D1.3 OpenScout Web Portal

ECP 2008 EDU 428016

OpenScout

D1.3 OpenScout Web Portal

Deliverable number	*D1.3*
Dissemination level	*Public*
Delivery date	*28 February 2010*
Status	*Final*
Author(s)	*Vassilis Pitsilis (NCSR), Aris Vidalis (NCSR), Fan Deng, Ricardo Kawase (LUH), Philipp Holtkamp, Henri Pirkkalainen, Jan M. Pawlowski (JYU), Marco Kalz (OUNL), Michele Dicerto (Giuntilabs), Alexander Mikroyannidis (OUUK), Elizabetta Parodi (Giuntilabs), Uta Schwertel (IMC), Daiva Vitkutė-Adžgauskienė (VMU)*

e*Content*plus

This project is funded under the *e*Content*plus* 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.

Table of Contents

1 Introduction

OpenScout aims at providing education services in the internet that enable users to easily find, access, use and exchange open content for management education and training. To make accessible content of different types from many disciplines of management science OpenScout inter-connects a large pool of different distributed open content repositories to form a federated content base on which OpenScout's web services operate. Access to OpenScout's services will be provided through a web portal to interested users. This document gives a basic description of the OpenScout web portal.

The OpenScout work package 1 (WP1) on Content Federation and Content Maintenance Infrastructure implements the basis for providing the OpenScout services. The activities and results of WP1 are the basis for the OpenScout project services provisioning. An integrated application profile is under development to include all different application profiles of the content repositories. Access to management related content through a federated infrastructure, on which value added services will be later integrated, is being analyzed and developed throughout this work package. Besides all the above mentioned development within this WP, a presentation layer for the end users is also being designed and applied. This presentation layer is the web portal which will provide access to OpenScout's project services to all interested users. Federated content, professional and open, and tools providing access to and allowing adaptation of this content will be presented through this web portal. In the following sections, a description of the total architecture and the integrated components is provided and accordingly an analysis of the intended technologies to be applied for the portal deployment are presented.

2 Architecture

The web portal is a single access point where users can use the OpenScout services and tools. The initial architecture is shown in the following diagram. The architecture might change during the development and evaluation phase according to specific requirements.

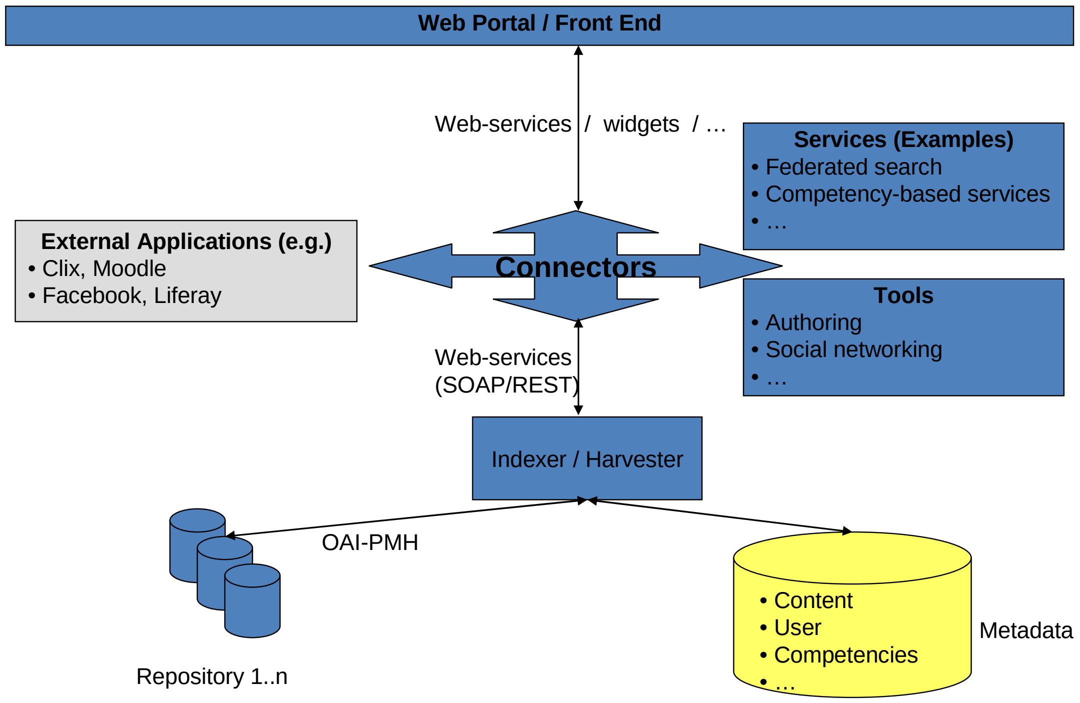


Figure 1: Architecture of OpenScoutWeb Portal

The bottom of the diagram shows the federated content infrastructure. The ***harvester*** collects metadata information from the distributed content ***repositories*** via OAI-PMH[2] protocol and LOM[3] standard, and stores them in a centralized metadata database. The ***indexer*** provides a fast access to the ***metadata*** to the ***connectors***, where other components can retrieve metadata via different technologies such as web services and widgets. Since a related project MACE has implemented a similar indexer and harvester, the work of trying to reuse part of the MACE project (mainly the metadata harvesting part) is in progress.

The diagram represents ***example services and tools*** of OpenScout which can be either accessed by internal/external components or directly by end users. For example, through connectors other components can search the federated infrastructure from the metadata repository and return the result set to the triggering components. Also, end-users can search the federated infrastructure (metadata repository) and obtain a list of content descriptions via ***the web portal***. To integrate other components into the web portal, different technologies are needed depending on the requirements of the components. For example, some ***external applications*** like Facebook provide their own application programming interfaces (APIs) which enable an easy integration into the web portal. Other tools and services may need

[2] OAI-PMH: Open Archives Initiative – Protocol for Metadata Harvesting (http://www.openarchives.org/)

[3] LOM: ***Learning Object Metadata*** is a data model, usually encoded in XML, used to describe a learning object and similar digital resources used to support learning. (From Wikipeida)

specific techniques like widgets, applets and SOAP[4]. More technological details can be found in Chapter 4.

In the next chapter, we will discuss different OpenScout components and the requirements needed to be integrated into the web portal.

3 Components

3.1 Content access

The content access component provides services to both end-users and developers of other components to access the underlying integrated content repositories via the OpenScout federated infrastructure. To further describe this component, we give explanations of the terms we used.

- **End-users:** This group of users refer to educators (e.g. business school instructors), learners (e.g. SME managers), content creators (e.g. e-Learning content producing companies)
- **Developers:** In addition to end-users, the federated content can also be accessed by other components in OpenScout, which are upper layers of the content access component. For example, WP5 needs to access the content repositories to build the mashups. Therefore, Content Access needs to provide services to the developers of other OpenScout components or even external components so that the integrated content can be easily accessed from a unique interface easily.

3.1.1 Content access for end-users

For end-users, the content access component provides initially four types of access methods: keyword-based search, category-based search, content recommendation and related content linkage.

1. Keyword-based search
    a. Quick search box

       This type of content access provides a standard quick search box similar to the search in major Web search engines. This type of content access is quick but not quite targeted. Based on the query, intelligent algorithms will be used to render the result set: all contents whose metadata contains the given keyword will be retrieved.

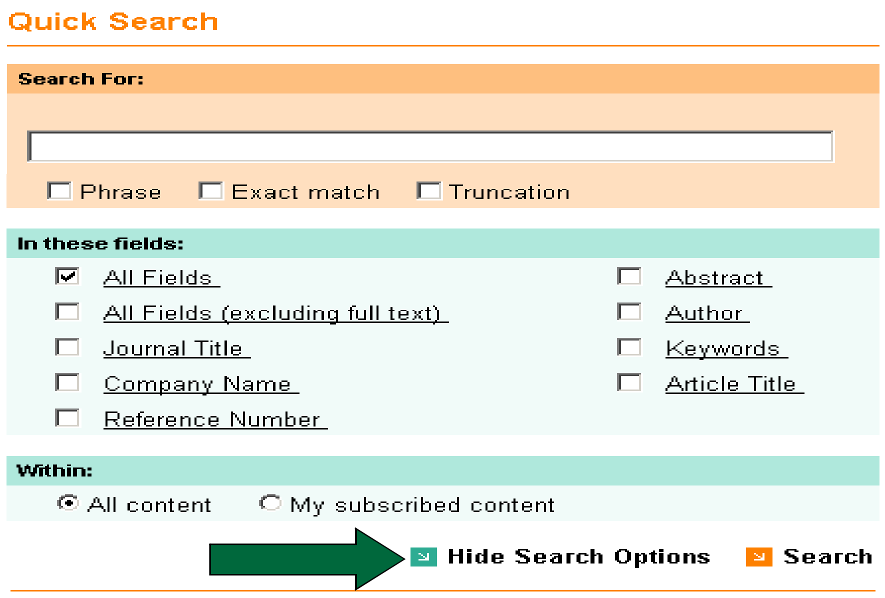
    b. Advanced search box

[4] SOAP: ***Simple Object Access Protocol***, is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks.

This type of content access provides sophisticated search functionality. It is used often when the quick search functionality fails. Detailed search criteria can be specified so that the search can be more accurate and targeted. The trade-off is that the end-users need to spend time on giving the input to the search box; it also has the risk that search conditions are over-specified and no results are found.

c. Search example

To visualize the keyword-based search, we give a graphic example[5].



Quick Search

Search For:

☐ Phrase ☐ Exact match ☐ Truncation

In these fields:

☑ All Fields	☐ Abstract
☐ All Fields (excluding full text)	☐ Author
☐ Journal Title	☐ Keywords
☐ Company Name	☐ Article Title
☐ Reference Number	

Within:

[✓ All content] My subscribed content

Hide Search Options Search

Figure 2: Example for keyword-based search

In this example, end-users can specify search terms in the search box; if no option is checked, the component will continue with a search in the metadata database and retrieve all records matching the given keyword; then using certain ranking algorithms, listing the most promising contents on the top of the result list.

If certain search criteria are specified, the search will be restricted within a subset of the metadata (e.g. Title, Author, Keywords and so on in the figure).

2. Category-based browsing

Different from keyword-based search, category-based browsing provides a hierarchy of the metadata which enables end-user to check sub-sets of contents based on topics.

- Advantage:

[5] From www.emeraldinsight.com

End-users are able to access contents based on repositories, topics, categories, languages and so on. This is particular useful for those users who do not know exactly what to look for. This can happen when the users have limited amount of knowledge about the subject and would like to know more about the topics.

- Disadvantage:

  End-users may have to spend more time to find what they want. This is a problem especially when there are no contents that interest the user, which may significantly reduce the credibility of the system. Another potential problem is that metadata may be difficult to be classified, and it also relates to the domain knowledge of end-users. For beginners, it is possible that they fail to find the desired content due to their wrong understanding of the subject.

3. Content recommendation

Keyword-based search and category-based browsing are two types of content access methods where end-users actively check content repositories. In contrast, content recommendation is another type of content access method where end-users passively accept contents.

- Advantage:

  End-users may find contents that are useful but hard to find via regular keyword based search or category-based browsing. Also, it involves little effort from the end-users side. In this sense, content recommendation is the most efficient way for access content repositories.

- Disadvantage:

  To provide content recommendation service, the component needs to know the profile of end-users, potentially from previous accessing history or end-users' registration information. Furthermore, it is technically challenging to find interesting contents only based on user profiles and content metadata. Active research is being done in the area.

4. Related content linking

Another type of powerful content accessing approach is via related content linkage. The idea is to link all related contents within the content repositories together. The relevance can differ depending on applications. For example, slides discussing similar topics, images with similar titles and videos with similar tags.

Whenever a desired content is found via certain aforementioned accessing methods, more related content can be easily shown to the end-users. With this linkage, end-users can go through the content network efficiently without much difficulty.

- Advantage:

Once a useful content is found, more can be accessed easily without much effort. It provides a unique way to browse the network of useful information and it is usually efficient.

- Disadvantage:

  It may not be easy to define relevance especially if it involves domain knowledge. Also, if the quality of the relevance is not high enough, it may discourage the end-users to use this approach since the links may lead to irrelevant contents. Furthermore, defining relevance itself is a challenging and perhaps subjective task.

In addition to the above mentioned search mechanisms, we will also implement services based on competencies and related concepts. As found in the initial user requirement analysis, user might search rather for problems than for competencies directly – this aspect is described in chapter 3.2.

Based on the above discussion, we will define our end-user services depending on different factors like the content types to be shown. For example, related content linking may be easier to implement compared to linking multimedia contents.

3.1.2 Content access for OpenScout components

In addition to end-users, the content access component also provides services to developers for them to access the underlying contents. To communicate with the federated infrastructure, Web services and Google gadgets are two of the options for connecting the federated contents and other components.

1. Web services

   The content access component integrates distributed content repositories and provides a common interface to other components so that developers of other components have no need to know the details of distributed contents. The content access component harvests metadata from distributed contents and manages the metadata centrally.

   With the format of Web services, other components can assess those metadata following the same standard and interface. Depending on the component functionalities, contents may or may not be forwarded from the distributed repositories to the triggering component.

   - Advantage:

     With Web services, all components can easily access the centralized metadata storage in a standard way. Also, web services are widely used and accepted. Thus, tools and information are freely available on the Internet, which makes the development easier.

   - Disadvantage:

Using HTTP protocol, the communication may not be so efficient for transferring large amount of data.

2. Google gadgets

   The content access component can provides services for developers to access contents. Google gadgets are small tools that can be inserted into web pages to fulfil certain simple functionality. For example, whenever new data become available in the centralized server, the content access component can communicate with other components and inform about the update of the centralized database so that users can monitor the content changes in real-time.

   - Advantage:

     Being small tools, Google gadgets are efficient for small tasks like updating simple content statistics. Also, Google gadgets are easy to be deployed on the upper level components: the developers just need to embed the gadgets into the Web page and not much programming is needed.

   - Disadvantage:

     It is not suitable for heavy tasks such as building complicated social mashups on top of the federated content repositories. Also, from the Content Access side, the workload will be heavier since gadgets are more like and end-user application rather than a programming interface.

The discussion above shows two options for the content access – further access options will be test in the first prototyping phase. In particular through the connector model, further access options are developed.

3.1.3 Integration Requirements

Potentially there are two types of services provided by content access component as described in the previous section: services for end-users and services for other components, which have different requirements to be integrated into the web portal.

1. Requirements for integrating end-user services

   Depending on how closely the services are to be integrated into the OpenScout Portal, the requirement can be different:

   1. Loose integration

      In this case, the Content Access component will provide its own user interface allowing end-users to access the contents. To be integrated into the web portal, it can be as simple as providing links to the Content Access user-interface. In this case, there are almost no requirements on the Web portal side.

   2. Tight integration

In this case, the end-user interface for Content Access is included as part of the web-portal. In fact, the Web portal developers may need close collaboration with the Content Access developers for the technology selection, user-interface and functionality design. This includes a decision on what Web development platform (e.g.Joomla, Drupal, PHP etc.) is to be used, what databases to be used (MySQL, PostgreSQL etc.), where the application server is located and so on. The requirements will be more precisely specified after further discussion with the web portal developers.

2. Requirements for integrating component services

   The components services from Content Access provide services to other components inside or outside OpenScout, e.g. Competency-based search, Mash-up services, widgets in end-users' personal portal etc. We plan to realize this service using a Service Oriented Approach (SOA) approach and provide web services. To integrate the services directly into the web portal and also with other components, via the connector we foresee the following requirement:

   1. The client (either the Web-portal or some other components) needs to support Web-services, SOAP standard and XML messages.
   2. The client needs to develop its own application calling the web-services.
   3. The client needs to understand and agree with the services that Content Access provides; communication between services and content providers is needed for this purpose.

3.2 Competence services

In compliance with the IEEE Standard for Learning Object Metadata (LOM)[6], we propose to store the metadata related to competencies for each learning object (LO) in the classification section of the LOM (example given). Each LO can have any number of competencies associated to it. For each competency of a LO, the minimum and maximum proficiency scale values, defined according to the European Qualification Framework (EQF), are included.

```
<classification>
      <purpose>
            <source>LOMv1.0</source>
            <value>competency</value>
      </purpose>
      <taxonPath>
            <source>
                  <string language="en">Classification System</string>
            </source>
```

[6] IEEE Standard for Learning Object Metadata 1484.12.1-2002 (http://ltsc.ieee.org/wg12/par1484-12-1.html)

```
            <taxon>
                  <id>Domain Identifier</id>
                  <entry>
                        <string language="en">Domain Title</string>
                  </entry>
            </taxon>
            <taxon>
                  <id>Competence Identifier</id>
                  <entry>
                        <string language="en">Competence Title</string>
                  </entry>
                  <mineqf>Minimum EQF Level</mineqf>
                  <maxeqf>Maximum EQF Level</maxeqf>
            </taxon>
      </taxonPath>
</classification>
```

To collect, catalogue, manage, and maintain the competence metadata a toolset has been created. The core of the toolset is the competence catalogue. It contains competence domains and their related competencies as well as external resources, experts, and a proficiency scale description related to these competencies. On top of this catalogue different applications and widgets can be used for displaying, dynamic updating, and editing competence metadata as well as for the administration of the competence catalogue.

The competence catalogue is an object-oriented application written in Java and is able to output the data in several output formats such as XML and JSON. It provides functionality to administrate domains and competences.

The competence services are an abstraction layer to the competence catalogue to provide access and administer it. We provide initially two web services, one for accessing the data (Competence Service) and one for the administration of the data (Competence Admin Service).

- *Competence Service:* Introduction to the service and its functionality as well as the relationship to the available clients (e.g. Competence Widget).

The Competence Service can be accessed using a SOAP API and provides the following methods:

Method	**Description**
getVersion	Get the current version of this webservice
getStatus	Get the current status of this webservice
getDomains	Get a list of all the available domains
getCompetence	Get one competence from the database and all the related experts and resources
getBasicCompetence	Get a basic competence from the system
getCompetenceListFromDomain	Get a basic list of competences from the databasefor a specific domain

getResourcesFromCompetenceId	Get all the resources for a certain competence
getExpertsFromCompetenceId	Get all the experts for a certain competence
getProficiencyScalesForCompetence	Get a proficiency scale for a competence from the database
getCompetenceList	Get the full list of competences from the database
getBasicCompetenceList	Get a basic list of competences from the database

Table 1: Competence Service methods

- *Competence Admin Service:* Introduction to the service and its functionality as well as the relationship to the available clients (e.g. Competence Admin).

The Competence Admin Service can also be accessed using a SOAP API and provides the following methods:

Method	**Description**
getVersion	Get the current version of this webservice
getStatus	Get the current status of this webservice
createDomain	Create a domain
updateDomain	Update a domain
deleteDomain	Delete a domain
createCompetence	Create a new competence
updateCompetence	Update the competence
deleteCompetence	Delete a competence
createResource	Add a resource
updateResource	Update a resource
deleteResource	Delete a resource
createExpert	Add an expert
updateExpert	Update an expert
deleteExpert	Delete an expert
updateProficiencyScale	Create/Update/Delete proficiency scale

Table 2 Competence Admin Service

3.3 Tool Library

According to the DoW, a significant aspect of the OpenScout service portfolio regards tools for improvement and republishing of contents and materials. In order to achieve this, access to authoring technologies needs to be given to stakeholders so that they can add content, repurpose content or re-aggregate content to new learning objects. These authoring technologies range from easy to use tools to more elaborate ones, depending of the format of the original content. Within the consortium, many tools are available, so that the most content formats can be supported without new technology development. These tools include services such as collaborative authoring, blogging, experience sharing, re-using and integrating (cross-border) learning scenarios, using accessibility tools, course creation and management, re-authoring, podcasting, metadata tagging, annotation services, etc.

We perceive the OpenScout Tool Library as an ecosystem of *people*, *stories*, and *resources* (Figure 3). The purpose of this ecosystem is to bring together people that are developing or using learning resources and provide them with the ability to share their stories and resources. These people come from diverse backgrounds and are involved in various stages of the lifecycle of learning resources. We have identified four major stakeholder clusters: content developers, educators, content providers and brokers, collaborators and social learners. Their stories include completed or running case studies and learning scenarios, their experiences with learning resources, as well as their future expectations from them. Finally the learning resources involved are either learning tools or content, such as Open Educational Resources (OER). The following sections describe in more detail the people, stories and resources of the OpenScout Tool Library.

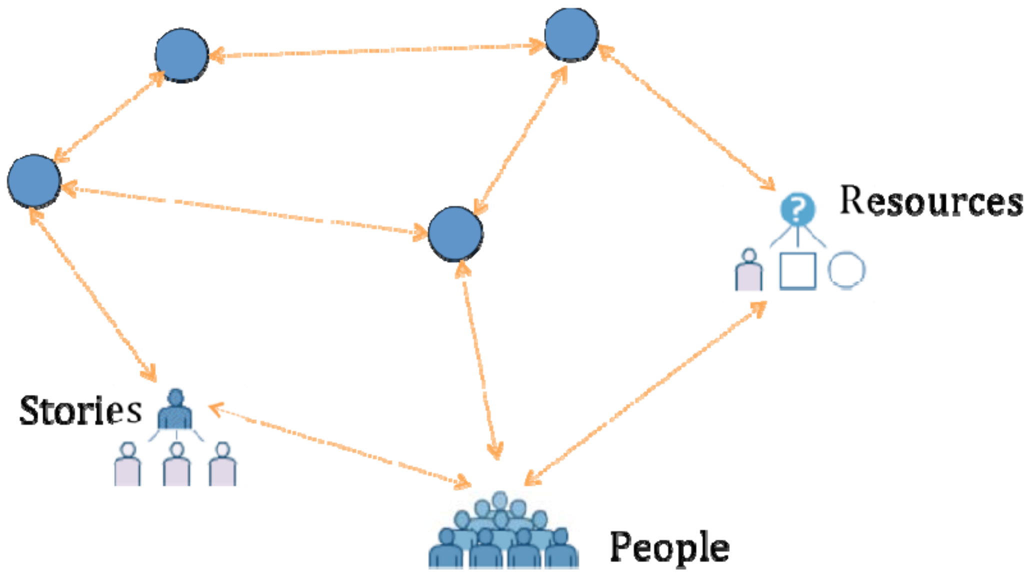


Figure 3: The OpenScout Tool Library connects people with stories and resources

3.3.1 People

The following clusters of people are involved in the OpenScout Tool Library, in accordance with D7.1.1 “First Dissemination Plan”:

:

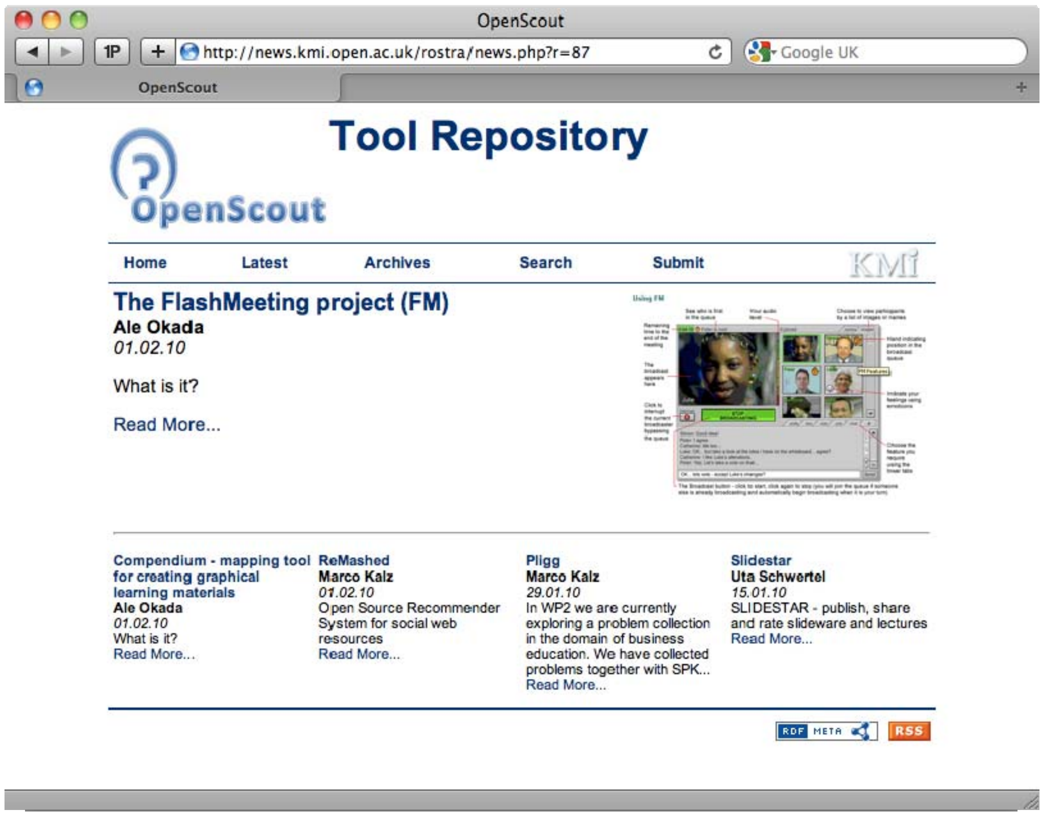
1. **Content providers and brokers**: This cluster includes Professors, Lecturers, Learning Designers and team leaders who aim to develop new courses, workshops or training sessions and authoring new learning materials. They are interested in selecting and integrating new media and content from different repositories for new learning scenarios. Their needs might also be services and tools for analyzing existing content for repurposing and re-authoring tools for content manipulation.
2. **Educators**: This cluster involves high schools, academic higher education institutions and universities as content users for education, plus Instructors (Professors and Teachers) & Trainers. The stakeholders of this cluster are interested in integrated services and recommendation mechanisms for rating and evaluating content.
3. **Collaborators**: This cluster includes Students/Learners within education that are interested in establishing partnerships and co-authoring learning resources internationally with other partners, developing research and expanding their existing communities. They might be interested in publishing remixed content, including multicultural resources in different languages for large scale and also contribute to the open content movement.
4. **Social Learners**: This cluster is interested in tools and social services (social metadata information).These tools and social services provide mechanisms for personal networks with users from different learning environments who may have similar learning interests.

3.3.2 Stories and Resources

Open Educational Resources (OER) are of particular importance for the OpenScout Tool Library. OER can be described as "teaching, learning and research resources that reside in the public domain or have been released under an intellectual property license that permits their free use or repurposing by others depending on which Creative Commons license is used"[7]. OER are freely available on the web and can be accessed through common web sites, Virtual Learning Environments (VLEs), or Personal Learning Environments (PLEs). They can be used, edited and shared by any interested party, such as learners, teachers, institutions, and learning communities.

In order to facilitate the collection of resources and stories about them, we have created the web-based repository shown in Figure 4, which is publicly accessible at: http://news.kmi.open.ac.uk/rostra/news.php?r=87. In this repository, users submit learning resources, describe their functionality and the settings the tools have been used at. The repository supports both textual and multimedia descriptions of resources and stories, including hyperlinks, screenshots, presentations, and videos. Users can provide feedback on existing resources and stories in the repository by rating them within a 5-star scale and commenting on them.

[7] Atkins, D. E., Brown, J. S. & Hammond, A. L. (2007) A Review of the Open Educational Resources (OER) Movement: Achievements, Challenges, and New Opportunities. The William and Flora Hewlett Foundation.



OpenScout

http://news.kmi.open.ac.uk/rostra/news.php?r=87

Google UK

OpenScout

Tool Repository

OpenScout

Home Latest Archives Search Submit KMi

The FlashMeeting project (FM)
Ale Okada
01.02.10

What is it?

Read More...

Compendium - mapping tool for creating graphical learning materials
Ale Okada
01.02.10
What is it?
Read More...

ReMashed
Marco Kalz
01.02.10
Open Source Recommender System for social web resources
Read More...

Pligg
Marco Kalz
29.01.10
In WP2 we are currently exploring a problem collection in the domain of business education. We have collected problems together with SPK...
Read More...

Slidestar
Uta Schwertel
15.01.10
SLIDESTAR - publish, share and rate slideware and lectures
Read More...

RDF META RSS

Figure 4: A web-based repository for collecting stories and resources

The following paragraphs present two cases of resources and their related stories, derived and adapted from the repository.

- **Resource:** FlashMeeting (http://flashmeeting.open.ac.uk/) is a browser-based videoconferencing tool. It provides a user-friendly interface for simultaneous video/audio/text conferencing, plus some more advanced features, including the ability to work collaboratively through a whiteboard, exchange files, replay meetings, view statistical analyses of meetings, create meeting minutes, as well as create and manage groups and contacts.
- **Story**: Figure 5 shows an OER in FlashMeeting, concerning learning styles in a digital scenario in the Portuguese language. The OER was developed by a member of the Colearn Community within the OpenLearn project and lecturer at a university in Portugal. She was invited by a lecturer from a Brazilian university to present and discuss the topic through FlashMeeting with a group of graduate students interested in Knowledge Technologies for professional training. The recording of this presentation and discussion is accessible here: http://fm-openlearn.open.ac.uk/fm/982a5a-8691.

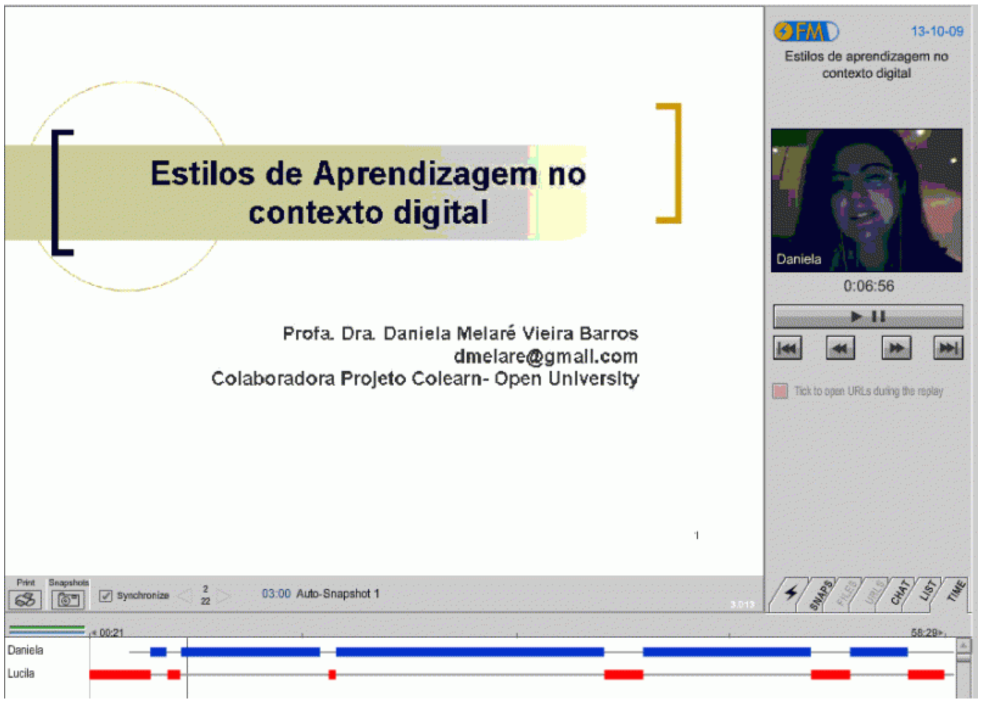
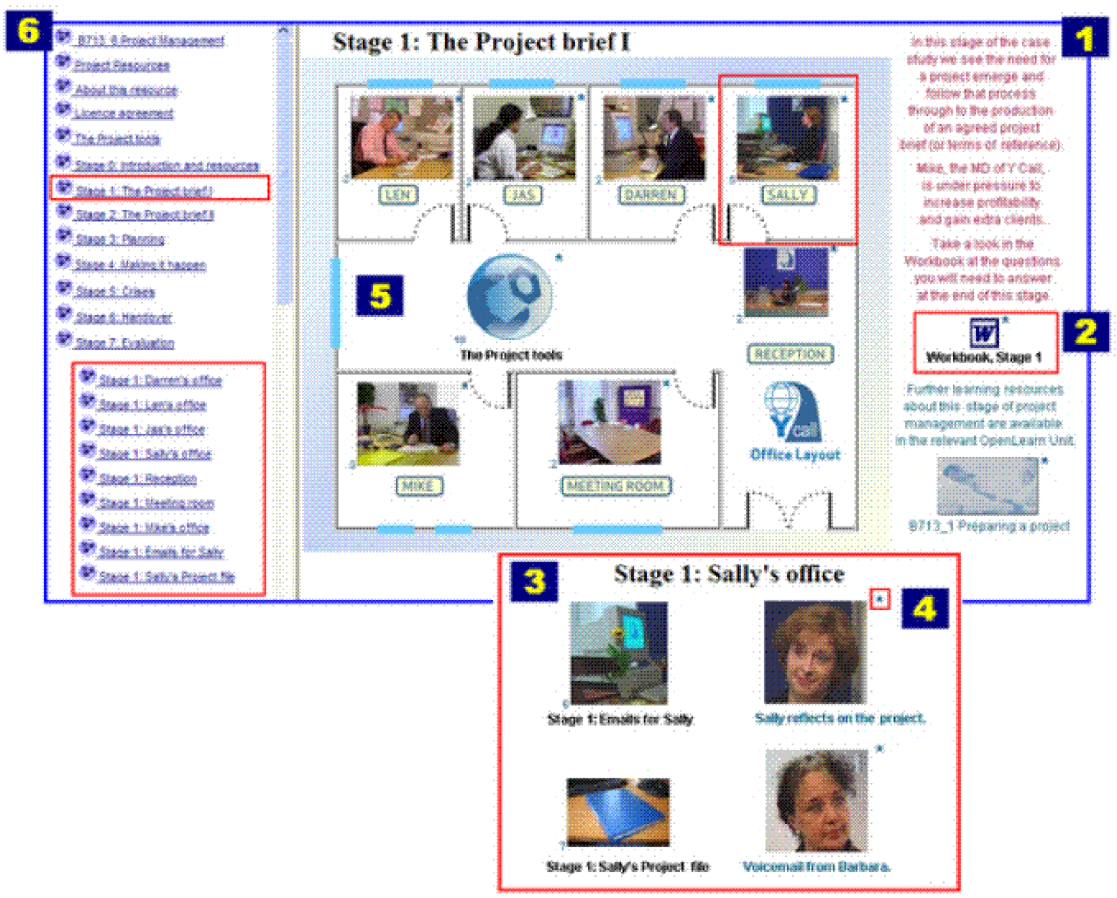

FM
13-10-09
Estilos de aprendizagem no contexto digital
Estilos de Aprendizagem no contexto digital
Profa. Dra. Daniela Melaré Vieira Barros
dmelare@gmail.com
Colaboradora Projeto Colearn- Open University
Daniela
0:06:56
Tick to open URLs during the replay
Print
Snapshots
Synchronize
03:00 Auto-Snapshot 1
SNAPS
FILES
URLS
CHAT
LIST
TIME
00:21
58:29
Daniela
Lucila

Figure 5: Learning styles OER in FlashMeeting

- **Resource:** Compendium (http://compendium.open.ac.uk/) is a tool for building maps of information, ideas and arguments. It has been designed as a sensemaking tool to link, interpret and annotate resources on the web, with a default visual language of icons and connections designed to provoke reflection on the differences between questions, ideas, and challenging or supporting evidence and arguments. Compendium has been employed in a wide variety of domains and case studies, such as mapping literature surveys, mapping learning design patterns, transcribing argumentation and group memory from a meeting, and more.
- **Story:** The Project Management OER shown in Figure 6 was created with Compendium. It comprises a set of maps that represent an integrated overview (Figure 6, label 5) of seven offices of the fictitious company called Y Call. The main goal of this OER is to develop a business project using the Y Call resources that are made explicit in the first map's description (Figure 6, label 1). Each office is, in fact, a map that presents a variety of information (Figure 6, label 3): text, images, audio and video files. In addition, learning activities (Figure 6, label 2) can be accessed in the associated Word-based workbooks. Y Call's office maps offer learners and educators an opportunity to experience first-hand the issues that can arise from project management. They are encouraged to work through the various clues (Figure 6, label 4) useful for developing their business project by following a recommended trail (Figure 6, label 6) that has seven pre-defined stages.



In this stage of the case study we see the need for a project emerge and follow that process through to the production of an agreed project brief (or terms of reference).

Mike, the MD of Y Call, is under pressure to increase profitability and gain extra clients.

Take a look in the Workbook at the questions you will need to answer at the end of this stage.

Further learning resources about this stage of project management are available in the relevant OpenLearn Unit.

Figure 6: Project Management OER in Compendium

3.3.3 Integration requirements

The integration of the Tool Library with the web portal is dictated by a set of requirements for sharing data between the Tool Library and other components of the portal. The purpose of these requirements is to establish a common infrastructure for storing data that will be accessible by all components of the web portal. In particular, the data that the Tool Library will share with the portal components are the following:

- **Common authentication**: The user of the Tool Library should have a single authentication point with the rest of the portal. In this way, the user will not be required to have different accounts for different services offered by the web portal, but only a central one that will automatically authenticate her to all individual components of the portal.
- **Shared user profile**s: The data kept about the profile of the Tool Library user should be shared across all components of the web portal. For example, the Tool Library keeps information regarding which resources someone uses and in what context (stories). This information can be useful for the Content Access component and in particular for offering content recommendation services to the user, based on the resources that she already uses. In addition, user profile information from other

components can facilitate building the user's connections with resources, stories, and other people in the Tool Library.

- **Shared resources**: The resources of the Tool Library will also need to be shared with other web portal components. In this way, related content linking performed by the Content Access component can be facilitated with information derived from the Tool Library about related resources.
- **Shared stories**: The stories (case studies and scenarios) of the Tool Library should be available to be reused within other components of the web portal.

3.3.4 Next steps

The next steps towards the design and implementation of the OpenScout Tool Library concern the refinement of its current specifications. This will be accomplished via gathering additional requirements from the OpenScout Future scenarios that are currently under development. The input derived from these scenarios will mainly address specific needs of stakeholder clusters, relevant functionalities of the tools (which tools are necessary and needed) derived from the users' needs as well as potential integration methodologies, e.g. in the form of workflows of tools. In addition, our web-based repository will be enriched with further submissions of stories and resources, so that a critical mass of data is brought together for the construction of the Tool Library.

3.4 Connector / Social Networks

Mash-up technologies enable the integration of OpenScout services (search for open management content, upload content,…) into other applications, e.g. Social networks, personal portals or Learning Management Systems. This requires the generation of embed codes/widget codes, which are then integrated into the HTML-code of the respective website or application. The functionalities are thus directly brought into the familiar target applications of the users. The content itself will reside in the respective original repository it can however be accessed from all applications where the modular code is embedded.

The OpenScout portal architecture should therefore achieve as one major goal the possibility of reusing and combining (mashing up) the services that OpenScout delivers

Different OpenScout services should be exposed to the users in several ways

- Call services directly on the OpenScout portal with an OpenScout specific user interface, the results are rendered within the OpenScout system. (cf. section **Error! Reference source not found.**).
- Make the portal functionality and services accessible from other systems and platforms (e.g. access from LCMSs like CLIX, or external social network platforms like iGoogle, Facebook or Netvibes). In this case user interaction with OpenScout is taking place via the external system. (cf. section **Error! Reference source not found.**)

OpenScout work package 5 connects OpenScout services with external applications, LCMS and other external applications like Social applications Network. Hence, the focus will be how to realize the access of the services from external systems. WP5 presupposes the existence of a federated content base that exposes its metadata in a standard format. (cf. chapter **Error! Reference source not found.**).

To be able to connect the OpenScout services to external applications the OpenScout architecture needs to offer interfaces to its services on the internet. In order for the services to read information from external users and to transfer information to external users the OpenScout system needs to be based on a service-oriented architecture (SOA), which conforms to existing standards for internet based services (Web services).

Requirements for the user-interface of the external target application

In order to make it possible for end-users to access the OpenScout services from external systems, each service needs to be integrated into a graphical user interface (GUI). Since the OpenScout Services will be offered through standardized interfaces the integration of the services can in principle be done using different graphical user interfaces. We consider it however a key requirement for the OpenScout system architecture that the services have a modular character so that they can be integrated – without much additional development effort – in diverse target applications and platforms. Additionally, it should be possible that a user of the target applications is able to subscribe to the services.

Widget Technology

There is a wide variety of APIs and sharing mechanisms to make OpenScout portal functionality accessible from external applications. The approaches are generally based on a widget like approach, which is used to implement the export filters [**Error! Reference source not found.**].

In order to guarantee the above described modularity and flexibility the OpenScout system should offer the possibility to deliver so-called widgets to the end-user with which the user can access the Webservice interface of OpenScout. Widgets in this context mean small, client-side applications allowing the access or manipulation of remote data. Widgets can be integrated into existing Web applications or into desktop environments. The target platform into which a widget is integrated is often called a widget engine or a widget container.

Examples for widget environments are among others the Apple Dashboard or the Yahoo widget engine. Examples for Web applications or widget containers are among others iGoogle [**Error! Reference source not found.**], NetVibes [**Error! Reference source not found.**] or Facebook [**Error! Reference source not found.**]. In order to guarantee re-usability and inter-operability of the widgets the implementation of the widgets should conform to existing standards. For the interaction with social network platforms the OpenSocial APIs [**Error! Reference source not found.**] should be used. Applications that use the OpenSocial APIs can be embedded within a social network itself, or access a site’s social data from anywhere on the web.

Using these and other standards for the widget implementation makes it possible that widgets need to be programmed just once and can then be integrated into different platforms, applications or different social networks.

In [**Error! Reference source not found.**] an alternative option to achieve modularity of the services is discussed. This approach provides the ability to build new applications by selecting and combining specific parts of already existent applications, thus emerging new functionalities. Under this approach different alternatives for fragment selection and composition have to be analyzed and a JavaScript library could be developed for making composition of code fragments easier.

OpenScout currently prefers a web widget based approach where the implementation of the widgets uses JavaScript and HTML technologies. HTML offers various tools to create user interfaces while JavaScript is used to make those interfaces dynamic.

Required central functionality of the OpenScout portal beyond widgets

Widgets usually run on the client side, and in most cases define presentation and interface aspects. Widgets do not cover other important issues of the OpenScout envisaged web applications, e.g. user management or data persistence. This has to be managed by the central OpenScout Portal.

Requirements how to offer services for external applications

In order for developers of external applications to integrate the OpenScout widgets into their applications the OpenScout services and tools collections could offer the following possibilities:

OpenScout widgets designated for use in external systems could be organized in a searchable repository. Each widget could be accompanied with information for the developers who integrate the widget into the target system (description of used standards, service description etc.). Testing of the widgets directly on the OpenScout platform should be possible before developers download the widget for their own use. The system could enable download of the widgets and should define appropriate terms and conditions of use.

Regarding the integration of external systems with the OpenScout infrastructure the following (modular) functionalities are expected:

The OpenScout platform could integrate for example the following search services as modular widgets.

Search for Open Content in the federated OpenScout content repository from external systems

Scenario 1: Search from Social Network Application

1. User adds the OpenScout search widget as a social application to his personal portal or social network.
2. User is offered a simple keyword based search or advanced access as described in section **Error! Reference source not found.**
3. Results of the search are rendered in the target application (e.g. the social network or the personal platform) through a widget.

The search widget to be embedded could look be abstracted as follows:

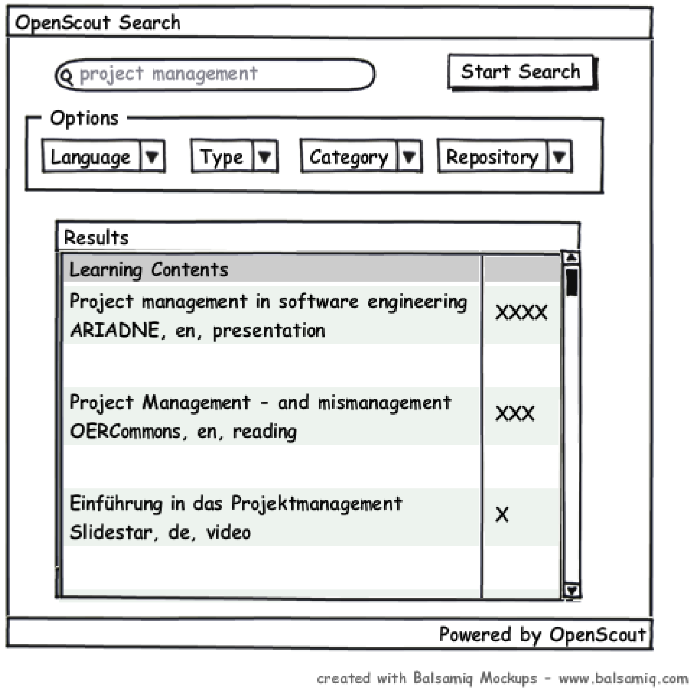
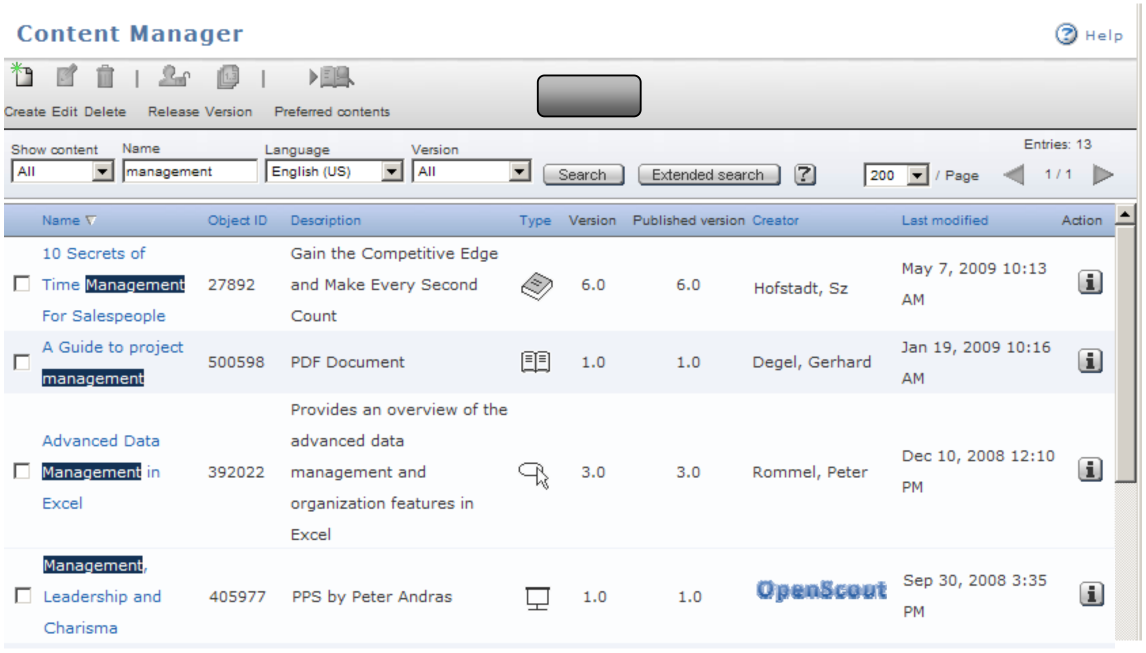

OpenScout Search
project management
Start Search
Options
Language
Type
Category
Repository
Results
Learning Contents
Project management in software engineering
ARIADNE, en, presentation
XXXX
Project Management - and mismanagement
OERCommons, en, reading
XXX
Einführung in das Projektmanagement
Slidestar, de, video
X
Powered by OpenScout
created with Balsamiq Mockups - www.balsamiq.com

Figure 7: OpenScout simple seach widget

Scenario 2: Search from within LCMSs

1. Teacher/instructor searches for suitable learning materials to add to the material for his new project management course.
2. From the search interface of his Learning Management system he is offered the OpenScout search button. The search criteria offered by the standard / extended search of the LCMS are connected to OpenScout search criteria and are transported to the OpenScout portal.
3. The results of the OpenScout search are added within to the search results the LCMS.
4. The teacher/instructor can select suitable learning objects and view them, add them to his personal bookshelf or add it to the course material of his Project Management course. The learning object resides in the original repositories and is not downloaded locally; it is just linked into the LCMS together with certain metadata information.



Content Manager

Help

Create Edit Delete Release Version Preferred contents

Show content	Name	Language	Version			Entries: 13
All	management	English (US)	All	Search	Extended search	200 / Page 1 / 1

	Name	Object ID	Description	Type	Version	Published version	Creator	Last modified	Action
☐	10 Secrets of Time Management For Salespeople	27892	Gain the Competitive Edge and Make Every Second Count		6.0	6.0	Hofstadt, Sz	May 7, 2009 10:13 AM	
☐	A Guide to project management	500598	PDF Document		1.0	1.0	Degel, Gerhard	Jan 19, 2009 10:16 AM	
☐	Advanced Data Management in Excel	392022	Provides an overview of the advanced data management and organization features in Excel		3.0	3.0	Rommel, Peter	Dec 10, 2008 12:10 PM	
☐	Management, Leadership and Charisma	405977	PPS by Peter Andras		1.0	1.0	OpenScout	Sep 30, 2008 3:35 PM	

Figure 8: Access OpenScout from within LCMSs

In addition to the above examples the OpenScout architecture has to realize further modular services that facilitate access from external systems. Following are examples for additional services:

- Collect **usage data**: collect the number of views, downloads, bookmarks etc. Report usage data from the external system to the OpenScout portal where metadata of learning objects are updated with usage data and are reported back to the external repositories.
- **Recommendation** services. It should be possible to add comments and user **ratings** to the learning object. This requires implementing a rating functionality (widget) that can be used on the OpenScout web portal but that should also be integrated into the external applications. The ratings should be stored and further processed as additional metadata on the central site. A more advanced possibility would be to connect existing rating functions in external system with the OpenScout rating functions. This would require application specific translation functions/plugins for the respective external system to be provided by OpenScout
- Score: Both usage data and ratings (and additional data like quality of the repository) can be used to calculate an overall score for the learning object.
- **Sharing and other social services:** If a user embeds the OpenScout search widgets into his/her social network application the social graph of the user should be accessed, e.g. to search which OpenScout contents your friends recommend, bookmark, share with friends, download, or use/adapt for their personal use (increase trust). To implement sharing services OpenScout needs to connect certain profile information (user data), friends information (social graph) and activities (things that happen on an

external social network, news feeds, bookmarks, etc.) with its services using the OpenSocial standard.

- Competency profiles: If a user already has a competency profile e.g. from his LCMS, it should be possible to export this profile and use it for a competency based search on OpenScout. Hence some relation/translation between the competencies of the LCMS and the OpenScout central competencies (WP2) need to be defined. As an advanced option a real time connection (without prior export/import) of the competencies in the external system with the OpenScout competence hierarchy could be considered. Again application specific plugins would help to perform the translation between the competency hierarchies.
- **Upload**: A user should be enabled to upload (changed) learning objects from within LCMS or within Social network to one of the OpenScout federated content repositories.
- **User profiles**: OpenScout could consider to relate general user profiles within social network with OpenScout's competencies.
- Application specific plugins: For a number of connection to external system additionally application specific plugins should be provided by OpenScout to enable developers of the target system to access the services without much additional programming effort.

.

4 Implementation Plan

Based on the concepts and descriptions provided in the previous sections about the architecture of the system and the demands and possibilities for content access a thorough study about the best web portal implementation has been carried out. Possible technologies were investigated and their suitability for the most efficient implementation of our web portal is estimated. In the following sections the layout of the service to be provided is described and possible technology solutions are analyzed.

4.1 Service Layout

A basic aim for the proposed portal is to keep the layout as clean and straightforward as possible offering the user the possibility to immediately realize most of its functionalities. Figure 9 is presenting the proposed layout. The drafted layout is divided into three logical sections providing the user with a meaningful interface. These three sections are the user menu, the tool and services full view (main content page) and the tools compact view section. The implementation of the tool can be chosen from a variety of technologies as are for example widgets loaded externally, interface to an external web service or possibly even a collection of data such as web links to external resources or local content.

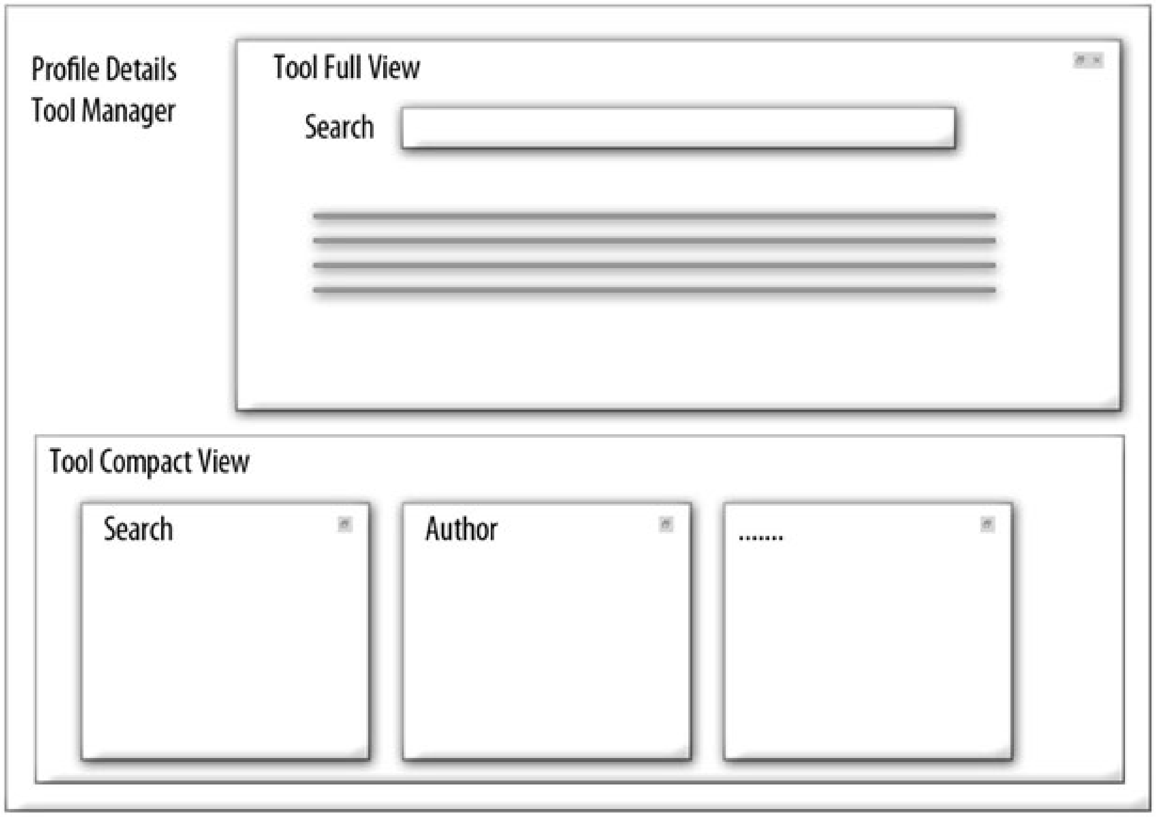


Figure 9: Tools/services main UI

A brief description of the portal layout is following.

4.1.1 User Profiles

Each user will have a specific user profile. Users will be able to process their profile details in the user menu. Every user has a profile page; this page holds accounting information as well as learning related information. Possible fields of information presented here can be skills, competencies, progress log of previous sessions, preferred learning content types, rankings etc. Profile data will be stored at the connector side in order to keep profiles from being centric to the web portal. A profile tool can be used to manipulate profile data, this tool will be implemented in the same manner as any of the other tools. Part of the profile data will be shared in a web service manner as this will enable us to employ user specific data and preserve a state among our tools.

4.1.2 Tools and services interface

According to the described design every tool will be presented in two ways. These two ways are the Full view and the Compact view. In the full view the tool is displayed with all available options whilst the second one displays minimal options. In terms of usability this makes the portal more pleasing since every tool is made visible and usable directly rather than having a simple link to it. For example the “Search tool” in its compact view can just display an input field for the search terms and a search button whilst in its full view will display more advanced capabilities as various filtering options, amounts of search-able data, available repositories, statistics and/or previous search queries.

Tools can be enabled, disabled or parameterized (whenever this choice is available) through the tool manager. Managing of our tools ensures that the user is presented only with the desired functionality and also keeps the portal interface from being bloated.

4.2 Platform description

The main architectural strategy for the portal implementation is to keep it as modular as possible ensuring that it can scale and that it can be modified and changed as the requirements will dictate during the OpenScout project progress. For the realization of the above concept the most widespread technologies in the web application world have been selected and studied. The implementation approach has been chosen to be based either on LAMP[8] technology or on a java middleware stack based on application servers such as tomcat or glassfish. Both approaches are made entirely from open source components, from the operating system to the databases to portal itself. Criteria for the choice of the components have been the existence of a support community behind them thus ensuring continuous development and help; a very important factor is also the size of active deployments of the components, a fact that ensures the previous well known behaviour and implementation of the solution.

The envisioned platform architecture is depicted on Figure 10.

[8]LAMP is an acronym for a solution stack of free, open source software, originally coined from the first letters of Linux (operating system), Apache HTTP Server, MySQL (database software), and PHP, Python or Perl.

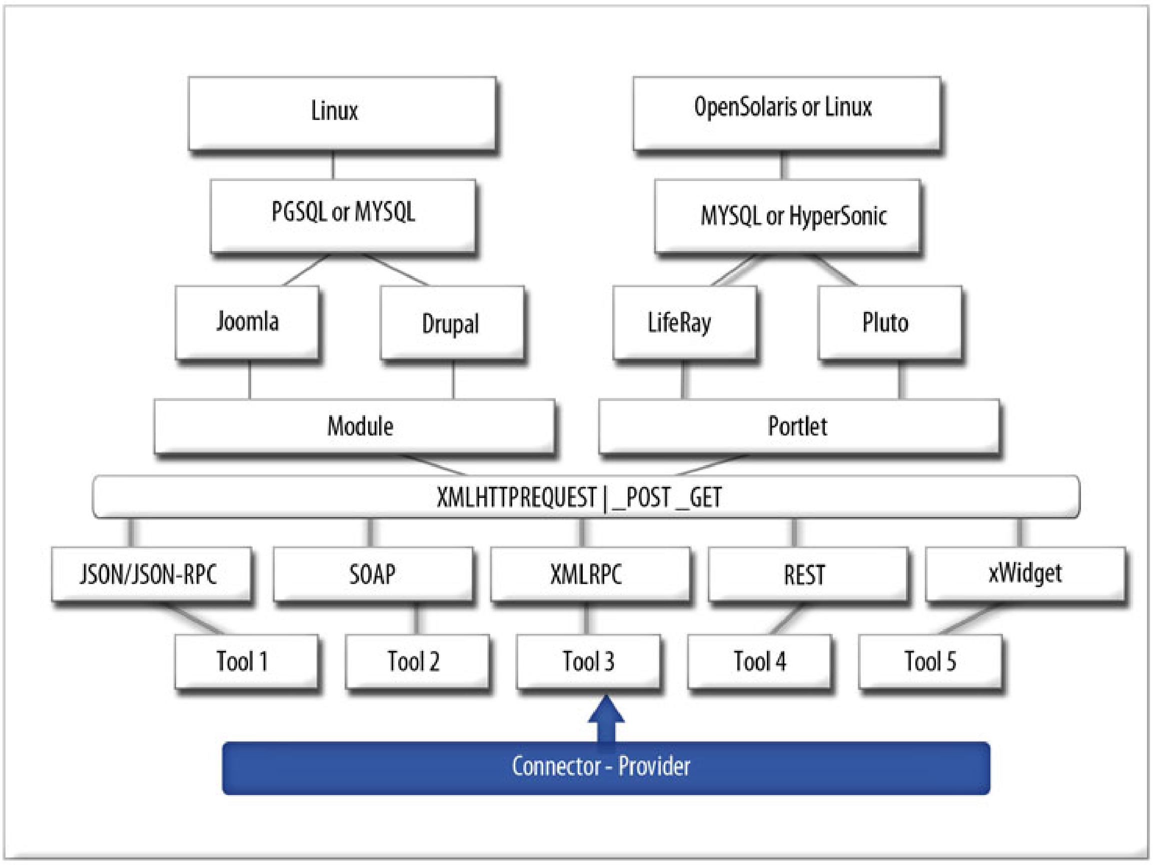


Figure 10: envisioned platform layout

As depicted in figure 10 at the base of each platform resides an open operating system. On top of it databases, required by the portal; and the portal itself are deployed. Depending on the platform used modules or portlets are integrated in the portal which exposes their corresponding methods for presenting the tools as they are exposed and served by the connector. The connector will provide an interface for each tool and service based on any of the previously mentioned technologies, thus the portal will consume and display the data accordingly. Interoperability between the tools is also guaranteed at the connector side through the use of the shared profile data.

4.2.1 LAMP stack

LAMP technology is the most widespread and fast solution for building web applications. It is widely used for a huge number of applications even mission critical applications all over the web-world. This can ensure us about its effectiveness as a platform, its robustness and about the experience provided through its users' community. This platform provides several advantages for developers who use it. It is easy to code and also it can be easily deployed as PHP comes as a standard apache module. Furthermore it has no special requirements regarding the server that will be deployed.

LAMP application platform consists of an operating system, web server, database and a scripting language. Their functional interconnection is depicted in figure 11.

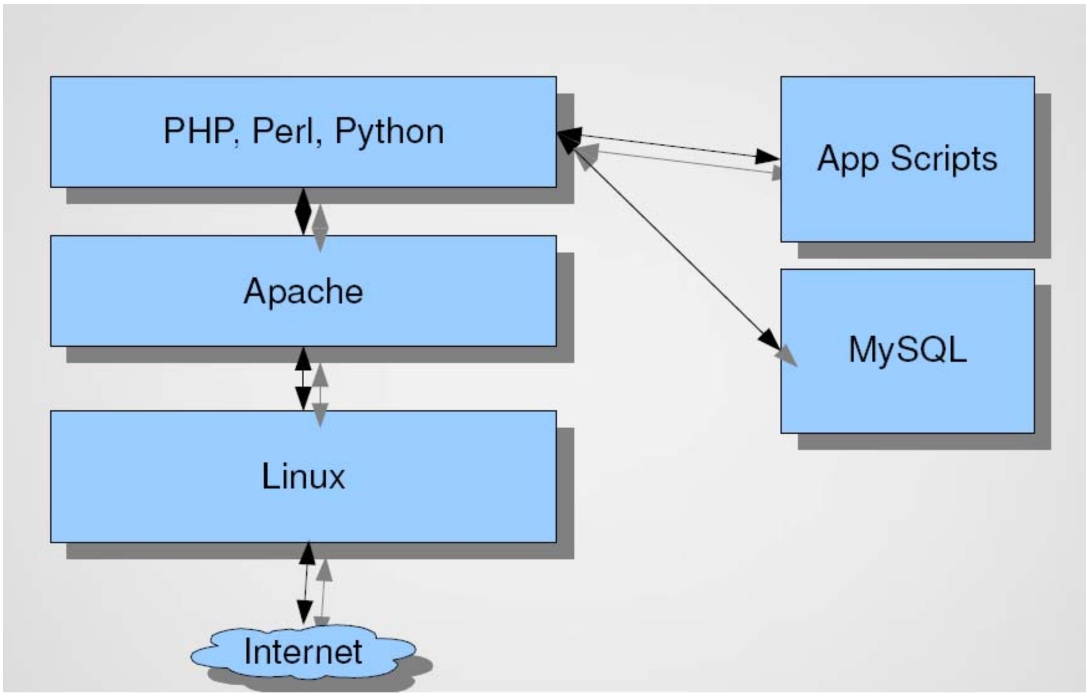


Figure 11: LAMP components functionality

LAMP implementations make use of Linux which is an open operating system and has proven to behave excellent for applications where server robustness is required. The applications, in LAMP implementation, are hosted on the Apache web server. Apache web server is a very fast, secure and reliable solution for serving static and dynamic content. Apache follows a modular approach. As a result of this modular approach most of the modern scripting languages can be used to build web applications as the Apache can integrate interpreters for these languages. Some of the most known ones are PHP, Perl and Python. These languages cover web technology in every aspect (robust rapid web service development, XML handling, database connectivity and more). Finally the database server used are MySQL and pgSQL which are open source relational databases with a lot of deployments; they are easy to use and administrate they are supported by all languages and frameworks, they are small and at the same time powerful engines and they upgrade easily.

4.2.2 Portal solutions for LAMP

The popularity and wide use of LAMP has led to the development of a variety of content management systems. Two of the most popular ones are Joomla and Drupal[9] which are both written using PHP.

[9]http://www.joomla.org, http://drupal.org

Joomla is a CMS written following the MVC[10] model thus can be extended in a straightforward way. Content is presented using a mixture of components, plug-ins and modules. A component in the Joomla lingo represents a separate application that can be shown inside Joomla as content. A plug-in modifies parts of Joomla before presentation, think of it as a macro implementation. Finally modules can be thought of as siblings to components. Modules are a quick way for displaying information relevant to the components, for example a newsletter component can have the subscription form displayed in a module. In OpenScout terms a separate component has to be build for every tool irrespective of the technology the tool is using (SOAP, Web widget, etc.). Modules can enhance the functionality for each tool (e.g. the search tool can have a "recent searches" module). Summing up on the Joomla CMS we can state the following:

- **Advantages**
  1. Follows the MVC model thus custom components can be integrated easily.
  2. Straightforward template system to use with new components.

- **Disadvantages**
  1. Limited user management (no groups).
  2. Lacks built in functionality for features like single sign-on, workflows and web service integration.
  3. Many components are commercial
  4. Poor developer documentation

Drupal is a solid programming framework rather than being just a CMS. Furthermore it is easy to learn, specialized for web applications, and very efficient at reusing code and libraries. It can be extended through community contributed modules. Although is not pure MVC it provides a very clean API that makes integration with external tools easy.

- **Advantages**
  1. External modules supporting integration with JSON, REST, SOAP services.
  2. Efficient user management.
  3. Built-in OpenID support.
  4. Workflow management.
  5. Excellent audit trail.

[10] Model – View –Controller (http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller)

- **Disadvantages**
  1. Slow learning curve in order to fully understand the inners of the system.

4.2.3 Java application Server

An alternative and competitive solution for the platform to be chosen is a Java based web implementation.

At the base of this platform an open operating system is used as Open Solaris or Linux. Accordingly the Java Platform (Java EE) is installed and used for server programming. Java EE is the industry standard for enterprise Java computing and is used to create next-generation web applications. The many years of numerous solutions' implementations based on this platform assure the experience to be provided by the community and the required robustness.

For the implementation using the Java EE the portal is considered as a web application. A Web application is a set of web components (servlets, JSP), libraries, static resources that are set up and configured in the web server and provide the required services. In this case additionally to the web server a web container or application server that can serve the content is needed. The web container is deployed on top of the web server and is used to run web applications. Famous application servers are TomCat, Glassfish, JBoss e.t.c.

The interaction between a web client and a web application is illustrated in Figure 12. The client sends an HTTP request to the web server. A web server that implements Java Servlet and JavaServer Pages technology converts the request into an `HTTPServletRequest` object. This object is delivered to a web component (Web components are either Java servlets, JSP pages, or web service endpoints), which can interact with JavaBeans components or a database to generate dynamic content. The web component can then generate an `HTTPServletResponse` or it can pass the request to another web component. Eventually a web component generates a `HTTPServletResponse` object. The web server converts this object to an HTTP response and returns it to the client

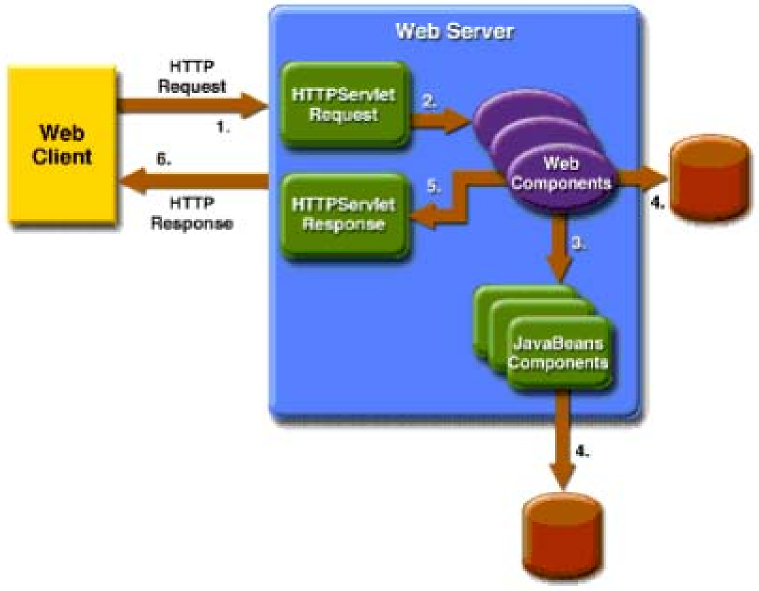


Figure 12: Java Web Application request Handling

Portlets produce fragments of markup code that are aggregated into a portal page.

Portlets are the way to provide the required functionality in a java technology implemented web portal. Portlets are small web applications that run in a portion of a web page and they constitute the heart of any portal as all of its functionality resides in its portlets. A portal's core is a portlet container. The container's job is to manage the portal's pages and to aggregate the set of portlets that are to appear on any particular page. This means that the core doesn't contain application code. Instead, all of the features and functionality of the portal application must reside in its portlets.
Portlets are pluggable user interface software components that are managed and displayed in a web portal and a Java standard portlet should be deployable on any portlet container which supports the standard. So this architecture provides the possibility of producing reusable tools which can be later integrated in other web applications.
As an example of this platform implementation we will consider two well known and tested portlet containers liferay and jetspeed

4.2.4 Portal solutions for Java

Several solutions exist extending the Java Servlets API providing both loose and strong MVC web application frameworks. Two of the most notable ones are Jakarta Struts and Springs. Liferay and Jetspeed2 lay on top of these frameworks and are the most widespread portal engines in the Java world.

Liferay

Liferay portal works on top of apache Struts framework and complies with Java Portlet Specification (JSR168, JSR286) thus ensuring easy integration and interoperability for portlets between different web portals. Liferay is an elegant portal with much out of the box functionality.

- **Advantages**
    1. Extensible and secure
    2. SSO Manager
    3. Collaboration features (integration with social networks)
    4. Built in support for web service integration
    5. Rich portlets using ICEfaces
    6. Easy layout customisation (dragable portlets)

- **Disadvantages**
    1. Documentation is outdated or hard to find
    2. Loose community
    3. Professional support is commercial
    4. Resource hungry
    5. Open source version is not extensively tested (bleeding edge)

Jetspeed2

Jetspeed is an Open Portal Platform and Enterprise Information Portal, written entirely in open source under the Apache license in Java and XML and based on open standards. Like Liferay discussed above Jetspeed2 can act as a portal container making information from multiple sources available in an easy to use manner. Jetspeed2 supports both 1.0 and 2.0 Java Portlet Standards.

- **Advantages**
    1. Better community support (Apache software foundation behind it)
    2. Spring-based Components and Scalable Architecture
    3. SSO Manager
    4. Rich portlets using ICEfaces
    5. Drag and drop moving of portlets

- **Disadvantages**
    1. Slow learning curve (requires in-depth knowledge of all java technologies behind it)

4.2.5 Summary

In the above study a brief analysis of the most popular portals and underlying technologies was made. Based on our studies and facts and on the analysis provided by CMSWire[11] the following chart summarizes our view on the issue.

[11] CMS Wire audience consists of expert technologists, decision makers, vendors and analysts with a focus on web and enterprise content management, social media, web publishing, collaboration practices and related technologies (http://cmswire.com)

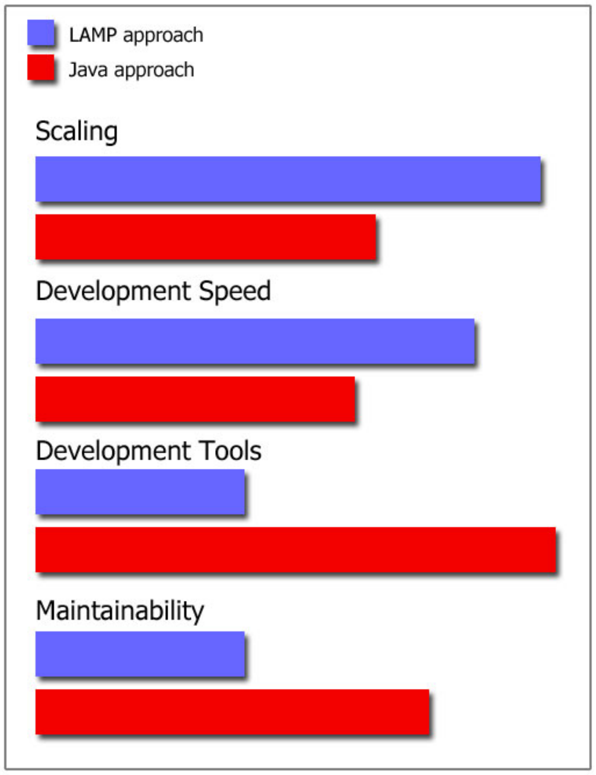


Figure 13: LAMP & Java web comparison chart

In order to have a best view of the intended results test implementations of the mentioned technologies will take place. At the moment widget integration tests in a LAMP stack are taking place. For this reason Joomla platform is used as underlying technology. Also a Liferay installation is planned to be performed real soon so an in-depth comparison can be performed. Finally other promising technologies as is Ruby on Rails will be reviewed as well.

4.3 MACE Implementation

Since there are functionality overlaps between OpenScout and the MACE project especially for the metadata harvester part, we briefly describe the MACE implementation, part of which can be re-used after further investigation.

According to its documentation[12], MACE also used OAI-PMH for the metadata harvester and SOAP for connecting remote services. As in other projects, LOM is also the choices of MACE for describing the application profile. In addition to OAI-PMH, MACE used RSS [13](Rich Site Summary, also known as “Really Simple Syndication”) for usage metadata harvesting. Regarding the web portal, MACE relies on Widget technology.

We will consider the MACE base technology which is available as open source as a basis for our developments as a back-end. The decision will be taken after further cooperation talks with the MACE project partners – the decision, however, does not influence the work plan or will cause any deviations but might lead to new synergies.

5 Conclusion and Outlook

OpenScout web portal will be the single point of access for tools, services and content of the project. Different tools and services can be integrated in other external applications but at the OpenScout web portal a thorough presentation of all tools and services for end users will be provided. Under this perspective the web portal can be seen as a “show case” for the various services presentation and mash up. Of great importance for the configuration and tuning of the portal will be the final implementation of the components interconnection.

The portal platform implementations and tests have already started but details will be finalized and fixed later on as the components connection implementation evolves.

[12] Moritz Stefaner, Elisa Dalla Vecchia, Massimiliano Condotta, Martin Wolpers, Marcus Specht4, Stefan Apelt, and Erik Duval. MACE – Enriching Architectural Learning Objects for Experience Multiplication. EC-TEL 2007, 2007, LNCS 4753, pp. 322–336, 2007.

[13] http://www.rssboard.org/rss-0-9-1-netscape

6 References

[1] Facebook start page. http://www.facebook.com.

[2] iGoogle start page. http://www.google.com/ig.

[3] Netvibes start page. http://www.netvibes.com.

[4] OpenSocial. http://www.opensocial.org/.

[5] ICT ROMULUS. D4.1 – annual report on mashup integration. Report, ICT ROMULUS, 12 2008.