MOPSEUS – A DIGITAL REPOSITORY SYSTEM WITH SEMANTICALLY ENHANCED PRESERVATION SERVICES

Dimitris Gavrilis
Digital Curation Unit, Institute for the Management of Information Systems, Athena Research Centre, Athens, Greece
d.gavrilis@dcu.gr

Stavros Angelis
Digital Curation Unit, Institute for the Management of Information Systems, Athena Research Centre, Athens, Greece
s.angelis@dcu.gr

Christos Papatheodorou
Department of Archives and Library Sciences, Ionian University, Corfu, Greece and Digital Curation Unit, Institute for the Management of Information Systems, Athena Research Centre, Athens, Greece
papatheodor@ionio.gr

ABSTRACT

Repository platforms offer significant tools aiding institutions to preserve the wealth of their information resources. This paper presents the data model as well as the architectural features of Mopseus, a digital library service, built on top of Fedora-commons middleware, designed to facilitate institutions to develop and preserve their own repositories. The main advantage of Mopseus is that it minimizes the customization and programming effort that Fedora-commons involves. Moreover it provides an added value service which semantically annotates the internal structure of a Digital Object. The paper focuses on the preservation functionalities of Mopseus and presents a mechanism for automated generation of PREMIS metadata for each Digital Object of the repository. This mechanism is activated whenever an object is modified and is based on a mapping of the Mopseus data model to the PREMIS data model that ensures the validity of the transformation of the information stored in a Mopseus repository to semantically equivalent PREMIS metadata.

1. INTRODUCTION

Nowadays there exist several platforms that support the development of digital repositories, but a few of them focus on preservation and facilitate the repository administrators to implement preservation plans. On the other hand the existing preservation platforms, such as CASPAR [8] and Planets [9], provide infrastructures to meet the requirements for preservation actions of large memory organizations such as national libraries and archives. A crucial issue is how much effort users are required to put in order to develop digital repositories on top of such platforms, especially when these users are small institutions with tight, small budgets [15]. Existing repository platforms, such as eSciDoc (http://www.escidoc.org/), offer a number of powerful services, while some, such as Blacklight (http://www.projectblacklight.org/), offer an easy interface and some other, such as RODA [13] (http://roda.di.uminho.pt/), provide preservation features. However they are complex for small – medium organizations and/or demand a number of pre-requisites to be setup.

This paper presents Mopseus, a digital library service, inspired by the conceptualization of [11] and built on top of Fedora-commons middleware that provides repository development and management services in combination with basic preservation workflows and functionalities. These functionalities are based on an infrastructure that semantically correlates the repository content. Mopseus is designed to facilitate institutions to develop and preserve their own repositories [1]. In comparison to the Fedora-commons platform, Mopseus provides a repository system, without the need of customization and the programming workload that Fedora-commons involves. Additionally, Mopseus indexing process is based on a RDMS, ensuring efficiency.

The main objective of the paper is to present an enhancement of the preservation features of Fedora-commons platform implemented by Mopseus. Mopseus is based on an expressive data model aiming to enrich the vocabulary of relationships between the entities of the Fedora-commons model, which are the repository objects and the data structures they contain. The proposed vocabulary revises and improves the existing relationships and defines them explicitly and formally using RDFS. This extension enables the management of information concerning the provenance of the Digital Objects. The new data model is mapped to PREMIS data model [12] in order to automatically generate and incorporate valid PREMIS metadata in Fedora-commons repositories. Each time a workflow, consisted of a number of events, is carried out and affects the status of a set of repository objects, then PREMIS metadata are automatically generated for each affected object and stored in the repository. Thus the PREMIS metadata generation mechanism is integrated with the Mopseus workflow management component, modifies essentially the logging mechanism and enriches the FOXML [7] schema of Fedora-commons.

In the next section the Mopseus architecture is outlined and its data model and main functional components are presented. In section 3 the main principles on which the preservation features of

Mopseus are based as well as the mapping of Mopseus data model to PREMIS data model are presented. Furthermore the implementation of PREMIS metadata generation mechanism is demonstrated. Finally in section 4 the Mopseus innovative features are discussed and in section 5 the main conclusions of the presented effort are sketched.

2. ARCHITECTURE

2.1. Data Model

Mopseus is based on the main Fedora-commons entities which are the *digital objects* and *datastreams* and provides an ontology that defines the relationships between them. In particular the content of a Mopseus repository is stored as *digital objects*, consisting of *datastreams*, which can be text/xml, text/rdf or binary (see Figure 1). Thus Datastreams can be correlated to form Digital Objects that are structures of data and metadata. Each Digital Object is described at least by a Dublin Core record implemented as a Datastream. Additional descriptive metadata, following any schema, could be incorporated as Datastreams. A new entity enhancing the Fedora-commons conceptual schema is the *container*. A Container is a Digital Object which aggregates a set of Digital Objects or other Containers. For instance a collection of the PhD theses of a University Department is a Container, which consists of several Digital Objects (PhD theses) and may belong to anotherContainer, e.g. the collection of the University's gray literature.

Each Mopseus Digital Object is an instance of one of the following entities named namespaces:

- **config**: The configuration of the repository itself is encoded by and stored as Digital Objects of this namespace. This makes Mopseus a self-describing repository, which means that all information regarding the setup of the repository is stored as Digital Object itself and thus is preserved following homogeneous and common preservation mechanisms.Thus, the required knowledge an administrator needs to have in order to configure and maintain the repository is XML.
- **cid**: This namespace contains Digital Objects that describe Containers. Containers can hold metadata (e.g. DC), binary Datastreams (e.g. a Thumbnail image) and can form any kind of graph through RDF relations.
- **iid**: This namespace contains all the Digital Objects that carry actual information (items), consisting of Datastreams.
- **trm**: This namespace contains all Digital Objects that carry terminology information. These Digital Objects are encoded in SKOS and each Digital Object that resides in the trm namespace represents a SKOS concept.

Another significant entity of the model corresponds to the notion of workflow, which is a sequence of states (or events). Each state incorporates a set of basic operations performed on Digital Objects or Datastreams. The descriptions of workflows, states and their basic operations are stored as Datastreams, in the form of XML documents, and they constitute a part of the Digital Objects description in the config namespace.

The Mopseus data model relationships are categorized to the following classes:

- Digital Objects relations: A Digital Object may be correlated to one or more Containers (or other objects) through a set of partitive or membership relations given by Fedora-commons ontology (http://www.fedora-commons.org/definitions/1/0/fedora-relsext-ontology.rdfs) and enriched by the DCTERMS vocabulary for the DC Relation property [6], forming thus a new ontology named RELS-EXT.

Relationship	Domain	Range	Description
isRDF	Object	Datastream	Denotes that a Datastream of an object is an RDF document.
isThumbnail	Object	Datastream	Denotes that a Datastream of an object is thumbnail.
isImage	Object	Datastream	Denotes that a Datastream of an object is an image.
isImageHighDef	Object	Datastream	Denotes that a Datastream of an object is a high resolution image.
isDocument	Object	Datastream	Denotes that a Datastream of an object is a document.
isDocumentPDF	Object	Datastream	Denotes that a Datastream of an object is a document.
isBinary	Object	Datastream	Denotes that a Datastream of an object is a bitstream.
migratedFrom	Object	Datastream	When an object is migrated by a repository, a Datastream is generated, holding all information about its provenace.

Table 1. Mopseus Digital Object – Datastream (RELS-INT ontology) relations

- Digital Objects - Workflows: The state of a Digital Object could be modified by a workflow meaning that there exists a correlation between a workflow and one or a set of particular affected Digital Object/objects. These relations are aligned to the vocabulary of PREMIS EventType element [12] and are directly implemented through the Mopseus services.
- Digital Objects - Datastreams: A Digital Object is consisted of one or more Datastreams through a rich

vocabulary of relations referred in Table 1. These relations enrich the semantics of Fedora-commons ontology. A crucial relationship for preservation repositories is the migratedFrom which denotes the incorporation of a Digital Object from other repositories.

All relationships are described in RDFS and stored in Datastreams. Specifically, two Datastreams residing in the config namespace have been implemented:

- RELS-EXT. Contains a slightly enhanced version of the Fedora provided RELS-EXTontology for describing relationship types between Digital Objects and Containers, such as isPartOf, etc.

- RELS-INT. Contains an ontology for characterizing the constituent Datastreams of a Digital Object and the relationships between them e.g.*isDocument*, *isThumbnail*, etc. Its main classes and relationships are presented in Table 1.

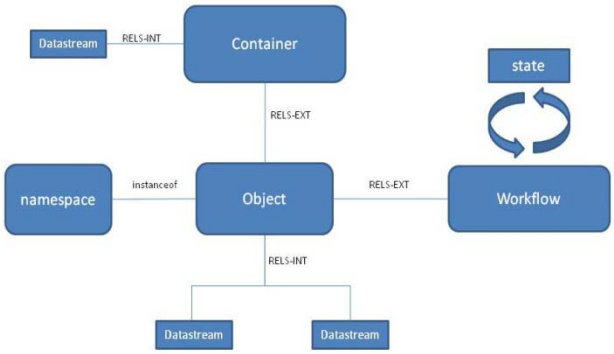

Figure 1. Mopseus data model

2.2. Functional Components

Mopseus consists of two different and distinct parts: a backend which implements the core services of Mopseus (written in Java) and a frontend PHP and JSP API which provides out of the box functionalities that are used to create the different web based GUIs. Only the backend part of Mopseus and certain Fedora-commons functionalities, such as the REST based retrieval mechanisms, communicates directly with Fedora-commons. The main architectural components of Mopseus (see Figure 2) are:

- **Dynamic definition of XML schemas**. Mopseus provides a service for the definition of metadata schemas. The service supports the development of XML schemas, defining the syntax of the metadata elements, their functionality (mandatory/optional elements) and presentation. A new XML schema is automatically transformed into HTML forms and the user can use them to ingest metadata and produce valid XML documents stored as Datastreams. The service that translates the XML schema definition to a working HTML form also supports a number of other features such as: creating an object from templates, creating an object from a mapping mechanism (see below), etc. The metadata schema definitions are stored in the *schemas* Digital Object of the config name space.

- **Relations manager**. The relations manager allows for the easy management (insert, delete) of relations both external (between objects and Containers) and internal (between Datastreams). The parameters of this service are the Datastream on which a relation should be added, deleted or modified and the ontology that keeps the Mopseus relationships. The service allows the user to define relations in a flexible manner and use different ontologies on different Datastreams.

- **RDBMS Synchronization**. A mechanism was developed to dynamically synchronize any or all the elements of the hosted XML schemas with an external RDBMS database (currently MySQL is supported). This process features a flexibility which is achieved by automatically mapping XPath Queries to SQL queries. Furthermore, this mechanism can also store in the RDBMS RDF information (e.g. relations) and Datastream information. This process drastically improves the efficiency and flexibility of the indexing of any kind of XML or RDF document stored in Datastreams, makes easier the implementation of a web frontend system and the searching process. All the RDBMS synchronization information are stored in the *sync* Digital Object of the config namespace.

- **Mapping between XML schemas**. This mechanism allows the mapping between metadata schemas. The mapping is created through an XSLT tansformation. The mapping service can take as parameters a Datastream that contains the XSLT transformation, the Datastream containing the source XML document and the target Datastream. It then can automatically perform the transformation and store the result onto the target Datastream. The XSLT document itself along with the mapping rules are stored in the *mappings* Digital Object (config:mappings) of the config namespace.

- **Workflow engine**. The workflow engine allows for executing sequences of states, such as *ingestion*, *revision*, etc. For each state its input parameters from the previous states and output parameters, which pass to the next states are described. A state invokes a specific service, which in general would be either internal (i.e. one of the mentioned Mopseus components, e.g. a mapping between XML Schemas service) or external (e.g. the use of a data format migration tool, which is not part of Mopseus). Notice that the current version of Mopseus does not support external services.

- **Preservation service**. This service is responsible for the preservation features of Mopseus and provides the following functionalities: (a) maintains a PREMIS log per Digital Object (residing in the PREMIS Datastream) containing all actions and operations that take place on a Digital Object, (b) maintains and checks the checksums for each Datastream and (c) performs simple migrations on binary Datastreams such as PDF documents (this service is currently under development).

- **Terminology service**. The terminology service allows for management of vocabularies, which can then be used in metadata schemas. Information regarding this service is stored in the *terms* Digital Object

(config:terms) of the config namespaceDigital Object. The terms are represented in SKOS.

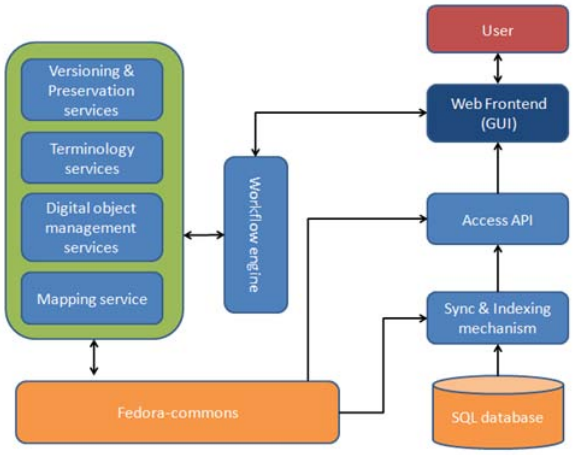

Figure 2. Mopseus architecture. This figure illustrates the basic high-level components of Mopseus and how they inter-operate.

Constructing a user interface in Mopseus is relatively easy. The developer can utilize ready to use components such as an Admin Panel which allows the user to perform operations on Digital Objects, relations and Datastreams. Furthermore, item short views (see Figure 3b) as well as detailed views (see Figure 3c) can be obtained directly from corresponding ready to use XSLT files. Binary Datastreams available for viewing and downloading (e.g. isDocumentPDF, see Figure 3d) can be displayed based on the RELS-INT relations. Finally the Containers an object belongs to can be obtained by the relevant RELS-EXT relations (see Figure 3a).

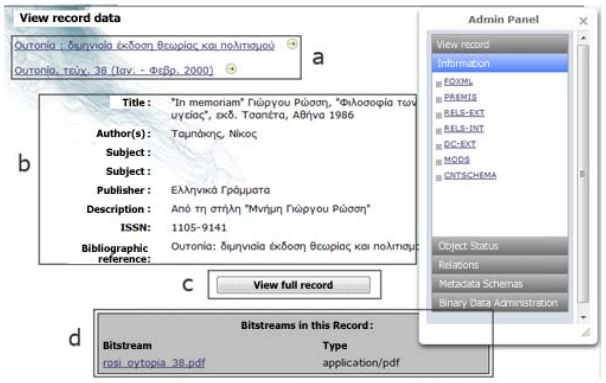

Figure 3. A screenshot of the Mopseus installation at Panteion University.

3. PRESERVATION STRATEGY

3.1. Outline of Strategy

Mopseus preservation strategy follows a set of rules that aim towards a long-term storage and access to the Digital Objects, with respect to small and middle-sized institutions, with a probable low budget. Mopseus is inspired by the OAIS [4] model principles in the sense that (a) the Digital Objects carry meaningful information about their binary content and relationships and (b) this representation information constitutes itself a Digital Object. Thus each Digital Object contains a set of Datastreams and relations. The Datastreams carry both representations of an object and the object's descriptive metadata.

Moreover, Mopseus supports ingestion, access, storage, data management, administration and preservation planning OAIS functionalities. In compliance with the OAIS the Submission Information Packages (SIPs) are transformed to Archival Information Packages (AIPs) with the use of a set of internal Fedora-commons mechanisms. The ingested Digital Objects are checked for integrity, descriptive metadata are generated semi-automatically via the mapping mechanism and preservation metadata are generated automatically in PREMIS. All the Digital Objects are preserved by the Fedora-commons mechanisms, which keep versions of the repository state and content. The versions of a repository are stored internally. Regarding preservation planning, Mopseus provides a migration process from other existing repositories, facilitated through the use of a desktop tool implemented in Java. Currently it supports migration from DSpace repositories. The final Dissemination Information Packages (DIPs) are promoted to consumers through a web-front that selects specific aspects of the Digital Objects to show to the end user.

One of the most representative installations of Mopseus is Pandemos, the digital library of Panteion University, Athens, Greece (http://library.panteion.gr/pandemos). Originally, Pandemos was a DSpace repository, holding approximately 2200 Digital Objects, migrated to Mopseus without any loss of information and at least 5000-5500 new Digital Objects were ingested. For the migration process, the migration tool mapped the DSpace communities, collections and subcollections to the Mopseus Containers by creating the appropriate RDF relationships with the Containers. The original metadata from DSpace were preserved into Mopseus while a PREMIS event was created to indicate the preservation action.

3.2. Mapping the data models

The generation of valid PREMIS metadata presupposes the mapping of Mopseus and PREMIS data models. Mopseus data model was presented in the previous section and PREMIS data model is briefly presented as follows [12]:

The PREMIS data model consists of five entities, according to Figure 4: the *Intellectual Entity* ("a coherent set of content that is reasonably described as a unit"), *Object* ("or Digital Object, a discrete unit of information in digital form"), *Event* ("an action that involves at least one object or agent known to the preservation repository"), *Agent* ("a person, organization, or software program associated with preservation events in the life of an object") and *Rights*, ("or Rights Statements, assertions

of one or more rights or permissions pertaining to an object and/or agent"). The Objects are categorized to three types: *file* ("a named and ordered sequence of bytes that is known by an operating system"), *bitstream* ("contiguous or non-contiguous data within a file that has meaningful common properties for preservation purposes"), and representation ("the set of files, including structural metadata, needed for a complete and reasonable rendition of an Intellectual Entity").

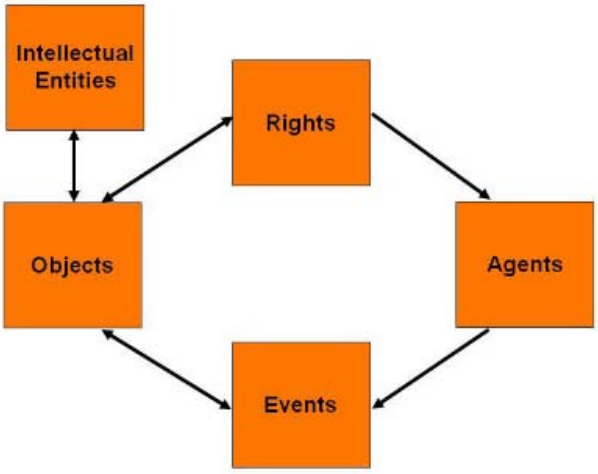

Figure 4. PREMIS data model

The relationships associate the instances of entities. PREMIS relationships associate the instances of the Object entity as well as the instances of entities of different types. The properties between objects are categorized to three types: *structural*, which are relations between the parts of objects, e.g. the relationships between the files that constitute a representation of an Intellectual Entity, *derivation* relationships, which result from the replication or transformation of an Object, when "file A of format X is migrated to create file B of format Y, a derivation relationship exists between A and B" and *dependency* relationships which "exists when one object requires another to support its function, delivery, or coherence of content". The relationships between different entities are expressed by including in the information for the first entity, a pointer to the second entity.

The mapping of two data models is defined as a sufficient specification to correlate each instance of the source model with the instances of the target model with the same meaning. The mapping of the two models is presented in Table 2 and analyzed as follows:

The central entity in both models is the Digital Object, though there exist semantic variations between them. A Mopseus Datastream that carries either a binary file or metadata is mapped to the File Category of a PREMIS Object. Moreover a Mopseus Digital Object is mapped to the Representation Category of an Object, since it represents a digital artifact with its binary representation(s) and metadata. Finally the Mopseus entity Container, actually represents a logical aggregation of objects and therefore is mapped to the PREMIS Intellectual Entity, noting that the Intellectual Entity refers to a collection of Objects of the Representation Category.

Mopseus entities	**PREMIS entities**
Datastream / binary	Object/ file
Datastream / metadata	Object / file
Digital Object	Object /Representation
Container	Intellectual Entity
Workflow / State	Event
Datastream / metadata / Rights	Rights
Datastream (metadata) / Person	Agent
Mopseus relationships	**PREMIS relationships**
Digital Object - Datastream	Structural relationships
Digital Object - DC Datastream (Fedora-commons default metadata)	Dependency relationships
Digital Object - Digital object (through a Workflow / State	Derivative relationships
Container - Digital Object	Relationships between different types
Workflow / State - Digital Object	Relationships between different types
Workflow / State - Agent	Relationships between different types

Table 2. PREMIS - Mopseus mapping

The Mopseus entity Workflow refers to a sequence of events and thus its subclass State is mapped to the PREMIS Event entity, which represents a particular action in the time-line. The classes Agents and Rights are not expressed explicitly in the Mopseus data model; nevertheless all information that correspond to these entities is stored and available in the Datastreams that hold the metadata of each Mopseus Digital Object.

Concerning the relationships of the two models, it should address the main differences of the two models. Mopseus defines particular relationships with clear semantics, while PREMIS defines relationship categories. Based on this clarification, the relationships between a Mopseus Digital Object and its Datastreams are Structural. In particular Mopseus data model enriches the vocabulary of PREMIS structural relationships and this is obvious by the descriptions of the relationship semantics presented in Table 1. The existence of at least one Datastream that hold the main metadata of a Digital Object expressed in Dublin Core terms is mandatory for Fedora-commons repositories, including Mopseus, defining thus a dependency relationship. The modifications of the Digital Objects, generated by the performance of a Workflow or State, define derivation relationships between a Digital Object.

Since a Container is mapped to an Intellectual Entity, the relationships between a Digital Object and a Container are defined as a Relationship between different types. This type of relationships employs the PREMIS vocabulary, enriched by a variety of terms that belong to the DCTERMS Relations vocabulary. Finally the relationships between a Digital Object and a State of a Workflow as well as the relationships between the

State of a Workflow and an Agent are categorized to the PREMIS categories of relationships between different types of entities. It should be noticed that these relationships are expressed similarly by both the models: The information of the domain entity of each relationship includes a pointer to the identifier of the instance of the range entity. For instance the metadata of the State of a Workflow contain the identifier of Digital Object which participates to the State.

Given the mapping of the two models, the next step is the automated generation of valid PREMIS metadata. This process is based on an XSLT document which retrieves the metadata kept in the Datastreams of a Mopseus Digital Object and writes them in an XML document that follows the PREMIS syntax; this document is stored in a new Datastream, which is updated on each modification of the Digital Object. This process is triggered when a new Digital Object is ingested in Mopseus as well as at each modification of it. This process is described in the next paragraph.

3.3. Generating PREMIS Metadata

Fedora-commons keeps a log of the operations that take place in the repository encoded in FOXML. Mopseus keeps a more detailed log in PREMIS. The service that maintains the log is invoked whenever a user (or a service) performs a *write* operation on the repository. This operation mainly includes the creation of Digital Objects (manually or from a migration service), the creation or modification of Datastreams, the creation or deletion of RDF relations, etc. For each Digital Object there is a log kept in the PREMIS Datastream. A sample of the log can be seen in Figure 5. Most information that is required for the creation of the PREMIS log is taken from various Digital Objects of the config namespace. For instance, the eventIdentifierType is encoded in the config:repository XML Datastreams whereas information regarding different services can be found in the config:services Digital Objects. Regarding the relationships, each Mopseus relationship described in the config:ontologies Digital Object, is mapped to the corresponding PREMIS relation type.

4. DISCUSSION

Many approaches towards building a digital repository with preservation functionalities have been implemented. We briefly present and compare them with the key features of Mopseus. Both CASPAR [8] and PLANETS [9] aim at providing a set of direction on creating practical services and tools for the purpose of long-term access and preservation, without providing an actual digital repository, and therefore are out of the scope of this discussion.

One of the most widely known and used repositories worldwide is DSpace [2] which provides an out of the box solution for grey literature management in institutional repositories. However, it doesn't address the preservation of its Digital Objects as efficiently as other repository platforms, it lacks in flexibility since it only allows flat and relatively simple metadata schemas and it limits the organization of Digital Objects by providing only a few level hierarchy of Digital Objects.

```
<premis
xmlns:premis="http://www.loc.gov/standards/premis" >
   <premis:objectIdentifier>

<premis:objectIdentifierType>hdl</premis:objectIdentifier
Type>
<premis:objectIdentifierValue>iid:1011</premis:objectIden
tifierValue>
   </premis:objectIdentifier>
   <agent>
      <agentIdentifier>
         <agentIdentifierType>uid</agentIdentifierType>
         <agentIdentifierValue>13</agentIdentifierValue>
      </agentIdentifier>
      <agentName>Dimitris</agentName>
      <agentType>user</agentType>
   </agent>
   <agent>
      <agentIdentifier>
         <agentIdentifierType>servlet</agentIdentifierType>
<agentIdentifierValue>org.dcu.mopseus.DigitalObject</age
ntIdentifierValue>
      </agentIdentifier>
      <agentName>modifyObject</agentName>
      <agentType>service</agentType>
   </agent>
   <premisEvent:event
xmlns:premisEvent="http://www.loc.gov/standards/premis/
v1">
      <premisEvent:eventIdentifier>
<premisEvent:eventIdentifierType>MIS</premisEvent:eve
ntIdentifierType>
<premisEvent:eventIdentifierValue>modification</premisE
vent:eventIdentifierValue>
      </premisEvent:eventIdentifier>
      <premisEvent:eventType>modify digital
object</premisEvent:eventType>
      <premisEvent:eventDateTime>2010-05-
04T17:48:39</premisEvent:eventDateTime>
      <premisEvent:eventDetail>modify digital object
(dLabel=, state=A)</premisEvent:eventDetail>
      <premisEvent:linkingAgentIdentifier>
<premisEvent:linkingAgentIdentifierType>uid</premisEve
nt:linkingAgentIdentifierType>
<premisEvent:linkingAgentIdentifierValue>13</premisEve
nt:linkingAgentIdentifierValue>
<premisEvent:linkingAgentRole>user</premisEvent:linkin
gAgentRole>
      </premisEvent:linkingAgentIdentifier>
      <premisEvent:linkingAgentIdentifier>
<premisEvent:linkingAgentIdentifierType>servlet</premis
Event:linkingAgentIdentifierType>
<premisEvent:linkingAgentIdentifierValue>org.dcu.mopse
us.DigitalObject</premisEvent:linkingAgentIdentifierValue
>
<premisEvent:linkingAgentRole>servlet</premisEvent:link
ingAgentRole>
      </premisEvent:linkingAgentIdentifier>
   </premisEvent:event>
</premis>
```

Figure 5. A PREMIS Datastream

eSciDoc [14] is a powerful e-Research middleware infrastructure providing innovative services focusing on the researchers collaboration and the management of their resources. It is based on Fedora-commons repository management software on which a new data

model is defined. An eSciDoc Object is represented by multiple manifestations organized in Components. Each component includes the manifestation metadata and the content itself. A single eSciDoc Object may be a composition of Fedora-commons Digital Objects correlated by whole/part or parent/child relationships. To facilitate the view of an eSciDoc Object as one entity, eSciDoc extends the Fedora-commons versioning mechanism by maintaining a datastream for each eSciDoc object that keeps track of all Fedora-commons digital objects modifications. Regarding preservation, eSciDoc incorporates JHOVE tool. Moreover the complexity of objects involved in e-Research, as well as the lack of appropriate metadata standards for their description, constitutes a barrier for the development of a concrete preservation strategy.

DAITSS [3] is a digital preservation repository application developed by the Florida Center for Library Automation and is intended to be used as a back-end to other systems, thus it has no public access interface, though it can be used in conjunction with an access system. The DAITSS system is a java application which handles all DAITSS functionality, a MySQL database to manage its archival collections and a storage back-end where DAITSS stores the information packages. DAITSS is designed to implement active preservation strategies based on format transformations including bit-level preservation, forward migration, normalization, and localization. It implements OAIS, it uses METS [10] and has a partial compliance with PREMIS. In short, the DAITSS is a functional digital repository application that is able to perform on a large scale. There is no front-end to support the preservation functions.

The British Library's eJournal system [5] is a system for ingest, storage and preservation of digital content developed under the Digital Library System Programme, with eJournals as the first content stream. It is an implementation of OAIS, making use of the British Library's METS, PREMIS and MODS application profiles. The AIP is tied to the technical infrastructure of the British Library's preservation system, that consists of an ingest system, a metadata management component and an archival store, and is linked with the existing integrated library system (ILS). The eJournal data model contains five separate metadata AIPs, journals, issues, articles, manifestations and submissions, with each being realised by at least one METS document. Descriptive metadata are stored as a MODS extension to the METS document, while provenance and technical metadata are captured as PREMIS extensions. Events related to the digital material are being recorded as provenance metadata and can be associated with any object type. The British Library's eJournal system is an example of use of a combination of existing metadata schemas to represent eJournal Archival Information Packages in a write-once archival system.

RODA is an open source service-oriented digital repository developed by the Portuguese National Archives. RODA is based on existing standards such as OAIS, METS, EAD and PREMIS and has the Fedora Commons at the core of its framework. RODA specifies workflows for each off the three top processes of the OAIS model (ingest, administration and dissemination). Every Digital Object being stored in RODA is subjected to a normalization process. RODA makes use of the Fedora main features adding to them a set of RODA Core Services. RODA also provides a web interface to allow the end user to browse, search, access and administrate stored information, metadata, execute ingest procedures, preservation and dissemination tasks. RODA supports a set of preservation services, such as (a) file format identification, (b) recommendation of optimal migration options, (c) conversion of Digital Objects from their original formats to more up-to-date encodings, (d) quality-control assessment of the overall migration process, (e) generation of preservation metadata in PREMIS format. RODA is a complete digital repository providing functionality for all the OAIS main units and a set of preservation services developed around Fedora.

Mopseus presents most similarities and shares a similar approach with RODA since both digital repositories are Fedora based and implement the OAIS model for storing and disseminating Digital Objects. However Mopseus does not encapsulate the Datastreams in METS documents, but correlates them semantically utilizing the internal ontology RELS-INT, while the Digital Objects are correlated with the Containers via the RELS-EXT ontology. Moreover since Mopseus is focused on small and middle sized institutions can be easily installed under different platforms and requires low implementation and support expertise. One of the most powerful features of Mopseus is the flexibility in defining and mapping of metadata schemas and generating preservation metadata. These features along with the collection migration functionality render Mopseus a repository management platform adaptive to the preservation needs of several types of small and medium sized information organizations.

5. CONCLUSIONS

Mopseus is an easily configurable open source repository management system, adaptive to the digital content and needs of a variety of information organization types. It enhances Fedora-commons platform with a powerful data model providing a set of semantically rich relationships between the content and its metadata, including information concerning the provenance of them. Moreover it provides powerful functionalities for metadata schemas definition and automated preservation metadata generation, while it offers mechanisms from migrating content from other repositories. These features enable information providers to manage and preserve their digital holdings.

Among the plans for Mopseus further development is the addition of workflow wizards to the workflow engine, to guide users to define, plan and perform content management activities using friendly and usable interfaces. Mopseus does not provide a format migration mechanism due to its low cost approach. Future work includes the development of an API on which a variety

of preservation planning tools such as PLATO, and format migration tools can be incorporated in the Mopseus environment. After these improvements a large scale user-based evaluation experiment will be conducted to investigate the acceptance of MOPSEUS functionalities by the user and stakeholder (libraries, museums, archives and records management services) communities.

6. REFERENCES

[1] Angelis, S., Constantopoulos, P. Gavrilis, D., Papatheodorou, C. "A Digital Library Service for the Small", DigCCurr 2009: Procs of the 2nd Digital Curation Curriculum Symposium: Digital Curation Practice, Promise and Prospects, 2009. http://www.ils.unc.edu/digccurr2009/

[2] Bass, M.J., Stuve, D., Tansley, R. "DSpace – a Sustainable Solution for Institutional Digital Asset Services – Spanning the Information Asset Value Chain: Ingest, Manage, Preserve, Disseminate Functionality", Internal Reference Specification. http://www.dspace.org/technology/architecture.pdf

[3] Caplan, P. "The Florida Digital Archive and DAITSS: A model for digital preservation", Library Hi Tech, 28(2), 2010.

[4] CCSDS, Reference Model for an Open Archival Information System (OAIS), CCSDS 650.0-B-1, Blue Book (the full ISO standard), 2002. http://public.ccsds.org/publications/archive/650x0b1.pdf

[5] Dappert, A., Enders, M. "Using METS, PREMIS and MODS for Archiving eJournals", D-Lib Magazine, 14 (9/10), 2008. http://www.dlib.org/dlib/september08/dappert/09dappert.html

[6] Dublin Core Metadata Initiative, "DCMI Metadata Terms", http://dublincore.org/documents/dcmi-terms/

[7] Fedora Object XML (FOXML), http://www.fedora-commons.org/download/2.0/ userdocs /digitalobjects/introFOXML.html

[8] Giaretta, D. "The CASPAR Approach to Digital Preservation", The International Journal of Digital Curation 2(1), 2007. http://www.ijdc.net/ijdc/article/view/29/32

[9] King, R., Schmidt, R., Jackson, A., Wilson, C., Steeg, F. "The Planets Interoperability Framework - An Infrastructure for Digital Preservation Actions", ECDL2009: Procs of the 13th European Conference on Digital Libraries, 425-428, 2009.

[10] Library of Congress, "METS - Metadata Encoding & Transmission Standard". http://www.loc.gov/standards/mets/

[11] Meghini, C., Spyratos, N. "Viewing Collections as Abstractions" DELOS Conference 2007: Procs of the 1st International DELOS Conference, 207-217, 2007.

[12] PREMIS Working Group, "Data dictionary for preservation metadata: final report of the PREMIS Working Group", OCLC Online Computer Library Center & Research Libraries Group, Dublin, Ohio, USA, 2005.

[13] Ramalho, J. C., Ferreira, M. "RODA: A service-oriented repository to preserve authentic digital objects", Open Repositories, 2009. http://redmine.keep.pt/attachments/8/OR09-0.3.pdf

[14] Razum, M., Schwichtenberg, F.,Wagner, S., Hoppe, M. "eSciDoc Infrastructure: A Fedora-Based e-Research Framework", ECDL2009: Procs of the 13th European Conference on Digital Libraries, 227-238, 2009.

[15] Roberts, G. "Small Libraries, Big Technology", Computers in Libraries, 25(3), 24-26, 2005.