



omii europe
open middleware infrastructure institute



EU project: RI031844-OMII-Europe

Project no: **RI031844-OMII-Europe**

Project acronym: **OMII-Europe**

Project title: **Open Middleware Infrastructure Institute for Europe**

Instrument: **Integrated Infrastructure Initiative**

Thematic Priority: **Communication network development**

D:SA1.0 Design Document for the Repository Including Database Schema Definition

Due date of deliverable: July 2006
Actual submission date: September 12, 2006

Start date of project: **1 May 2006**

Duration: **2 years**

Organisation name of lead contractor for this deliverable: INFN

Revision [2.4]

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	Public
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Document Change History

Version	Date	Comment	Author/Partner
2.4	2006-09-07	Revised by Andrea Caltroni/INFN, Alberto Di Meglio/CERN and Steven Newhouse/SOTON	Andrea Caltroni/INFN, Alberto Di Meglio/CERN Steven Newhouse/SOTON
2.3	2006-08-29	Merged suggestions by Steven Newhouse/SOTON	Andrea Caltroni/INFN
2.2	2006-08-28	Merged suggestions by Alberto Di Meglio/CERN	Andrea Caltroni/INFN
2.1	2006-08-21	Revised	Antonia Ghiselli/INFN
2.0	2006-08-20	Revised	Alistair Dunlop/Project Leader
1.3	2006-08-12	Minor revision	Andrea Caltroni/INFN
1.2	2006-08-10	Revisions. Some sections expanded.	Andrea Caltroni/INFN
1.0	2006-07-31	First version	Andrea Caltroni/INFN
0.9	2006-07-20	Ideas from Steven Newhouse/SOTON merged	Andrea Caltroni/INFN
0.1-0.8	2006-07-01	First drafts	Andrea Caltroni/INFN

Table of Contents

1	Executive Summary	4
2	Glossary	5
3	Introduction.....	6
4	Goals of the Repository	7
4.1	General Scope	7
4.2	General and Detailed Objectives.....	7
4.3	Operations on Repository Objects	8
4.4	Contents of the Repository.....	8
4.5	Certification	9
4.6	Branding.....	9
5	Stakeholders and Users	10
5.1	Contributors	10
5.2	Downloaders	10
5.3	Build Agent.....	10
5.4	Repository Administrators	11
6	Use Cases	12
6.1	Access to the Repository.....	12
6.2	Access to the Repository (Administration).....	12
6.3	Automatic Population of the Repository from ETICS.....	12
6.4	Manual Upload of Software Components or Documentation.....	13
6.5	Manual Download of Components	13
6.6	User Management (Administration)	14
Note: Groups are used mainly to give users special privileges connected with the activities they need to perform on the repository.		
Implementation Constraints		14
Implementation Constraints		15
6.7	Standards.....	15
6.8	License	15
6.8.1	License for the Software Components	15
6.8.2	License for External Tools.....	15
6.9	Scalability of the Number of Stored Objects	15
6.10	Scalability of the Number of Accesses	15
7	Interfaces.....	16
7.1	Client Access API Description	16
7.2	Web-based Access (User Side).....	17
7.3	Administration API Description	17
7.4	Web-Based Access (Administration).....	18
8	Database.....	19
8.1	Overview.....	19
8.2	Database Schema Specification	19
8.3	Implementation	22
9	Security: Authentication, Authorization and Accounting.....	22
9.1	Authentication.....	22
9.2	Authorization	22
9.3	Accounting.....	22
10	Concluding Statement.....	23
11	References.....	24

1 Executive Summary

This document provides the current (September 2006) high-level design of the OMII-Europe repository.

It has been driven by the experiences from the SA1 participants in supporting related activities, the interactions with SA2 (Quality Assurance) and the primary user community.

We describe how we see the repository being exploited by the European and international research community to both find the software being used within OMII-Europe and to contribute their own software products from their projects. This user base drives the identification of the principal stakeholders within the repository and the primary use cases describe their interaction.

The authentication and authorisation of the stakeholders in carrying out these use cases is also considered.

A provisional database schema, derived from these use cases, which takes into account the current experience of ETICS and OMII-UK is provided as the foundation for the OMII-Europe repository.

2 Glossary

Component : A *software component* or any other packaged component.

Data Stream : Information associated to a software component which represent a data source, i.e. the data stream content. A software component can have many data streams.

Build Agent : An agent managing an external source of packaged components. The default build agent at the beginning will be an agent managing the ETICS infrastructure.

Data Stream Content : The physical source for a software component. It can be mime-typed data or metadata and can be stored locally or managed by an external source.

Harvesting : In the OAI context, harvesting refers specifically to the gathering together of metadata from a number of distributed repositories into a combined data store.

Protocol : a set of rules defining communication between systems. FTP (File Transfer Protocol) and HTTP (Hypertext Transport Protocol) are examples of other protocols used for communication between systems across the Internet.

Registered user : Any user with a certificate or a username/password recognized by the repository server.

Repository : A hardware and software infrastructure supporting the storage of software components and related information.

Repository Object : any basic entity stored directly or indirectly in the repository.

Software Component : a logical unit of software which is made of one or more repository objects.

Users : Administrators, downloaders and contributors.

3 Introduction

The broader objective of OMII-Europe repository activity is:

- to enable open source software projects within Europe to contribute their releases in order to promote their activity;
- to annotate the software within the repository with quality and interoperability statements;
- to support the adoption of e-Science software within European Industry by providing a repository of e-Science software of known quality;

OMII-Europe will capitalise on the number of Grid software components which have been developed over the years by attracting them into the repository. This will ensure that the investment provided by the European Commission and National Governments is exploitable after the end of the individual project.

Contributed components will be assessed on their quality, reliability and ability to run in different environments. This will be assessed against the major middleware stacks within OMII-Europe. These components may be tools, services or even complete middleware distributions. The OMII-Europe repository may contain source or binary versions of these software components, but where reliable pre-existing external project repositories exist users will be directed to these to obtain the relevant downloads.

The quality, interoperability, compliance and benchmarking statements provided by the repository will be generated by both the SA2 Quality Assurance activity and the JRA4 Benchmarking activity, from the contributions made into the repository made by the JRA1 Re-engineering and Infrastructure Integration activities. The quality and interoperability statements relating to these software components will be formed against their execution within the main middleware platforms.

The software repository will provide access to the results of the benchmarking and compliance tests, and the documentation and training material developed in the other OMII-Europe activities.

4 Goals of the Repository

4.1 General Scope

The SA1 Repository activity is one of several activities within the project seeking to establish and develop an archive of software components suitable for and usable by the wider scientific community.

The objectives of the service activities are to establish a repository containing useful, quality-assured software components, running on the major Grid middleware stacks, offering a degree of portability between those stacks. This will give the wider research community, comprising scientists not expert in the Grid, more confidence to adopt Grid-based solutions because their investment of time in developing such solutions will carry with it the comfort of knowing that some, at least, of the solutions they have developed will be able to move from middleware stack to middleware stack with a reduced, if not negligible, effort.

The final goal of SA1 within the initial 2 year phase of the OMII-Europe project is to establish a repository hosted by INFN. This repository may consist of a single instance or be a federation of several repositories. The federated repository capability is likely to come from unfunded work with the project.

The involvement of major global players in OMII-Europe will be an important first step towards the establishment of those distributed, federated repositories. This compositional model will also apply to the components placed within the repository - some software may be stored within the OMII-Europe repository or that of the original contributor.

4.2 General and Detailed Objectives

The main objectives of the repository are:

- exploit middleware components and Grid environments developed in other projects and available as Open Source software in order to build a set of coherent and interoperable services. As appropriate these services will be based on emerging Grid standards and Service-oriented architectures working towards the goal of hardware and operating system independence;
- deliver and certify a suite of well documented and supported Grid Services supporting early adoption and exploitation by industry and following where appropriate existing and emerging standards;

The expected outcome of the project activities involved in the creation of the repository will be:

- an integrated suite of software components running on the major Grid infrastructures available through a software repository to the scientific community and other interested parties;
- supporting the wider use of those infrastructures by the scientific and research communities;

The success and impact of this activity can be measured simply by the use made of the repository components, the number of downloads made and the use made of the associated support services and training.

In detail the SA1 activity should:

- interact with the other activities of the project to define the quality guidelines and the certification procedures and definition of the distribution license for software components willing to be distributed by OMII-Europe;
- interact with the other activities of the project to select components from the existing middleware distributions for initial inclusion into the repository;

- define the specifications for the OMII-Europe repository; this includes hardware specifications for the servers, analysis of the requirements and design of the server software and the user interface;
- develop the basic repository functionalities;
- initial population of the repository with the selected components;
- inclusion in the repository of other information relevant to OMII-Europe activities such as details of the Grid evaluation infrastructure, the support activity and the training;

4.3 Operations on Repository Objects

It should be possible to store any type of digital objects in the repository although most of the objects stored will be packaged software components.

The public APIs will allow performing operations on the repository objects, like accessing the metadata information of an object or obtaining the URL to reference it.

An interesting possibility would also be to perform operations which allow a different view on the object itself. If we had, for example, a software component stored as an RPM package, we would be able to obtain a reference to the same object packaged as a TAR.GZ object.

The APIs will also allow establishing links among objects. This could be useful to represent dependencies or group objects into categories.

Another interesting possibility would be to aggregate different objects into one. If we have, for example, many independent digital objects consisting in the documentation of a software package but stored as different entities, the user might be presented with a single logical object.

4.4 Contents of the Repository

We foresee the repository to contain:

- Components developed within OMII-Europe certified as working with the above distributions. These components will be available for download together with associated manuals, documentation and supplementary material;
- Components developed outside OMII-Europe certified as working with the above middleware stacks. These components will be available for download together with associated licence agreements, manuals, documentation and supplementary material;
- Information on OMII-Europe compliance and compliance-testing;
- Details of the OMII-Europe certification process and comparative test and benchmarking results;
- Information on the submission of components for inclusion in the repository;
- An archive of case studies showing the inter-working of the components with the principal middleware stacks;
- Links to sites where distributions of Grid middleware stacks can be downloaded together with associated manuals, documentation and supplementary material; Since the OMII-Europe repository is meant to complement and leverage the existing work of INFN in the C-OMEGA initiative, that of SOTON in the OMII-UK initiative, that of CERN in the ETICS project and that of BU and CNIC in CROWNGrid, linkages will be made to the OMII-UK, C-OMEGA, EGEE and NMI repositories.

4.5 Certification

One of the objectives of the project is to certify the components which are made available in the repository. Certification is key to the building of an OMII-Europe brand (please refer to the next section for more information on branding.)

This task will require SA2 to establish well-defined certification standards and processes for software components, documentation and tests.

These quality rules should be available to all contributors. The task of SA1 is to make them available in the repository.

Before a contribution will be accepted into the repository, it will have to satisfy all requirements stated in the certification process.

4.6 Branding

There will be an effort to establish an OMII-Europe brand associated with the software components distributed through the repository. The brand should carry with it the idea of reliable, robust and portable grid software.

The added value associated with the brand should encourage the submission of additional components by other projects, creating the premise for the success of the OMII-Europe founding principles.

5 Stakeholders and Users

5.1 Contributors

These individuals are contributing components (software, documentation, etc.) from their projects to the OMII-Europe repository. They will need access to add their projects, contribute new releases from their projects and to be able to provide and update information relating to their project.

They will also need access to the certification standards and procedures in order to be able to certify their tools for inclusion in the repository.

They may also wish to see activity relating to their releases – reviewing the history of their activity or monitoring the number of package downloads.

5.2 Downloaders

Users of the OMII-Europe repository will want to browse or search the available software (by keywords, category, etc.), its releases (by platform, version, middleware release, etc).

Downloaders will also be members of the partners of the project who will build their middleware releases from the software distributed by OMII-Europe.

5.3 Build Agent

From Annex I - Description of Work, (March 2006): *“The major risk of this activity is that there is insufficient interest from the target user base of the wider scientific and research community in the contents of the repository. The following strategy will be adopted to address this. Firstly OMII-Europe will have immediate access to the user bases of GLOBUS, UNICORE, EGEE and CROWNGrid. This will give the repository wide visibility to part of its user base. This part of the user base will be interested in using or assessing the components of the repository because they will be useful, adding value to an already used Grid infrastructure.*

Through this initial use of components, through increasing utility as further components are added and through diffusion of knowledge resulting from collaborative projects involving that user base, it is expected that greater and greater use will be made of the repository.”

In order to pursue this strategy, the cooperation with ETICS will be strategic from the beginning of the project.

One of ETICS' objectives is to establish an international and well managed capability for software configuration, integration, testing and benchmarking for the scientific community. Software development projects will use the capabilities provided by ETICS to build and integrate their software and perform complex distributed test and validation tasks.

OMII-Europe will initially setup and configure an ETICS *build server* whose main task will be to produce repository components taken from projects like EGEE 2 which are already using ETICS as their build system.

A *build agent* will administer the build server through a secure authorised access channel, adding new components to the automatic build procedure, removing source components or editing their configuration.

The build agent should also have the possibility to start, stop and monitor the build process from the repository administration pages.

The build agent will also be able to configure build parameters like the destination of the build output, i.e. the packaged components.

5.4 Repository Administrators

The administrators of the OMII-Europe repository will be able to control and manage the various entities described above: contributors, downloaders, build agent and the server configuration itself through a web interface.

6 Use Cases

6.1 Access to the Repository

Goal: Accessing the repository in order to perform operations on the contents of the repository, like searching for, downloading or contributing software components.

Primary Actor: End users (visitors and contributors).

Scenario:

1. User registers on the site and is provided with a username/password or installs a certificate which has been provided to her by an OMII-Europe certified CA, into her browser.
2. User logs into an unprivileged account and is able to perform a set of operations on the contents of the repository.

Notes: Accesses will be logged and monitored to prevent misuse.

6.2 Access to the Repository (Administration)

Goal: Accessing the repository in order to perform restricted operations on the contents of the repository, like administering users, operating on the components of the repository (removal of a component or update of its metadata and so on), allowing a manually contributed component into the repository, administering the build agent configuration.

Primary Actor: Administrators.

Scenario:

1. Administrator is provided with a restricted account either consisting of a username/password or a certificate which she installs it into her browser.
2. Administrator logs into her privileged account and is able to perform a set of administrative operations on the contents of the repository.

Notes: Accesses will be logged and monitored to prevent misuse.

6.3 Automatic Population of the Repository from ETICS

Goal: Given the cooperation between OMII-Europe and ETICS, as a first step in the population of the repository, a procedure is established to allow the integration of the output of the ETICS build process into the repository of OMII-Europe.

Components selected by OMII-Europe for repository inclusion are properly configured into an OMII-Europe-controlled ETICS server and the output is automatically made available to the OMII-Europe clients through the repository interface.

Primary Actor: Administrators and an automatic procedure.

Scenario:

1. The repository services are temporarily stopped if necessary.
2. An ETICS server is configured to produce packaged versions of the components selected by OMII-Europe for automatic population.
3. ETICS is configured to produce upload the packages into the OMII-Europe repository server or another location controlled by OMII-Europe.
4. The repository server updates or creates the profiles of all the packages.
5. The repository services are restarted and the packages are again visible to users.

Notes: This procedure in no way prevents the repository from connecting to other sources acting as content feeds.

As a matter of fact, it will be possible from the beginning to refer to a software component which is not directly stored into the OMII-Europe repository. The repository stores all the metadata identifying the component but the physical package will be referred to by an external link. Of course, external components abide by the same rules for repository inclusion and certification of all the other packages.

This allows us to completely decouple the management of the ETICS server from the OMII-Europe repository (even if they might share the same platform).

6.4 Manual Upload of Software Components or Documentation

Goal: A user wants to manually contribute a software component to the repository because. Not all components will be integrated in an automatic population procedure. The reasons might be that it is not possible to do so or it is not convenient or easy to have it integrated.

Primary Actor: End users (uploaders) and administrators.

Scenario:

1. A contributor makes sure her packaged component meets all the OMII-Europe standards.
2. She logs in into his account.
3. She follows the directions to a page where she can upload her contribution and fill in all the necessary metadata information associated with the component.
4. The component is uploaded into a staging area ready to be examined by an administrator.
5. An administrator logging into the repository administrative interface checks the component for compliance with OMII-Europe's quality and safety rules.
6. If no problem arises, the component is uploaded into the repository and an entry is created for it.

Notes: A procedure will be prepared to allow contributors to verify their components against OMII-Europe software standards.

6.5 Manual Download of Components

Goal: A user wants to download a software component. Repository components, like any other web-based content, are accessible via URLs that return mime-typed streams.

Primary Actor: End users.

Scenario:

1. A user logs in into her account.
2. She will be able to look for a component in the following ways:
 - a. She can search for it with a search engine. The search engine will offer the possibility to do a substring searches and also specify parameters to refine the search output. The list of available parameters is taken from the component's metadata and include: name, software category, middleware release, author and so on.
 - b. Browsing the contents of the repository following links of software categories, or other categorizations based on the same parameters used for the search like name, software category, middleware release, author and so on.
 - c. She can invoke directly the URL which points to a component using a REST-ful approach build into the repository engine

3. She downloads the component.

6.6 User Management (Administration)

Goal: An administrator wants to access and operate on a user's profile through a secure administrative interface.

Primary Actor: Administrators.

Scenario:

1. Administrator logs into her privileged account.
2. She is guided to the user management area where she will be able to
 - a. Search for a user.
 - b. Confirm the registration of a new user.
 - c. Modify a user's profile including assigning her to a particular user group with predefined privileges.
 - d. Delete a user.
 - e. Suspend a user.

Note: Groups are used mainly to give users special privileges connected with the activities they need to perform on the repository.

Implementation Constraints

6.7 Standards

There are a few emerging standards in the fields of digital archival which will be closely considered in the implementation of the repository infrastructure.

We present here a list of those protocols just for reference.

The Open Archives Initiative Protocol for Metadata Harvesting:

<http://www.openarchives.org/OAI/openarchivesprotocol.html>

also referred to as OAI-PMH, provides an application-independent interoperability framework based on *metadata harvesting* [2].

The Reference Model for an Open Archival Information System (OAIS):

<http://public.ccsds.org/publications/archive/650x0b1.pdf>

which deals with the long term preservation of digital information.

The Dublin Core Metadata Initiative also provides standards to facilitate the finding, sharing and management of information.

<http://dublincore.org/>

The Dublin Core Metadata Initiative is an open forum engaged in the development of interoperable online metadata standards that support a broad range of purposes and business models.

6.8 License

6.8.1 License for the Software Components

Software components will be made available as Open Source subject to a BSD or BSD-like non-viral license. This will maximise their uptake by enabling not only wide use and modification throughout the scientific community, but will also enable customisation and commercialisation by companies wishing to productise and support the components in the repository.

6.8.2 License for External Tools

External tools will be used in the implementation of the repository framework. Care must be used with those tools to ensure they are distributed with a license compatible with the OMII-Europe environment.

6.9 Scalability of the Number of Stored Objects

The objective of OMII-Europe is to demonstrate a proof of concept using a small and agile consortium over a period of two years. This successful proof of concept would then provide a basis for a significant enlargement of the activity to include both industrial vendors and a wider group of technology developers.

For this reason, in the long run the repository should be able to store a significant amount of data and information although the physical resources should not necessarily be available in the short term. The short to medium term goals will be to make sure the software structure is capable of managing an increasing number of components.

6.10 Scalability of the Number of Accesses

For the same reason of the above section, the hardware and software of the repository server should be able to accommodate for an increasing number of user accesses from the user interface or from direct requests from automated tools.

7 Interfaces

The repository services will be exposed to the outside world through public APIs based on web services.

These public interfaces will serve two categories of agents:

- Software agents will be able to invoke the methods provided by the interfaces to interact with the repository and obtain the services the repository exposes.
- End users will access a web-based portal using HTTP-enabled tools like web browsers. The portal will be based on the APIs, but will present a more user-friendly interface conceived for manual interaction. That doesn't exclude automatic access to the portal by software agents. This will possibly be facilitated by the use of a REST-based syntax.

REST or Representational State Transfer is a software architecture style based on the concept of *resources* which, in this context, is a short name for *sources of specific information*. Each resource can be referred to using a global identifier (URI). Resources are usually accessed via HTTP.

Two main APIs are forecast: a *Client access API* and an *Administration API*.

We will present the main areas that will be covered by these two APIs and how these APIs will serve the web-based interface.

For a detailed description of the APIs including methods with parameters and full documentation, please refer to the upcoming executive design document.

The web site, on the other side, will provide ways to interact with the underlying repository in the following areas:

- Browsing the contents of the repository.
- Searching the contents of the repository. This includes the possibility to do searches based on parameters in the object metadata like name substring, category, author and so on.
- Performing administrative tasks.

7.1 Client Access API Description

The client access API defines an open interface for performing operations on the contents of the repository (software components, documentation, and so on). It is intended to respond to a client's request to discover information about the repository itself, an object and to perform certain operations on an object.

The areas which this interface will cover are:

- *Methods to discover information about the repository.* The repository entity will be formalized and a number of operations aimed at finding out a description of the repository will be provided.

This API will take into account the fact that the project's long term objective is to explore the implementation of distributed repositories and the current implementation will make sure there will be no conflicts when setting up support for distributed repositories.

- *Methods to discover information about a single repository object.* These methods will allow exploring the metadata information of an object stored in the repository which form the object's profile including information on how the object can be accessed like if it's stored locally or at a remote external location.

It will also be possible to access information about the history of the object in the repository.

- *Methods to operate on a collection of repository objects.* These methods will operate on collections of objects through various parameters. It will be possible to retrieve collections of objects based on given parameters.

7.2 Web-based Access (User Side)

These are the relevant web pages or functionalities necessary to support the users of the repository:

- Pages to search for components in the repository using parameters taken from an object's metadata like software category, author, release date, middleware release, and others.
- Pages to list the existing components in some type of order (alphabetical, by category, ascending or descending, etc) allowing the repository to be browsed. This same page might be used to list the results of a search. Items in the page will be paged. Each item will have a link/button to allow the direct download of the component or be instructed on how to do so.
- Pages to submit a new software component. Registered users will have the possibility to submit a new component to be evaluated for inclusion in the repository. The page will provide all the fields necessary to build the object's metadata.
- Page showing the details of a component.

7.3 Administration API Description

The administration API defines an interface for performing administrative operations on the contents of the repository. It is intended to be used by administrators accessing the web-based administrative interface, but it will also serve users performing controlled administrative tasks like uploading a software component to be later reviewed by the repository managers for inclusion.

Many functions will implement the so-called CRUD operations. CRUD stands for Create, Read, Update and Delete and reflects the basic type of operations which can be performed on the data of a database.

The Create operation will include methods to import and export a digital object or a collection of objects into the repository or vice-versa.

The areas which this interface will cover are:

- *CRUD operations on an object.* It will be possible for administrators to perform the basic database operations on a single object. Create a new object or import the metadata information, update the metadata associated to an object, retrieve the object's profile data and delete an object.
- *CRUD operations on a collection of objects.* It will be possible for administrators to perform some of the basic database operations (where it makes sense) on collections of objects.
- *Methods to discover information on a user.* User administration will be one of the main functionalities of the administrative interface. Administrators will need methods to create a new user, update his/her profile, retrieve the data for a user's profile or delete the user.
- *Methods to control the build agents.* We will need methods to control the administration operations which can be performed on the build agents. A first step could be to integrate the administrative interface of ETICS into the repository administrative interface.

7.4 Web-Based Access (Administration)

All pages available to end users will also be available to administrators. Some of these pages will have extra functionalities granted by the special role of the person who's logged in.

- Pages to list the existing components in some type of order (alphabetical, by category, ascending or descending, etc) allowing the repository to be browsed. This same page might be used to list the results of a search. Items in the page will be paged.
Each item will have a link/button to allow the direct download of the component or be instructed on how to do so.
This is essentially the same page the users will have access to. In addition, administrators will see associated to each item a link/button to allow the *editing* or *removal* of the corresponding item.
- Pages to submit a new software component. Administrators will have the possibility to directly submit a new component for inclusion in the repository. The page will provide all the fields necessary to build the object's metadata.
- Page showing the details of a component. An administrator will be able to modify the metadata associated to a component.
- Pages for user administration. It should be possible list/search the registered members and edit or remove a member's profile.
- Page to set the preferences for the OMII-Europe ETICS build server.

8 Database

8.1 Overview

The repository service handles both digital objects (software packages and related documentation in a variety of formats) and associated metadata. Digital objects may be persisted using a number of different technologies, from plain file system to sophisticated XML-encoding schemas that allow storing documents and even binary objects as encoded string representations in a database management system. The associated metadata can in turn be persisted in a variety of formats either a separate linked objects or together with the digital objects themselves. Several initiatives and standards have been proposed on the subject and the research on this topic is quite active. Some of these initiatives and standards have been mentioned earlier in this document (the Open Archival Information System - OAIS, the Metadata encoding and Transmission Standard – METS, and so on). In order to bootstrap the repository activity in OMII-Europe and provide a working proof-of-concept within the given timeframe, we will adopt a simple approach based on a backend to store the software packages and related documentation and a metadata store to describe the objects and their location. The principle of location independence will be followed, whereby access to the objects will not require direct knowledge of the object location or persistency mechanism by the user, but will be guaranteed by the information in the metadata store.

The metadata store will be based on a simple schema to describe the objects and related information. The schema is presented in the following section with the understanding that it may evolve or change as the OMII-Europe project evolves and as a consequence of implementations choices and interaction with other actors (like the ETICS project).

8.2 Database Schema Specification

The schema presented here is a first attempt at providing a high-level specification for the OMII-Europe repository metadata. The goal of this schema is to provide a way of describing software objects and the possible context where the objects have to be used (like releases for example). The schema is expressed using a simple XML format. In order to simplify the representation of the schema in this document, only the most relevant properties of each object are listed. It is understood that additional properties may exist now or in the future if required to support additional requirements.

```
<?xml version="1.0" encoding="utf-8" ?>
<xsd:schema targetNamespace="http://www.omii-europe.org/XMLSchema.xsd"
            elementFormDefault="qualified"
            attributeFormDefault="unqualified"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:element name="project">
        <xsd:annotation>
            <xsd:documentation>A project represents a high level
aggregation of software objects and documentation.
            It can be an organization, a software development project and
so on</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="author" minOccurs="0"
maxOccurs="unbounded" type="authorType"/>
                <xsd:element name="activeRoles" minOccurs="0"
maxOccurs="unbounded" type="roleType"/>
                <xsd:element name="allowedUsers" minOccurs="0"
maxOccurs="unbounded" type="userType"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

```

```

    </xsd:complexType>
    <xsd:attribute name="id" type="xsd:ID"/>
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="description" type="xsd:string"/>
    <xsd:attribute name="createDate" type="xsd:dateTime"/>
    <xsd:attribute name="lastModifiedDate" type="xsd:dateTime"/>
  </xsd:element>

  <xsd:complexType name="authorType">
    <xsd:annotation>
      <xsd:documentation>The author, contributor or maintainer of an
object in the repository.</xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="id" type="xsd:ID"/>
    <xsd:attribute name="fullName" type="xsd:string"/>
    <xsd:attribute name="emailAddress" type="xsd:string"/>
    <xsd:attribute name="createDate" type="xsd:dateTime"/>
    <xsd:attribute name="lastModifiedDate" type="xsd:dateTime"/>
  </xsd:complexType>

  <xsd:complexType name="userType">
    <xsd:annotation>
      <xsd:documentation>A registered user of the
repository.</xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="id" type="xsd:ID"/>
    <xsd:attribute name="fullName" type="xsd:string"/>
    <xsd:attribute name="emailAddress" type="xsd:string"/>
    <xsd:attribute name="FQDN" type="xsd:string"/>
    <xsd:attribute name="createDate" type="xsd:dateTime"/>
    <xsd:attribute name="lastModifiedDate" type="xsd:dateTime"/>
  </xsd:complexType>

  <xsd:complexType name="roleType">
    <xsd:annotation>
      <xsd:documentation>A role identifies a set of operations that
a user may perform on the repository objects.
      In order to perform an operation, a user must have an
associated role and the object must include
      the users in its list of allowed users and the role in its
list of active roles. The set of operations
      that a role allows to perform is not specified in this schema,
but it must be defined by the
      application implementing the schema</xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="id" type="xsd:ID"/>
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="createDate" type="xsd:dateTime"/>
    <xsd:attribute name="lastModifiedDate" type="xsd:dateTime"/>
  </xsd:complexType>

  <xsd:element name="release">
    <xsd:annotation>
      <xsd:documentation>A release is a collection of artifacts
(packages or documents) that has been approved
      for some type of usage by a project
administrator.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="packages" minOccurs="0"
maxOccurs="unbounded" type="artifactType"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:attribute name="status">

```

```

        <xsd:simpleType>
            <xsd:restriction base="xsd:string">
                <xsd:enumeration value="NEW" />
                <xsd:enumeration value="SC_PASSED" />
                <xsd:enumeration value="REJECTED" />
                <xsd:enumeration value="ALPHA" />
                <xsd:enumeration value="BETA" />
                <xsd:enumeration value="RELEASED" />
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="projectId" type="xsd:ID"/>
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="description" type="xsd:string"/>
    <xsd:attribute name="createDate" type="xsd:dateTime"/>
    <xsd:attribute name="lastModifiedDate" type="xsd:dateTime"/>
</xsd:element>

<xsd:complexType name="artifactType">
    <xsd:annotation>
        <xsd:documentation>An artifact represents a high-level object
in the repository. It can be a software package, a document,
or any other object to be stored. More specific types of
artifacts</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
        <xsd:element name="metadata">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="supportedPlatform"
minOccurs="0" maxOccurs="unbounded" type="platformType"/>
                    <xsd:element name="licence"
type="licenceType"/>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:attribute name="id" type="xsd:ID"/>
            <xsd:attribute name="packageName" type="xsd:string"/>
            <xsd:attribute name="projectId" type="xsd:ID"/>
            <xsd:attribute name="type" type="xsd:string"/>
            <xsd:attribute name="createDate" type="xsd:dateTime"/>
            <xsd:attribute name="lastModifiedDate"
type="xsd:dateTime"/>
        </xsd:element>
        <xsd:element name="location">
            <xsd:attribute name="id" type="xsd:ID"/>
            <xsd:attribute name="repositoryURL" type="xsd:string"/>
            <xsd:attribute name="path" type="xsd:string"/>
            <xsd:attribute name="protocol" type="xsd:string"/>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="platformType">
    <xsd:annotation>
        <xsd:documentation>A platform is a combination of OS
distribution, architecture and compilers that uniquely
identifies a working environment where a package can be
installed.</xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="id" type="xsd:ID"/>
    <xsd:attribute name="name" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="licenceType">
    <xsd:annotation>

```

```

        <xsd:documentation>The licence under which an artifact is
released. This can be the name of OSI-approved licence
        or a pointer to a licence file</xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="id" type="xsd:ID"/>
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="URI" type="xsd:string"/>
</xsd:complexType>

</xs:schema>

```

8.3 Implementation

Although the goal of this document is not to present the implementation details of the repository service, it seems appropriate to briefly discuss the main principles on which the decisions about possible implementations will be taken and some direction that has already been evaluated by the SA1 activity members.

Given the time constraints of the project and the fact that the focus of SA1 is not necessarily to implement from scratch a repository tool, some investigation has started to identify existing tools that can satisfy the requirements of SA1.

Among the tools investigated, the Fedora digital library framework [12] seems to possess most of the required features. In particular it is of particular interest its compliance to a number of standards (like OAIS and METS), its flexibility and the level of customization it provides, including extendable metadata and a customizable web front-end.

In addition, the ETICS project with which OMII-Europe collaborates to provide certification of middleware components is also planning to adopt Fedora as the internal repository mechanism, which would greatly facilitate the exchange of objects and metadata.

More details on the chosen implementation will be given in separate documents to be released during the course of the project.

9 Security: Authentication, Authorization and Accounting

9.1 Authentication

For a web portal, a simple username/password provides a traditional portable mechanism for identifying an individual.

Other mechanisms include the use of a browser provided client side X.509 certificates which may be suitable for some, but not all, communities.

The programmatic API may use a mixture of technologies: a web service interface to access repository control functions (e.g. initiate tests, upload results) and a variety of direct HTTP mechanisms to perform authenticated downloads (e.g. username/password or client based certificate authentication).

9.2 Authorization

Within the repository access to various functions will need to be controlled. Only defined authenticated users should be allowed to upload build artefacts into the repository or to trigger test runs.

9.3 Accounting

Not only should the authenticated use of the repository be used to log activity, but downloads, uploads, builds & tests should be accounted for to provide statistics for analysis.

10 Concluding Statement

In this document we have presented a high-level design description of the SA1 repository including a general database schema as a reference.

This document will be followed by an executive design document which will detail all the design specifications and implementation decisions, any external tools which will be used and the final version of the database schema.

11 References

1. *Open Archives Initiative*, <http://www.openarchives.org>
2. *Archives Initiative Protocol for Metadata Harvesting*, <http://www.openarchives.org/OAI/openarchivesprotocol.html>
3. *OAI for Beginners - The Open Archives Forum online tutorial*, <http://www.oaforum.org/tutorial/>
4. *Reference Model for an Open Archival Information System (OAIS)*, <http://public.ccsds.org/publications/archive/650x0b1.pdf>
5. *The Dublin Core Metadata Initiative*, <http://dublincore.org/>
6. *The Educational Community License*, <http://www.opensource.org/licenses/ec11.php>
7. *The Mellon Fedora Project: Digital Library Architecture Meets XML and Web Services*, Sandra Payette and Thornton Staples, <http://www.fedora.info/documents/ecdl2002final.pdf>
8. *XML Schema Language*, <http://www.w3.org/XML/Schema>
9. *World Wide Web Consortium*, <http://www.w3.org/>
10. *Network Development and MARC Standard Office of the Library of Congress*, <http://lcweb.loc.gov/marc/ndmsso.html>
11. *METS, Metadata Encoding and Transmission Standard*, <http://www.loc.gov/standards/mets/>
12. *Fedora*, <http://www.fedora.info>
13. *ETICS*, <http://www.eu-etics.org>