SEVENTH FRAMEWORK PROGRAMME

CAPACITIES

Research Infrastructures

INFRA-2007-1.2.1 Research Infrastructures

DRIVER II

Grant Agreement 212147

"Digital Repository Infrastructure Vision for European Research II"

Compound Object Model Integration Report

Deliverable Code: D8.4

Document Description

Project	
Title:	DRIVER, Digital Repository Infrastructure Vision for European Research II
Start date:	1st December 2007
Call/Instrument:	INFRA-2007-1.2.1
Grant Agreement:	**212147**

Document	
Deliverable number:	D8.4
Deliverable title:	Complex Object Model Integration Report
Contractual Date of Delivery:	1st of December 2008
Actual Date of Delivery:	1th of December 2008
Editor(s):	CNR
Author(s):	Paolo Manghi
Reviewer(s):	Wolfram Horstmann
Participant(s):	
Workpackage:	WP8
Workpackage title:	Enhancement of Data Layer Services
Workpackage leader:	UNIBI
Workpackage participants:	CNR, ICM, NKUA, SURF, UGOE
Distribution:	Public
Nature:	Deliverable
Version/Revision:	0.2
Draft/Final:	Final
Total number of pages: (including cover)	12
File name:	D8.4.pdf
Key words:	Content Services, MDStore, Index, Information Space

Disclaimer

This document contains description of the DRIVER II project findings, work and products. Certain parts of it might be under partner Intellectual Property Right (IPR) rules so, prior to using its content please contact the consortium head for approval.

In case you believe that this document harms in any way IPR held by you as a person or as a representative of an entity, please do notify us immediately.

The authors of this document have taken any available measure in order for its content to be accurate, consistent and lawful. However, neither the project consortium as a whole nor the individual partners that implicitly or explicitly participated in the creation and publication of this document hold any sort of responsibility that might occur as a result of using its content.

This publication has been produced with the assistance of the European Union. The content of this publication is the sole responsibility of DRIVER consortium and can in no way be taken to reflect the views of the European Union.

The European Union is established in accordance with the Treaty on European Union (Maastricht). There are currently 25 Member States of the Union. It is based on the European Communities and the member states cooperation in the fields of Common Foreign and Security Policy and Justice and Home Affairs. The five main institutions of the European Union are the European Parliament, the Council of Ministers, the European Commission, the Court of Justice and the Court of Auditors. (http://europa.eu.int/)

DRIVER-II is a project funded by the European Union

Table of Contents

Table of Figures

Summary

This deliverable presents how Content Services – a novel service that is capable of managing compound objects – is to be realized in WP8 and how it will impact on the current technical realization of the Information Space, based on MDStore Services and Index Services. This deliverable also lays out the plan how to seamlessly connect to the studies on “enhanced publications” – as specific exemplars of compound objects – that were carried out in WP4.

1 Introduction

The Digital Library-oriented compound object model presented in deliverable D8.1 (as described in [1]) will be supported by special DRIVER services, called Content Services. As such, Content Services will give tailored support for storing and searching compound objects to digital library applications. In this scenario, following a well-established design methodology in database applications, application designers will first define the structure of the data they need (i.e. document model) and then rely on a data management system (i.e. Content Service) to allocate the space needed to efficiently store and retrieve instances of such data. On top of the data management systems, developers will build applications (e.g. user interfaces), enabling users to operate over their data.

Content Services can store and search objects of various typologies, including relationships between them. In particular they will provide optimized storage and efficient search mechanisms for metadata formats. In that sense Content Service functionalities “overlap” with those currently offered by MDStore Services and Index Services, used to support construction of Information Spaces. In this deliverable we shall explain how Content Services may coexist or gradually take over the Information Space tasks of the current Services in the DRIVER infrastructure.

2 Data Layer architecture

The DRIVER Data Layer architecture is described in Figure 1. As described in deliverable D5.1, in the running infrastructure it consists of an arbitrary number of instances of MDStore Services, Index Services, Aggregator Services. In particular, Aggregator Services are used to transfer Dublin Core (DC) metadata records from OAI-PMH repositories into MDStore Services, and then transform the incoming DC records into corresponding DRIVER Metadata Format (DMF) records, to be stored again in MDStore Services. To conclude the bottom-up data-flow, newly generated DMF records are fed to Index Services. The pool of Index Services constitutes the DRIVER Information Space, i.e. a searchable space of uniform metadata records, in this case DMF. Search Services reply queries over the Information Space, received from portals (user interfaces), by forwarding them to the "best" Index Service they discover in the infrastructure – where "best" depends on the indexing strategy adopted by the orchestration framework.

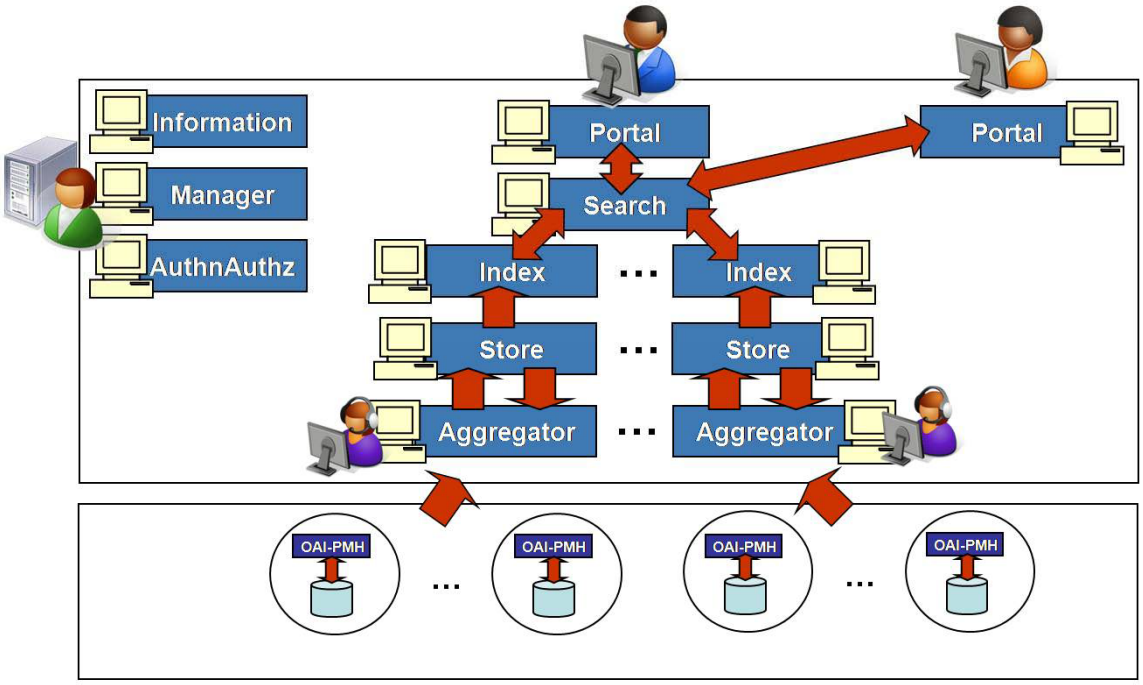

Figure 1 – Data Layer architecture

D5.1 (**Error! Reference source not found.**) illustrates the orchestration policies adopted by the Manager Service to support Aggregator Managers activities at:

1. The application of DC records harvesting;
2. The application of DC-to-DMF transformation mappings;
3. The subsequent indexing of the resulting DMF records.

To this aim, the Manager Service ensures that each OAI-PMH repository registered to the infrastructure has:

1. a number of dedicated DC MDStore Data Structures, for the storage of the harvested records;
2. a number of dedicated DMF MDStore Data Structures for hosting the result of the DC-to-DMF transformations;

3. a number of Index Data Structures, fed and up-to-date with the content of the relative DMF MDStore Data Structures.

Manager Service orchestration is heavily based on the *factory service* pattern, to which both MDStore and Index Services are inspired. The running instances of such services are capable of managing Data Structures. In particular, factory services offer two levels of functionality:

- Data Structure management: creation, delete and update of data structures (from which the adjective “factory”);
- Data Structure interaction: execute an operation over a given data structure, created by the given service.

For example, MDStore Services manage MDStore data structures. Any consuming service authorized to do so, can therefore discover an MDStore Service available and send it a request for the creation of an MDStore Data Structure that matches its metadata needs. Once the MDStore Data structure is created, authorized consuming services can interact with the relative MDStore Service so as to add, remove or retrieve records within the MDStore. Index Services are factory services managing Index Data Structures. Similarly to MDStore Services, Index Services can create or delete Index Data Structures, which are capable of indexing records of a given metadata format. Authorized consuming services can feed records to the Index Data Structures, empty them, or query them, through SRU/SRW interfaces.

3 Data Layer with Content Services

As explained in D8.1, Content Services provide Compound Objects management by offering functionalities for:

- Management of Sets of objects: creation of (initially empty) Sets of objects with a certain structure; e.g. similar to creation of relational tables in Relational DBMSs;
- Interaction with Sets of objects: addition, removal, update and search of such objects; e.g. similar to management of records in Relational DBMSs.

This approach matches in principle the one of factory services, which were indeed inspired by data management systems in the first place: applications declare to the system the kind (type) of the data they want to manage, and the system will provide primitives to efficiently and safely store and retrieve such data.

Content Services support special types of objects: *atoms*, i.e. files of different media, *relations*, i.e. objects connecting two objects of other Sets, and *descriptions*. Description objects are intended to describe other objects in the environment (all those connected to them through one relation object) or instead entities in the outside world. Designers can create sets of description objects customizing their property labels and the relative value domains; e.g. integer, real, Boolean, restricted customized "vocabularies", i.e. sets of values, or record themselves, i.e. sets of lablel-domain pairs (see D8.1 for details).

In fact, descriptions match the notion of metadata format records currently supported by DRIVER MDStore Data Structures. Consuming services, e.g. Manager Services, can contact Content Services and send them a request for the creation of a new Set of type description capable of storing objects corresponding to the metadata records they require. Such sets, unlike MDStore Data Structures, can also provide efficient access to the objects therein, through integrated index data structures. Informally, a Content Service Set is therefore both an MDStore and an Index Data Structure for the records, i.e. description objects, it contains. Indexing is optional, and in that case, the two notions of description set and MDStore Data Structure coincide.

Such observation implicitly suggests how Content Services may replace MDStores and Index Services in the infrastructure activities and in the orchestration part, but nothing prevents their coexistence in the same infrastructure to serve different applications.

3.1 Content Services and enhanced publications

As described in D8.1, Content Services are flexible enough to represent and store compound objects matching any community document model. Application designers will first have to define the Sets capturing the compound objects they need and then realize the applications managing such objects.

DRIVER-II will experiment on "enhanced publications", specific exemplars of compound objects that are capable of representing publications together with a number of related data sets and referred publications [3]. Enhanced publications will be represented by specific Sets in Content Services. WP4 has delivered a description of a document model [4] that can be used for the DRIVER infrastructure according the data model [1]. A dedicated meeting was held on 5-NOV-2008 in Utrecht, the Netherlands, in order to make sure that the studies of WP4 are well connected to WP8 and results (M4.1 and M4.2) from WP4 will be usable for technical research and deployment in 2009. In WP9, NKUA will be responsible of the realization of the applications handling enhanced publications. End-users will be able

to access a space of publication and data set objects and form enhanced publications by declaring the proper relationships between such objects. The initial set of publication objects is taken from the DRIVER Information Space. Accordingly, publications into DMF MDStore or DMF Index Data Structures will have to be imported into the corresponding Content Service Sets, so as to initiate the creation of the graph of enhanced publications. However, this operation will not coincide with the migration from the old Information Space solution to the new one. In DRIVER-II, the DRIVER Information Space will survive based on its old architectural scaffolding.

4 References

[1] *Typed Compound Objects Models for Digital Library Repository Systems.* Leonardo Candela, Donatella Castelli, Paolo Manghi, Marko Mikulicic, Pasquale Pagano. ISTI Technical Report 2008-TR-023

[2] *D5.1 Information Space Report.* DRIVER-II deliverable.

[3] *D4.1 Report on Enhanced Publications state-of-the-art*, DRIVER-II deliverable

[4] *D4.2 Report on Object Models and Functionalities,* DRIVER-II deliverable