metadata. We would like to bypass the strict (often manual) ingestion process while at the same being compatible with it. According to the traditional approach, the ingestion phase starts with assigning identifiers to the objects and then extracting/creating metadata for these objects. These metadata can be expressed using various metadata schemas and formats (like DIDL, METS, etc) and usually the update/movement of a metadata record is prohibited. We want to relax these constraints and automate the process of metadata extraction and management. Automation is crucial for the preservation of emergent systems and structures, like file systems, which are much more complex and dynamic, in comparison to the traditional digital archives.

We propose an approach where some of the metadata are extracted automatically and periodic re-scans are used for keeping them up-to-date. The user is able to view the extracted metadata and add additional knowledge. What we propose is quite similar in spirit with the crawlers of Web Search Engines. In our case we scan the file system, we extract the embedded metadata and build an index. The difference in our case is that we need to support (a) more advanced extraction services, (b) manual addition of metadata, (c) more expressive representation frameworks for keeping and exploiting the metadata (i.e. metadata schemas expressed in Semantic Web languages), (d) rescans that do not start from scratch but exploit the previous status of the index, and (e) associations with external sources (e.g. registries). The latter is important since the *intelligibility* of a digital object depends on the availability of other digital objects. To this end we propose linking objects with external sources and providing dependency management services for identifying obsolescence risks.

The rest of this paper is organized as follows: Section 2 elaborates on the requirements. Section 3 reviews the related work. Section 4 describes the architecture and the functionality of the tool `PreScan` and reports experimental

MEDES 2009 October 27-30, 2009, Lyon, France

results. Finally section 5 concludes and identifies issues that are worth further work and research.

2. METADATA AND PRESERVATION REQUIREMENTS

Metadata can be described as *"structured, encoded data that describe characteristics of information-bearing entities to aid in the identification, discovery, assessment, and management of the described entities"* [1]. Table 1 shows some examples of what can be considered as metadata.

Table 1: Examples of Metadata

Type	Metadata
Book	title, author, date of publication, subject, ISBN, dimensions, number of pages, text language
Photographs	date and time of capture, details of the camera settings (focal length, aperture, exposure), coordinates (geotagging)*
Audio files	album name, song title, genre, year, composer, contributing artist, track number and album art
Relational Databases	Database catalog storing information about the names of tables and columns, the domains of values in columns, number of rows, etc
Software	For example, in Java, the class file format contains metadata used by the Java compiler and the Java virtual machine to dynamically link classes and to support reflection.
Web Pages	Meta-tags, general purpose descriptions expressed using Semantic Web languages.

*: Many digital cameras record metadata in exchangeable image file format (EXIF)

Metadata can be stored either *internally*, i.e. in the same file as the data, or *externally*, i.e. in a separate file. The former are usually called *embedded*, the latter *detached*. The detached metadata are usually stored in a special repository. Both approaches have advantages and disadvantages. One benefit of the embedded metadata is that they are transferred with the data and thus their access and manipulation is straightforward. However embedded metadata can create redundancies and this approach does not allow holding and managing all metadata together. On the other hand, if the metadata are detached, then this means that they are stored in a special repository. This approach has less redundancy, we can support efficient metadata search, and we can manipulate them efficiently, e.g. we can perform bulk metadata updates. However, the way metadata are linked to data should be treated with care as inconsistencies may arise. Overall, to manage a corpus of digital objects requires tackling several issues and problems. From our experience and analysis, we have identified the following basic requirements:

- **Automatic Scanning** of file systems
  We need systems that can operate like the crawlers of Web search engines, i.e. scan the desired parts of the file system (or network) according to a given configuration (regarding desired folders, extensions of files, etc).

- **Automatic Format Identification** and **Extraction of Embedded Metadata**
  It is useful to extract the embedded metadata so that to make them visible to the curators. Since several formats contain embedded metadata we need to extract them easily. Various format identification tools and metadata extractors have been developed and can be used for this purpose (e.g. JHOVE[2], meta-extractor[3], Droid[4]).

- Support for **Human-entered/edited Metadata**
  Users (or curators) should be able to add extra metadata to an already scanned file (apart from the automatically extracted). For example one might want to add extra metadata about provenance, digital rights, or the context of the objects.

- **Periodic Re-Scannings** without loosing the human-provided metadata
  As the contents of the files change frequently we need to update their metadata in a flexible manner. To this purpose, periodic re-scannings are useful for ensuring the freshness of the metadata. However the human-provided metadata should be preserved.

- **Referential Integrity** services
  If a file/folder is moved to another location we would like to identify such changes in order to reflect them to its detached metadata. This is important also for ensuring that the human-entered metadata will be preserved.

- **Dependency Management** and **Intelligibility-preservation** services
  A digital object might depend directly or indirectly on other modules. A module can be either a hardware/software component, a file format etc. Therefore we need to record such dependencies in order to facilitate tasks like: (a) the identification of the objects that are in danger in case a module (e.g. a software component or a format) is becoming obsolete (or has been vanished), (b) the decision of what metadata need to be captured and stored for a designated community, and (c) the reduction of the metadata that have to be archived, or delivered (as a response to queries) to the users of that community.

- Exploitation of **Existing Registries**
  External registries (like Pronom [26], GDFR [21], Registry of CASPAR) that contain information about file formats and versions of software for each one of them, should be exploited.

- **Usability and Control of the Whole Process**
  The functionalities must be performed in a simple and easy to use manner with the support of graphical user interfaces.

3. RELATED WORK

There are only few related works. For instance, the work presented in [11, 12] scans the filesystem, extracts the type of the digital files and detects those files having obsolete types. Its functionality relies on three registries: a software version registry, a format registry, and a recommended format registry.

[1] American Library Association, Task Force on Metadata Summary Report, June 1999

[2] http://hul.harvard.edu/jhove

[3] http://meta-extractor.sourceforge.net

[4] http://droid.sourceforge.net/wiki/index.php

Another approach based on migration is described in [9]. It adopts a SOA (Service-Oriented Architecture) for enabling the combination of different format detectors and registries to support automatic migration. The key difference with our work is that we extract the embedded metadata, we link them with data that enable dependency management services and we focus on supporting the entire life-cycle of metadata (i.e. the automatic identification of file removals and movements).

[23, 24] presents an approach for web page preservation. It aims at enabling the web server as a just-in-time metadata-extractor/generator. The archivist (or client) receives the metadata of a web page at dissemination time (as an XML-formatted response), and of course this adds an extra overhead for the web-server. That work pre-supposes that all digital files are placed in a web server and that the extended web-server has been installed.

Empirical Walker [3] is probably the most similar tool with `PreScan`. It also scans the file system, it determines file formats, it analyzes file contents calculates checksums, and associates external metadata. It also adopts JHOVE as metadata extractor. Regarding file format identification, Empirical Walker first assigns a MIME-type to every file according to a mapping table (that maps file extensions to MIME-types) and in a later phase it uses JHOVE to validate the associated MIME-types and extract more technical metadata (only for those files that are supported from JHOVE). However associating MIME-types based only on the file-extensions is error-prone because file extensions are often unreliable or missing. Furthermore, and in comparison to our work, we allow the addition of user-provided metadata and we can output the resulting metadata in various ways and formats also exploiting the expressive power of Semantic Web languages. Specifically the resulting metadata can be accessed or edited: (a) through the developed GUI, (b) in txt format (in a human readable format), (c) in XML which is directly output from JHOVE and (d) through a SW repository.

Metadata Miner Catalogue[5] is a commercial software tool that lists files and folders summary information, extracts file properties and their metadata and create reports in various formats (including XML, HTML and RDF according to Dublin Core schema). Furthermore it identifies several file formats but does not recognize the specific version of that format. For example it will recognize a .doc file as a Microsoft Word document but cannot identify whether it is a Microsoft Word 6 document or a Microsoft Word 2003 document.

Table 2 compares our work (`PreScan`) with other tools like Empirical Walker **(EW)**, DROID, Metadata Extractor **(ME)** and Metadata Miner Catalogue **(MMC)**. The symbol $n1$ indicates file format identification but without version information, while $n2$ indicates that some structural dependencies are extracted (however no further details are available).

Notice that `PreScan` is the only tool that allows file system re-scans that protect the human-provided metadata, and identifies file movements.

4. THE FUNCTIONALITY OF `PreScan`

PreservationScanner, for short `PreScan`, is a system that

[5] http://peccatte.karefil.com/software/Catalogue/MetadataMiner.htm

Table 2: Comparison with related systems

	`PreScan`	EW	Droid	ME	MMC
ReScan with preservation of manual metadata	✓				
Identification of file movements	✓				
Mapping Confirmation by the user	✓				
Export to RDF	✓				✓
Exploitation of format registries	✓ (Pronom + Dependencies)		✓ (Pronom)		
Format identification	✓	✓	✓	✓^{n1}	✓^{n1}
Compliance with dependency management	✓	✓^{n2}			

we have developed based on the previously described requirements. `PreScan` consists of four main components: the **scanner**, a component responsible for scanning file systems, the **metadata extractor**, a component for extracting the embedded metadata of the scanned files, the **repository manager**, a component for storing and managing these metadata, and the **controller**, a component that controls the entire process and metadata life-cycle. The overall architecture of the system is shown in Figure 1. We have adopted a modular design with well-defined interfaces in order to be able to extend or replace a component easily. For instance, we can easily switch to another metadata extractor or use different repository storage approaches (as we discuss later on).

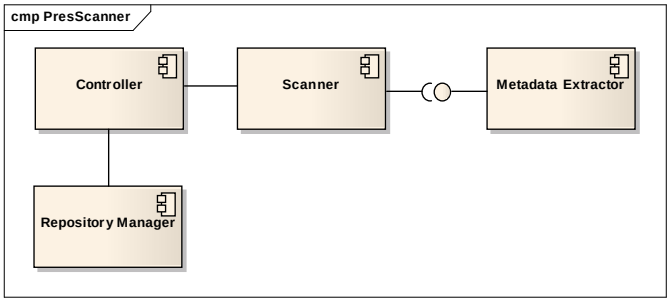

Figure 1: The Component diagram of `PreScan`

4.1 Controller

The tool starts like an AntiVirus program, i.e. it starts scanning all files and subfolders that originate from a certain folder (that is specified by the user). At the first scan a metadata record is created for each encountered file and stored through the Repository Manager. After the end of the scan the user can browse the repository and add extra metadata or edit the extracted ones. The difficult task is to keep the repository consistent while the file system changes. Recall that files are deleted, renamed or change positions and new files are created. `PreScan` uses the full pathname of a file as its identity. Let us consider the case where a

file is renamed. For sure we would not like to delete the old metadata record of that file and create a new one since we would loose all metadata that could have been entered manually by the user. If at each scanning we keep a copy image of the entire filesystem then at each subsequent scan we can compare the contents of each encountered file with the contents of the copied files of the previous scan in order to identify file name/position changes. However to keep a copy of the entire file system would be space consuming and the comparison would be unacceptably slow. To overcome this problem we compute and store the *md5 checksum* of the contents of every scanned file inside its metadata record, which is typically expressed as a fixed size hexadecimal number (32 digits). If a file does not change over time, then its md5 will remain intact. Obviously the md5 of a file should be updated whenever a file changes. We deal with file movements in a similar way.

The re-scanning process consists of two phases: the *scanning phase* and the *integration phase*. At the first phase the algorithm scans the filesystem. At the second phase (integration phase) the system tries to identify file additions, modifications and deletions based on the contents signatures (in our case md5), and asks from the user to verify the identified events.

Figure 2 sketches the algorithm of rescan. At first we retrieve the list of scanned files and for every file we encounter we compare it with the list of the previously scanned files to decide weather this file existed during the previous scan or it is a new file. If a file existed in the previous scan and has been changed since the last scan, we extract its new metadata and update the repository appropriately. *PendingList* is used for keeping the files that were not present at the previous scan (obviously these files are not associated with any metadata record). Let sf be such a file. This means that either: (a) sf is a new file, or (b) sf was moved from another folder, or (c) sf is an old file that has been renamed and `PreScan` cannot recognize it. In contrary, *ObsoleteList* keeps metadata records that no longer correspond to a file which means that either the specified file has been removed or the file has been moved to another location. At a final step we recognize file movements and removals by comparing the content of the files in the *PendingList* and *ObsoleteList* lists.

We have developed a GUI to aid users in establishing mappings between the elements of the *pending* and the *obsolete* list. Figure 4 shows an indicative screendump. The upper part of the window contains the new files that were found during the re-scanning and the bottom part all possible mappings from files that were moved/renamed. In this particular case we have 4 new files, 1 file movement and 1 file that was renamed.

4.2 Metadata Extrator

For every file `PreScan` extracts and keeps its filetype, path, owner, last modification date and size. Moreover we extract and keep fileformat-specific embedded metadata. `PreScan` uses JHOVE as a format detector and metadata extractor. It recognizes several formats (AIIF, ASCII, BYTESTREAM, GIF, HTML, JPEG, JPEG 2000, PDF, TIFF, UTF-8, WAVE, XML) and Table 3 shows some of the extracted metadata while Figure 3 shows a part of the JHOVE XML output schema. The element *repInfo* contains a subelement *property* that contains name-value pairs with technical information about a file (i.e. for images it will contain ColorSpace information, ExposureMode information, resolution etc).

Algorithm **ReScan**()
Input: Rep, Scanner, Extractor
Output: updated repository

1. PendingList = ObsoleteList = $\emptyset$
2. PreviousFiles = Rep.getAllFiles($ScannerConf$)
3. CurFiles = Scanner.getScannedFiles($ScannerConf$)
4. for each file f in CurFiles
5. if $f \in$ PreviousFiles //f *existed in the previous scan*
6. if f.lastModified > Rep.getLastModified(f.path)
//f *has changed*
7. m=Extractor.extractMetadata(f.path)
8. Rep.update(f.path,m)
9. else //f *is a "new" file*
10. PendingList = PendingList $\cup\{f\}$
11. end for
12. $ObsoleteList$ = PreviousFiles \ CurrFiles
13. for each file f in $PendingList$
14. m=Extractor.extractMetadata(f.path)
15. of = ObsoleteList.getMatched(f.content)
// *e.g. through MD5*
16. if (of==null) then // *nothing matched*
17. Rep.add(f,m)
18. $PendingList = PendingList \setminus \{f\}$
19. else
20. If UserVerifiesMapping(f, of) then
21. Rep.updateMetadata(of, f,m)
22. $PendingList = PendingList \setminus \{f\}$
23. $ObsoleteList = ObsoleteList \setminus \{of\}$
24. end for
25. Rep.DeleteOldRecords($ObsoleteList$)

Figure 2: The algorithm of `PreScan`

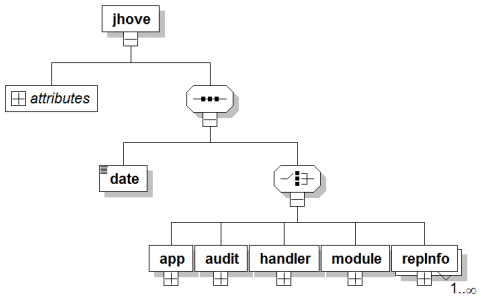

Figure 3: A fragment of the JHOVE output XML schema

4.3 Repository Manager

The Repository Manager is responsible for storing and updating the metadata records. The metadata record of a file includes both the extracted and the human-provided metadata. There are more than one choices regarding where these metadata are stored. The options that are currently supported are listed below (they are not mutually exclusive):

- (SF) For each scanned file its metadata record is created and stored in a Specific Folder specified by the

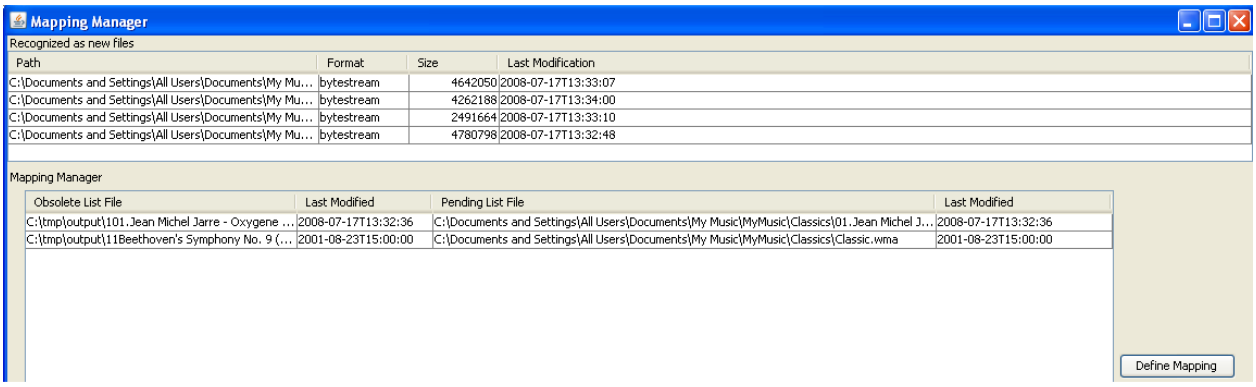

Figure 4: The GUI for managing mappings

Table 3: Recognized formats and extracted metadata

Format	Extracted Metadata
ALL Formats	ScanDate, FilePath, LastModificationDate, Size, Format, Status, Module, MimeType
AIFF	AudioDataEncoding, ByteOrder, FirstSampleOffset, Hours, Minutes, Seconds, Sample, NumberOfSamples
ASCII	LineEndings
GIF	GraphicRenderingBlocks, LogicalScreenWidth, LogicalScreenHeight, ColorResolution, BackGroundColorIndex, PixelAspectRatio, CompressionScheme, TransparencyFlag, ImageWidth, ImageLength, BitsPerSample
HTML	PrimaryLanguage, OtherLanguages, Title, MetaTags , Frames , Links, Scripts, Images , Citations, DefinedTerms, Abbreviations , Entities, UnicodeEntityBlocks
JPEG	CompressionType, ScanerManufacturer, ScannerModuleName, ImageWidth, ImageLength, BitsPerSample, SamplesPerPixel, PixelAspectRatioX, PixelAspectRatioY, Precision, ColorSpace, PixelXdimension, PixelYdimension, DateTimeOriginal, DateTimeDigitized, ExposureTime, LigthSource, Flash, FileSource, SceneType, Saturation, Sharpness
JPEG 2000	Brand, MinorVersion, Compatibility, precedence, XSize, YSize, BitsPerSample, SamplesPerPoxel, Creator
PDF	PageLayout, PageMode, Creator, Producer, CreationDate, Fonts
TIFF	ByteOrder, CompresiionScheme, SamplingFrequency, XSamplingFrequency, YSamplingFrequency, ImageWidth, ImageLength, BitsPerSample, SamplesPerPixel
UTF-8	LineEndings, Additional Control Characters, NumberOfCharacters, UnicodeCodeBlocks
WAVE	SchemaVersion, AudioDataEncoding, FrameCount, TimeBase, videoField, CountingMode, Hours, Minutes, Seconds, Frames, SampleRate, NumberOfSamples, NumChannels
XML	Version, Encoding, StandAlone, DTD, Schemas,Root, NameSpaces, Notations, CharacterReferences, Entities, ProcessingInstructions, Comments

user.

- (OF) For each scanned file its metadata record is created and stored in the same folder with the Original scanned file.
- (KB) The contents of the metadata records of scanned files are stored in a Semantic Web-based Knowledge Base. In that case the Repository Manager also offers querying services.

The extracted metadata of a digital file that are stored using the repository manager and can be later viewed or enriched by the user. For example assume that `PreScan` scans a file named `forth20090115.jpg`. It extracts various metadata about that file (e.g. image resolution, date, information about the digital camera captured this photo etc.) and stores them in the repository. Later on the user can enrich these metadata by adding human provided metadata such as a description for that photograph (e.g. "FORTH cake event in 15-01-2009"). Now suppose that this file is moved to another location. At the next scan, `PreScan` will identify this change and will update its metadata record with the new file location. Therefore the (user-provided) metadata for that file will not get lost. In case the KB option has been adopted, the user can also search for a file by querying the repository according to the extracted information. For example we could search for a photograph with description about "FORTH cake event" or photographs taken by a specific digital camera, etc.

4.3.1 Architecture of Ontologies

Recall that JHOVE outputs the extracted metadata in XML format. These files can be stored in a XML database and then queried using XQuery. Instead of having an XML-based framework an alternative approach is to define an ontology that allows expressing all the extracted metadata and can exploit the inheritance semantics of RDFS.

Regarding the adopted ontology a good choice is to adopt CIDOC CRM and extensions of that ontology. CIDOC Conceptual Reference Model (ISO 21127) is a core ontology describing the underlying semantics of database schemata and structures from all museum disciplines, archives and libraries. One recent extension of this ontology is CIDOC CRM Digital[6] which is appropriate for digital objects since

[6]http://cidoc.ics.forth.gr/rdfs/caspar/cidoc_digital2.3.rdfs

it contains extensions for capturing the properties and the provenance (creator, derivation chain) of digital objects. For example Figure 5 shows how two images (their metadata) are modeled. The PNG image was derived by converting a JPG image. The figure also shows information about the converter and the conversion. Such metadata may be added by users or by a workflow management system.

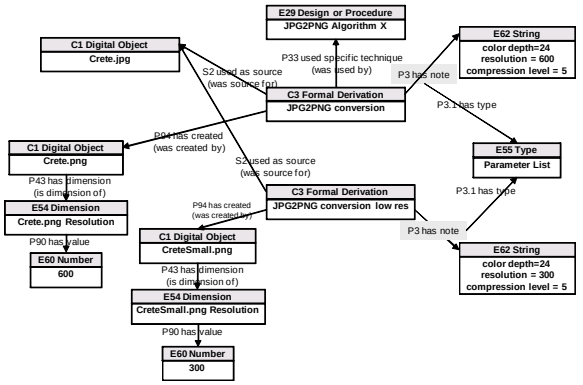

Figure 5: JPG2PNG Converter using CIDOC CRM Digital Ontology

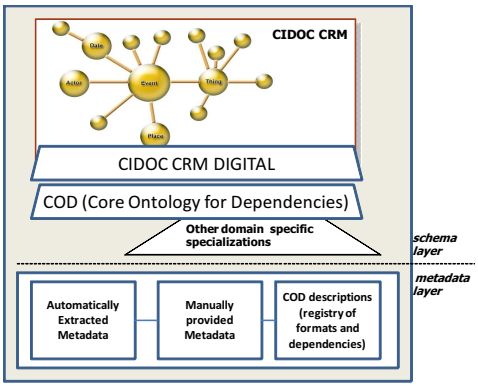

Figure 6: Architecture of SW Ontologies and Data

Figure 6 shows the general architecture of ontologies that we propose. On top we have CIDOC CRM, then we have CIDOC CRM Digital. Under that we have COD (Core Ontology for Dependencies)[7], which is an ontology for expressing dependencies. In brief, COD allows expressing (typed) modules as well as (typed) dependencies between these modules. In addition, it allows expressing designated community profiles, for short *profiles*, indicating those modules that are assumed to be known by a community. Figure 7 shows a part of that ontology (for more see [29, 30]). Finally we have domain-specific extensions.

The instance layer contains the automatically extracted metadata, the manually provided metadata and COD-based descriptions. The later capture information about file formats and software modules that were loaded by exporting the contents of the Pronom registry and correspond to the repository of `GapMgr`[8] In this way the information about the scanned files are linked with the descriptions of these formats. Specifically each scanned file/module is assigned a unique module identifier which in our case is the path of the file. Some of the dependencies of these modules are automatically extracted based on their type (i.e. *PreScan.jpeg* `dependsOn` *JPEG Module*), but the user can easily add more dependencies or edit the existing ones by using the `GapMgr` API or the GWT application.

`GapMgr` is a tool that is based on the same repository and relies on COD. In brief, `GapMgr` offers: (a) a programmatic interface written in Java offering methods for dependency management (`GapMgr` API), (b) two different implementation of the API: the first is a main memory implementation where the persistence layer is a plain file-system based, the second is an implementation over the SWKM (Semantic Web Knowledge Middleware)[9], and (c) an end-user Web-based application developed over GWT (Google Web Toolkit)[10] which can work with both implementations of the API. Figures 8 and 9 shows some indicative screendumps. The first one for defining (adding, changing or deleting) dependencies, the last one for computing intelligibility-aware packages [29].

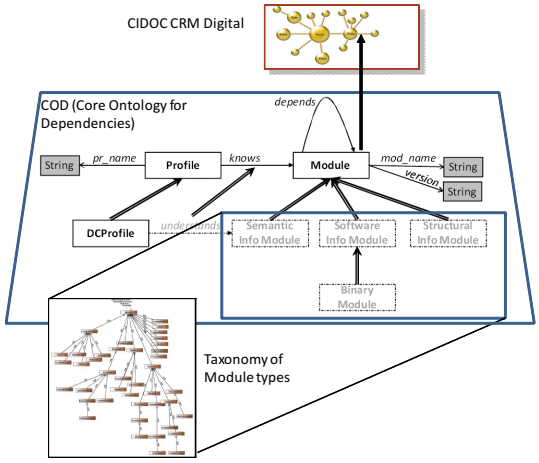

Figure 7: COD Ontology

Regarding the physical layer, the metadata according to the previous architecture can be stored as plain files in RDF/XML format (or Trig) for the options SF and OF. For the KB option, we can keep them in in a SW-based repository (like SWKM). The benefits of adopting a SW-based repository is that we can benefit from its validation mechanism (to keep the descriptions consistent), and its declarative query and update languages [14, 16]. In addition, the user can browse the SWKM repository through StarLion [28][11].

Regarding the transformation from XML to RDF `PreScan` parses the output of JHOVE and produces instances of an ontology that extends COD ontology expressed in RDF. In future we plan to investigate methods for making more easy

[7] http://cidoc.ics.forth.gr/rdfs/caspar/module_schema.rdfs

[8] `GapMgr` is one of the key components of the CASPAR project.

[9] http://athena.ics.forth.gr:9090/SWKM/

[10] http://code.google.com/webtoolkit

[11] http://www.ics.forth.gr/~tzitzik/starlion/

Table 4: Time Performance of the Scanning process

DataSets							Time (sec)			
Files	html/ xml	wave/ aiff	ascii/ utf8	jpeg/ gif	pdf	unknown	Extract	MD5	Store	Overall
10	2	2	1	4	1	1	3.2	2.5	0.8	9.2
10^2	19	11	7	36	8	19	26.8	46.4	1.8	77.0
10^3	201	115	56	251	103	274	632.6 (∼11 min)	487.2 (∼9 min)	15.7	1139.2 (∼19 min)
10^4	2123	1096	922	4629	594	636	5654.0 (∼95 min)	2636.1 (∼44 min)	365.0 (∼7 min)	8667.4 (∼145 min)
10^5	34069	1207	25467	13478	1240	24539	30066.1 (∼8.5 hr)	2849.3 (∼48 min)	2277.5 (∼38 min)	35230.8 (∼10 hr)

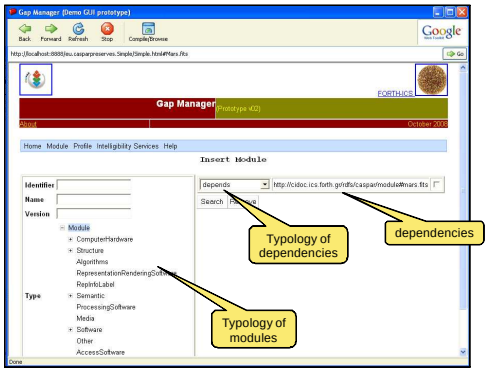

Figure 8: GapMgr GUI for managing dependencies

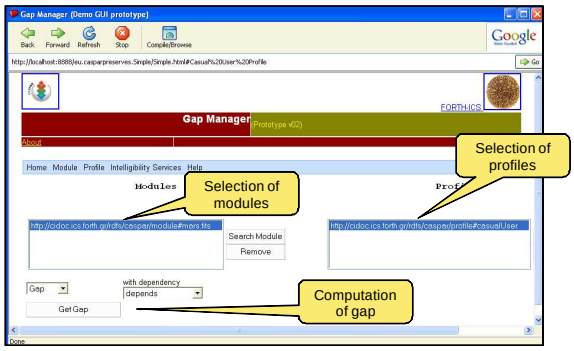

Figure 9: GapMgr GUI for computating the intelligibility gap

the mapping of the JHOVE-extracted metadata to instantiations of CIDOC CRM Digital.

4.4 Evaluation of PreScan

Regarding efficiency, the extraction of the embedded metadata depends on the type and the size of the file. In general, the bigger the size of a file is, the more time it takes to be scanned. Unknown filetypes (i.e. filetypes that are not recognized by JHOVE) are identified as BYTESTREAM modules and only the basic metadata (1^{st} row in Table 3) are extracted.

We performed some experiments over various datasets. Table 4 summarizes the results of the evaluation. For every dataset we report the total time for the scanning process as well as the time for every subtask. For every dataset we also report the number of the different files comparing to the total files of the set. As we can see the more time-consuming tasks is the metadata extraction and the computation of MD5 checksum. Clearly, these two tasks are independent from the repository mode (Section 4.3). However, the Overall time depends on the repository mode adopted. Here we have used the SF option and stored the metadata records as XML documents. We have noticed that the entire process takes about 10 hours for 100 thousand files.

5. CONCLUDING REMARKS

This paper described the design and implementation of a digital preservation system that automatically extracts the embedded metadata of digital objects, binds them with manually provided, and supports processes for ensuring the freshness of the metadata repository without loosing the human provided metadata. Regarding the metadata repository it supports a Semantic Web approach based on an architecture of ontologies appropriate for interoperability and for enabling dependency management services that assist the preservation of the intelligibility of the digital objects. The Alpha release of PreScan is available to download and use [12].

What we have proposed is quite similar in spirit with the crawlers of Web Search Engines (recall the differences mentioned at the introductory section), an approach that is feasible and appropriate for dynamic digital ecosystems.

In future we foresee the creation of specialized preservation scanners for various kinds of digital objects (e.g. for documents, software, videos, etc), and the provision of extractors from the publishers of formats to allow plugging them automatically to systems like PreScan.

Acknowledgements Motivation for this work is the ongoing EU project CASPAR (FP6-2005-IST-033572).

[12] http://www.ics.forth.gr/prescan/

6. REFERENCES

[1] "CASPAR (Cultural, Artistic and Scientific knowledge for Preservation, Access and Retrieval), FP6- 2005-IST-033572 (http://www.casparpreserves.eu/)".

[2] "PLANETS - Digital Preservation Research and Technology, HPRN-CT-2002-00308 (http://www.planets-project.eu)".

[3] R. Anderson, H. Frost, N. Hoebelheinrich, and K. Johnson. The AIHT at Stanford University: Automated preservation assessment of heterogeneous digital collections. *D-Lib Magazine*, 11:12, 2005.

[4] Carl Carl and Andreas Rauber. "Preserving digital media: Towards a preservation solution evaluation metric". In *Procs of the ICADL'2004*, pages 203–212, Shanghai, China, 2004.

[5] James Cheney, Carl Lagoze, and Peter Botticelli. "Towards a Theory of Information Preservation". In *Procs of the 5th European Conference on Research and Advanced Technology for Digital Libraries, ECDL '01:*, pages 340–351, London, UK, 2001. Springer-Verlag.

[6] B.F. Cooper and H. Garcia-Molina. InfoMonitor: unobtrusively archiving a World Wide Web server. *International Journal on Digital Libraries*, 5(2):106–119, 2005.

[7] M. Day. Integrating Metadata Schema Registries with Digital Preservation Systems to Support Interoperability: a Proposal. *DC 2003: Supporting Communities of Discourse and Practice-Metadata Research & Applications, Seattle, Washington (USA), September 28-October*, 2, 2003.

[8] M. Ferreira, A.A. Baptista, and J.C. Ramalho. An intelligent decision support system for digital preservation. *International Journal on Digital Libraries*, 6(4):295–304, 2007.

[9] M. Ferreira, A.A. Baptista, and J.C. Ramalho. A Foundation for Automatic Digital Preservation. 2008.

[10] M. Hedstrom. Digital Preservation: A Time Bomb for Digital Libraries. *Computers and the Humanities*, 31(3):189–202, 1997.

[11] J. Hunter and S. Choudhury. PANIC: an integrated approach to the preservation of composite digital objects using Semantic Web services. *International Journal on Digital Libraries*, 6(2):174–183, 2006.

[12] Jane Hunter and Sharmin Choudhury. A semi-automated digital preservation system based on semantic web services. In *JCDL '04: Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*, pages 269–278, New York, NY, USA, 2004. ACM Press.

[13] International Organization For Standardization. "OAIS: Open Archival Information System – Reference Model", 2003. Ref. No ISO 14721:2003.

[14] G. Karvounarakis, V. Christophides, and D. Plexousakis. Querying Semistructured (Meta)data and Schemas on the Web: The case of RDF & RDFS. Technical Report 269, ICS-FORTH, 2000. Available at: http://www.ics.forth.gr/proj/isst/RDF/rdfquerying.pdf.

[15] K.H. Lee, O. Slattery, R. Lu, X. Tang, and V. McCrary. The State of the Art and Practice in Digital Preservation. *Journal of Research-National Institute of Standards and Technology*, 107(1):93–106, 2002.

[16] M. Magiridou, S. Sahtouris, V. Christophides, and M. Koubarakis. "RUL: A Declarative Update Language for RDF". In *Procs. 4th Intern. Conf. on the Semantic Web (ISWC-2005)*, Galway, Ireland, November 2005.

[17] P. Maniatis, M. Roussopoulos, TJ Giuli, D.S.H. Rosenthal, and M. Baker. The LOCKSS peer-to-peer digital preservation system. *ACM Transactions on Computer Systems (TOCS)*, 23(1):2–50, 2005.

[18] M.L. Nelson, F. McCown, J.A. Smith, and M. Klein. Using the web infrastructure to preserve web pages. *International Journal on Digital Libraries*, 6(4):327–349, 2007.

[19] Arcot Rajasekar, Reagan Moore, Fran Berman, and Brian Schottlaender. Digital preservation lifecycle management for multi-media collections. In *ICADL*, pages 380–384, 2005.

[20] Carl Rauch, Pavuza Franz, Stephan Strodl, and Andreas Rauber. "Evaluating Preservation Strategies for Audio and Video Files". In *Procs of the DELOS Digital Repositories Workshop*, Heraklin, Crete, Greece, May 2005.

[21] GDFR (Global Digital Format Registry). (http://www.gdfr.info).

[22] S. Ross and M. Hedstrom. Preservation research and sustainable digital libraries. *International Journal on Digital Libraries*, 5(4):317–324, 2005.

[23] J.A. Smith and M.L. Nelson. A quantitative evaluation of dissemination-time preservation metadata. In *Proceedings of the 12th European conference on Research and Advanced Technology for Digital Libraries*, pages 346–357. Springer, 2008.

[24] J.A. Smith and M.L. Nelson. Creating preservation-ready web resources. *D-Lib Magazine*, 14(1/2):1082–9873, 2008.

[25] S. Strodl, C. Becker, R. Neumayer, and A. Rauber. How to choose a digital preservation strategy: evaluating a preservation planning procedure. In *Proceedings of the 2007 conference on Digital libraries*, pages 29–38. ACM Press New York, NY, USA, 2007.

[26] The technical registry PRONOM (The National Archives). (http://www.nationalarchives.gov.uk/pronom).

[27] K. Thibodeau. Overview of Technological Approaches to Digital Preservation and Chall enges in Coming Years. *The State of Digital Preservation: An International Perspective*, 2002.

[28] Y. Tzitzikas, D. Kotzinos, and Y. Theoharis. On Ranking RDF Schema Elements (and its Application in Visualization). *Journal of Universal Computer Science*, 13(12):1854–1880, 2007.

[29] Yannis Tzitzikas. "Dependency Management for the Preservation of Digital Information". In *Procs of the 18th International Conference on Database and Expert Systems Applications, DEXA'2007*, Regensburg, Germany, September 2007. Springer-Verlag.

[30] Yannis Tzitzikas and Giorgos Flouris. "Mind the (Intelligibily) Gap". In *Procs of the 11th European Conference on Research and Advanced Technology for Digital Libraries, ECDL'07*, Budapest, Hungary, September 2007. Springer-Verlag.