

ECP-2006-DILI-510001

NEEO

1st report on the SOA architecture design

Deliverable number	<i>D5.1</i>
Dissemination level	<i>Restricted</i>
Delivery date	<i>10 January 2008</i>
Status	<i>Final</i>
Author(s)	<i>Thomas Place</i>



eContentplus

This project is funded under the *eContentplus* programme¹,
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.

¹ OJ L 79, 24.3.2005, p. 1.

1. Introduction

SOA stands for Services-Oriented Architecture. In a services-oriented architecture there are distributed services that communicate with each other. The communication medium is the Web; the messages that are exchanged between the services make use of the transport mechanism of the Web. The most used underlying protocol is *HTTP*. In the communication between services, there are service requests and service responses. Service requests can be URLs using *HTTP Get* or XML documents using *HTTP Post*. Service responses are always XML documents.

In this deliverable the service components are identified that make part of the NEEO domain. In addition the communication protocols are described that are used for exchanging information between the components.

The NEEO domain consists of

1. repositories with metadata and object files
2. the NEEO Gateway including a harvester, a crawler and a search engine for collecting and indexing information from the repositories
3. the NEEO metadata enrichment server for enhancing the metadata with JEL codes and reference lists
4. the NEEO publication list generator
5. the NEEO portal
6. the NEEO portlet

2. The repositories

The NEEO content is stored as metadata records and object files in repositories. There are three types of repositories:

1. the Institutional Repositories of the NEEO partners. The metadata records can be harvested as *DIDL* documents with *OAI-PMH*. The object files can be accessed via *HTTP*. A third source is the log information about the number of downloads of the object files. This information can also be harvested using *OAI-PMH*.
2. the RePEc archives. There is a central registry of all RePEc archives that can be harvested with *FTP* or *HTTP*. The metadata are in a non-XML format called *ReDIF*. There is, however, a central RePEc repository that can be harvested with *OAI-PMH*. The harvested metadata are in *AMF* (Academic Metadata Format). In NEEO, the central repository of RePEc will probably be used for harvesting the RePEc content.
3. non-NEEO repositories (possibly including web sites). It is not yet decided whether a selection of these repositories will be harvested. In the following, this category will not be taken into account.

3. The NEEO Gateway

At the core of the Gateway is the metadata store that is filled with the *DIDL* documents that are harvested from the repositories. The NEEO harvester uses *OAI-PMH*. The RePEc metadata in *AMF* will be converted into *MODS*, the preferred format for descriptive metadata, and in a next step, the *MODS* records will be packaged in *DIDL* documents.

The download information will also be harvested by the NEEO harvester. This information will be added as a *Digital Item* to the relevant *DIDL* document in the central store.

The metadata in the central store are indexed by the NEEO search engine. The search service is available via *SRU*. In the *extraResponseData* of the *SRU* response facet information of the result set will be stored. *SRU* is the base layer in a protocol stack that will support the following protocols for accessing the central metadata store of the NEEO Gateway:

- *SRU*
- *OAI-PMH*
- *RePEc*
- *RSS*
- *Atom*

The last component of the NEEO Gateway is the crawler that fetches the object files with full text by using the URLs that are made available by the harvested metadata. After fetching the object files from the repositories they are indexed by the search engine of the Gateway. In the case that a fetched object file is a scanned document, *OCR* is first applied to the document before handing over to the search engine.

The crawler can also be used for crawling web sites.

i. Metadata enrichment server

This server searches the metadata store of the Gateway for records that can be enriched by automatically generated JEL codes and by automatically extracted reference lists. The *SRU* interface of the Gateway is used to search for the records that need to be enriched. The URLs of object files are retrieved from the metadata records in the central store and are used to fetch the files. After analysing the object files, the generated JEL codes and the extracted reference lists are added to a repository that can be harvested via *OAI-PMH* by the harvester of the Gateway. The JEL codes and the reference lists are added as new *Digital Items* to the corresponding *DIDL* documents in the metadata store of the Gateway.

ii. Publication list generator

This server generates publication lists by searching the metadata store of the Gateway with digital author identifiers as search keys. The search protocol that will be used is *SRU*.

iii. Portal

Another service that makes use of the *SRU* interface of the NEEO Gateway is the portal that allows end users to search in the metadata and in the full text publications. The portal will display the search results together with facet information that is also transported in the *SRU* responses. *OpenURLs* will be included in the display of the records.

A component of the portal will be an *Ajax* (Asynchronous JavaScript And Xml) server with which the browser of the user can interact to ask for information from external web services.

This information is integrated in the display of the search results. The *Ajax* server is configurable to support all kind of web services.

The *Ajax* server will process the requests that are generated by *Ajax* functions in the JavaScript that is incorporated in the HTML pages which are sent by the Portal to the browser. After receiving a request, the *Ajax* server can make use of one or more web services for composing a response that is sent back to the browser. This mechanism makes it possible that a browser asks in an asynchronous way for extra data (coming from external web services) without interfering with the display of the HTML page. When the extra data is available it will be added to the page that is already displayed to the user. This will also allow for adding new information to an existing page as a reaction to an action of the user; there is no need for the server to send a new HTML page.

iv. Portlet

For integration in other portals, a portlet will be developed following the *JSR 168* standard.

