Project Report

**ID5.15: New Core specifications v2**

**Workpackage**  WP5

**Task**  5.2

**Date of delivery**  
- **Contractual:** 31-03-2009  
- **Actual:** 10-04-2009

**Code name**  ID5.15  
**Version:** 1.0  
**Type of deliverable:** Report

**Security (distribution level)**  Public

**Contributors**

**Authors (Partner)**  
- Alessandro Mazzetti (Giunti Labs),  
- Michele Dicerto (Giunti Labs),  
- Alexander Grigorov (SU),  
- Sergej Zerrr (UHANN),  
- Juri Luca De Coi (UHANN),  
- Ricardo Kawase (UHANN),  
- Mar Perez (UPF),  
- Angel Roldan (Altran),  
- Pradeep Mittal (INSEAD)

**Contact Person**  
- Alessandro Mazzetti (Giunti Labs).

**WP/Task responsible**  Giunti Labs

**EC Project Officer**  Martin Májek

**Abstract (for dissemination)**  
This document describes the new functionalities for LearnWeb/KRService, which will be implemented in the new version called V0.3 and which will be available in late spring 2009. The main topics cover: resource functionalities, integration with other TENCompetence tools and social functionalities.

**Keywords List**  
WP5, internal deliverable, core services, specification, LearnWeb, KRService

TENCompetence Project Coordination at: Open University of the Netherlands  
Valkenburgerweg 177, 6419 AT Heerlen, The Netherlands  
Tel: +31  Fax: +31  Email: ten.competence@ou.nl
Version History

<table>
<thead>
<tr>
<th>Version</th>
<th>Status</th>
<th>Date</th>
<th>Author(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>Draft</td>
<td>23.10.08</td>
<td>Alessandro Mazzetti, Giunti Labs</td>
</tr>
<tr>
<td>0.2</td>
<td>Draft</td>
<td>30.12.08</td>
<td>Alessandro Mazzetti, Giunti Labs</td>
</tr>
<tr>
<td>0.3</td>
<td>Draft</td>
<td>13.01.09</td>
<td>Alessandro Mazzetti, Giunti Labs</td>
</tr>
<tr>
<td>0.4-xx</td>
<td>Draft</td>
<td>15.02.09</td>
<td>Every partner</td>
</tr>
<tr>
<td>0.5</td>
<td>Draft</td>
<td>24.02.09</td>
<td>Alessandro Mazzetti, Giunti Labs</td>
</tr>
<tr>
<td>1.0</td>
<td>Final</td>
<td>10.04.09</td>
<td>Alessandro Mazzetti, Giunti Labs</td>
</tr>
</tbody>
</table>

Summary of Changes

<table>
<thead>
<tr>
<th>Version</th>
<th>Section(s)</th>
<th>Synopsis of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>All</td>
<td>Draft: Table of Contents</td>
</tr>
<tr>
<td>0.2</td>
<td>All</td>
<td>New ToC compliant to DIP-4</td>
</tr>
<tr>
<td>0.3</td>
<td>All</td>
<td>Assignments to partners</td>
</tr>
<tr>
<td>0.4-xx</td>
<td>All</td>
<td>Every partner’s contribution</td>
</tr>
<tr>
<td>0.5</td>
<td>All</td>
<td>Integration of all contributions</td>
</tr>
<tr>
<td>1.0</td>
<td>All</td>
<td>All contributions reviewed and agreed</td>
</tr>
</tbody>
</table>
# Table of Contents

1. **INTRODUCTION (GIUNTI)** ................................................................. 5

2. **ARCHITECTURE** .............................................................................. 5
   2.1 AUTHENTICATION (SU) .................................................................. 5
   2.2 REGISTRATION (SU) ....................................................................... 7
   2.3 SINGLE SIGN ON (UHANN) ............................................................ 8

3. **USER INTERFACE** ............................................................................. 8
   3.1 HOME PAGE AND PAGES FLOW (UPF + GIUNTI) .......................... 8
   3.2 TOOLBARS (UHANN + GIUNTI) .................................................... 11
   3.3 BROWSERS (UHANN) ................................................................. 13
   3.4 LIFERAY ISSUES (GIUNTI) .......................................................... 14

4. **RESOURCE FUNCTIONALITIES** ......................................................14
   4.1 TAG CLOUDS (ALTRAN) ............................................................... 14
       4.1.1 KRService issues ................................................................. 14
       4.1.2 LearnWeb issues ................................................................. 16
   4.2 DOWNLOADING (SU) .................................................................... 16
       4.2.1 LearnWeb issues ................................................................. 16
   4.3 SUPPORTING UoL (SU) ............................................................... 16
       4.3.1 KRService issues ................................................................. 17
       4.3.2 LearnWeb issues ................................................................. 17
   4.4 CATEGORY MANAGEMENT (GIUNTI) .......................................... 17
       4.4.1 KRService issues ................................................................. 17
       4.4.2 LearnWeb issues ................................................................. 17
   4.5 RESOURCE GROUPS (UHANN) .................................................... 18
       4.5.1 LearnWeb issues ................................................................. 18
   4.6 BOOKMARKING (UHANN) ............................................................ 18
       4.6.1 LearnWeb issues ................................................................. 18
   4.7 SHARING (UPF + GIUNTI) ........................................................... 19
       4.7.1 KRService issues ................................................................. 20
       4.7.2 LearnWeb issues ................................................................. 20
       4.7.3 LearnWeb issues ................................................................. 20

5. **INTEGRATION FUNCTIONALITIES** .................................................21
   5.1 INTEGRATION WITH ReCOURSE (GIUNTI) .................................... 21
       5.1.1 KRService issues ................................................................. 21
       5.1.2 LearnWeb issues ................................................................. 21
   5.2 INTEGRATION WITH PDP (SU) .................................................... 21
       5.2.1 KRService issues ................................................................. 22
       5.2.2 LearnWeb issues ................................................................. 22
   5.3 INTEGRATION WITH PCM (SU) ................................................... 22
       5.3.1 KRService issues ................................................................. 22
       5.3.2 LearnWeb issues ................................................................. 29
   5.4 INTEGRATION WITH TenTUBE (GIUNTI + INSEAD) ..................... 31
       5.4.1 KRService issues ................................................................. 31
       5.4.2 LearnWeb issues ................................................................. 31

6. **SOCIAL FUNCTIONALITIES** ..........................................................32
   6.1 COMMENT RATING (GIUNTI) ...................................................... 32
       6.1.1 KRService issues ................................................................. 32
       6.1.2 LearnWeb issues ................................................................. 32
6.2 USER SEARCH (ALTRAN)........................................................................................................32
  6.2.1 KRService issues..............................................................................................................33
  6.2.2 LearnWeb issues..............................................................................................................33
6.3 MANAGING ABUSES (GIUNTI) ..........................................................................................33
  6.3.1 KRService issues..............................................................................................................34
  6.3.2 LearnWeb issues..............................................................................................................34
6.4 STIMULATING USERS (GIUNTI) .........................................................................................34
  6.4.1 KRService issues..............................................................................................................36
  6.4.2 LearnWeb issues..............................................................................................................36

7 CONCLUSIONS (GIUNTI) ......................................................................................................36
8 REFERENCES..........................................................................................................................37
9 APPENDIX A ..........................................................................................................................37
  9.1 PROTUNE (UHANN)............................................................................................................37
1 Introduction (Giunti)

This document specifies the functionalities and the architecture of the version to be delivered in late spring 2009 of the tools:

- LearnWeb
- KRService

This document refers to the following:

- ID5.13 (new requirements), delivered on 28-11-2008, [http://hdl.handle.net/1820/1607](http://hdl.handle.net/1820/1607)
- Outcome of Sofia and Düsseldorf meetings, document of 15-12-2008 [http://docs.google.com/View?docid=ddhtqnsh_663c8h7vtf](http://docs.google.com/View?docid=ddhtqnsh_663c8h7vtf)
- DIP-4 objectives and tasks, document of 19-12-2008 [http://docs.google.com/Doc?id=ddsnp3t6_28hcxmphpf](http://docs.google.com/Doc?id=ddsnp3t6_28hcxmphpf)

The functionalities here described will be implemented in LearnWeb v.0.3 that will constitute ID5.16

2 Architecture

2.1 Authentication (SU)

LearnWeb2.0 Web tool and the KR Web services will use a CAS (Central Authentication Service) server for SSO user authentication within TENCompetence as suggested by WP3. A simplified diagram of the workflow is shown on Figure 2.1.

![Figure 2.1. User authentication using a CAS server](image-url)
Here is a short description of how authentication with CAS works (a more detailed description of SSO with CAS is given in [1]).

**Authenticating a user**

A non previously-authenticated user tries to access LearnWeb for an action that requires authentication; the browser is redirected to the CAS server. The CAS server is presented an authentication form, in which the user is invited to enter a netId and a password. If netId and password are correct, the server sends a cookie called TGC (Ticket Granting Cookie) to the browser. The TGC is the user’s passport against the CAS server. Its lifetime (validity) is limited (typically a few hours). It is the way for web browsers to get tickets (meant for CAS clients, i.e. LearnWeb) from the CAS server, without needing to re-authenticate.

**Accessing a protected web resource when authenticated**

When accessing a resource protected by a CAS client (i.e. LearnWeb), the web browser is redirected to the CAS server. The browser, previously authenticated, provides the CAS server its TGC.

On presentation of the TGC, the CAS server delivers a Service Ticket (ST). It is an opaque ticket (no user information), and is usable only by the service that required it. At the same time, the CAS server redirects the browser to the calling service (LearnWeb).

The ST is then validated by the CAS client (LearnWeb) against the CAS server and the wanted resource can be delivered to the browser (see Figure 2.2).

![Figure 2.2. Validation of Service Ticket](image)

All the redirections above are transparent for the user: he accesses the resource without authenticating, and without interacting at all.
Accessing CAS back-end services

When an access to CAS back-end services (i.e. KR Services, TENC server) is required, LearnWeb should be used as a CAS proxy.

A CAS proxy, when validating a Service Ticket to authenticate a user, also enquires a PGT (Proxy Granting Ticket) from the CAS server:

A PGT is a CAS proxy’s passport, for a user, against the CAS server. It is the way for CAS proxies to get tickets (meant for CAS back-end services) from the CAS server, without needing to validate a ST. It is an opaque and re-playable ticket, delivered by the CAS server through a secured request, to insure its integrity and confidentiality. PGTs’ lifetime is limited (a few hours, as well as TGCs).

PGTs are for applications the equivalent of TGCs for web browsers. A PGT allows applications (CAS proxies) to authenticate a user against the CAS server, and get Proxy Tickets. Proxy Tickets are validated by the CAS server before giving access to protected resources (see Figure 2.3).

![Figure 2.3. Validation of a Proxy Ticket by a back-end service](image)

For the implementation of authentication with CAS in LearnWeb we will use the casPHP library and the sample code from WP3. The KR services will implement CAS authentication in a similar way as in the TENC server.

### 2.2 Registration (SU)

All TENCompetence web tools will use a centralized single place (web page) for user registration. So user registration stays out of the scope of LearnWeb.
Meanwhile, by the time the centralized user registration is implemented, LearnWeb will support and maintain a page for user registration.

### 2.3 Single Sign On (UHANN)

Since SSO cannot be performed on the client side browser independently, in the future the sign on into all used Web2.0 tools will be done using the provided API. In case the tool does not provide an authentication API, one authentication wrapper over the API should be created, or the tool should be exchanged against more suitable one.

The authentication through the API, on example of resource upload, works as follows:

1. The user selects a link for uploading a resource. This links leads to the authentication page of the Web2.0 tool.
2. If the user is not logged into the Web2.0 tool, the login page is displayed, otherwise the user is already authenticated, step 3 is omitted and the process continues with the step 4.
3. User authenticates him/herself directly by the Web2.0 tool using the mechanism and pages, provided by the tool.
4. The user is redirected to the LearnWeb2.0 upload form and can upload the resource.

Currently YouTube, Flickr and Ipernity are providing the authentication API’s and this is used in LearnWeb. However the authentication is not yet durable, that means that in order to execute an action each time the authentication process has to be run and new token is requested.

This should be improved. The authentication should occur once per session and the session token should be stored in LearnWeb and re-used.

The form provided for storing user credentials has been dropped out.

### 3 User interface

#### 3.1 Home page and pages flow (UPF + Giunti)

**Entering the system for the first time**

The home page is the first page that the users see when entering the system. This first page serves as the presentation card of the system (Figure 1). It should contain the following information:

- A little Welcome message for the new users
  - E. g. “Welcome to LearnWeb 2.0, the tool for discovering, exploring and sharing resources. Please, login in the registration button on the upper menu”
- Registration and login functionalities will be on the right-hand of the upper menu.
The search browser on the left-hand

**Home Page functionalities for a logged user**
The home page is the main page that the user will interact with. The look & field has to encourage sharing within users and, for that, we include the following functionalities and views:

- An upper menu accessing the main functionalities:
  - We drop out from the menu the MyHomePage. However, the Home tab will remain because will facilitate the navigation for the user if he wants to come back to the main search page.
  - Home: includes recently updated resources and a welcome message encouraging the users to register the system (e.g. “LearnWeb 2.0 a tool for sharing and exchange knowledge resources”).
  - Search: the page where the users search for resources classified by categories. It includes a simple explanation of how the searching functionality works.
  - Help: manuals and guides for the users.
- On the top, the search filed (in the same way that in the current version)
- On the left, a line with the presentation of the tool and a link to the help tools.
- In the middle, a list of recently uploaded resources.
- On the right, the Tag Cloud (referring to all tags).

**Figure 1** Main page when the user has not been login.

**My Home Page**
Once the user has been registered to the system, he/she can access to the system and will be redirected to the same page than in the beginning but with a new functionality called “MyHomePage”(Figure 2).

MyHomePage is a page where the user can manage the main functionalities that the tool offers:
“Manage Resources” (Left-hand of MyHomePage). It is where the user can find all the functionalities regarding the management of resources: upload a file, a picture, create a group....

“My resources” (Right-hand of MyHomePage). It is where the user can find the information about his/her resources: my tags, my rates.

**Figure 2** MyHomePage of a fogged user.

**New functionalities**

This section contains the functionalities that have to be added to the tool in order to improve its usability.

**Navigation buttons**

In order to provide the user with a more friendly interface it will help to add a back button in the bottom part of every page. This button will allow the user to come back to the previous page.

**Elimination of the Profile**

The Profile page can be deleted. The user should access directly to his profile by pressing over his/her profile picture. He/she would be redirected to a page for allowing the different applications that the LearnWeb manages (Group me!, YouTube ...) to be used in the program.

**Log files**

The system should include a log file system for analyzing and study how the user interact with the interface. For this, we can use the free tool called Google Analytics (https://www.google.com/analytics/). It stores the information of the user interaction in each of the pages of the tool. All this information will be later used for improving the usability of the tool and adapt it to the users’ necessities.
3.2 Toolbars (**UHANN + Giunti**) 

The goal is to develop a toolbar (extension) for the web browser (first moment to the FifeFox browser). This toolbar will work as a shortcut interface for the LearnWeb2.0 users. The aims are:

1. allowing direct links to LearnWeb pages and functionalities;
2. allowing adding a web resource directly while visiting it, without the burden to search it from LearnWeb;
3. allowing LearnWeb search without entering LearnWeb home page;
4. allowing the integration with the SpreadCrumbs in-context annotation system.

The below figures explain the toolbar functionality:

---

**Fig 3.2.1. LearnWeb toolbars**
When adding the current resource to LearnWeb the user is redirected to the LearnWeb “add resource” page with the contents of the resource and with fields already filled. In addition, the users can tag/comment/rate the resource.
Additionally a search field for the resources, annotated in LearnWeb2.0 will be available in the bar (see right part of figure 3.2.2). This brings the user directly into LearnWeb search page with the list already filled.

The implementation consists basically in the FireFox extension. It is a package of XUL and javascript implementation with AJAX calls for the PHP pages on our servers and for some external services APIs.

### 3.3 Browsers (UHANN)

Since LearnWeb2.0 is running on the server as PHP application and Java services, the application itself is browser and operation system independent. The basic limited functionality is available without forcing the user to install, or activate anything in her browser or desktop. However in order to increase the usability as well as to extend already available LearnWeb2.0 functionality, client running scripts, flash based applications, browser plugins and toolbars are used. Since all these Add-ons are running on the client side directly in the browser, there are some platform dependent features.

**Scripts:** We assume that the user has Javascript enabled in their browser. The code used in LearnWeb2.0 is kept platform independent and should work on any browser.

**Flash:** In order to display the resource, Flash player has to be installed on the browser. Macromedia provides a Flash plugin for the most browsers and platforms, thus the
players used in LearnWeb2.0 are supported on any configuration that is also supported by Macromedia.

**Plugins/Toolbars:** In a prototype only Geko based browsers (Seamonkey, Firefox, Geko) will be supported. This does not mean that IE, Safari and Chrome user will miss the basic functionality. The application functionality does not rely on the plugins.

### 3.4 LifeRay issues *(Giunti)*

LearnWeb will be an iFrame into LifeRay and authentication will be done automatically

### 4 Resource functionalities

#### 4.1 Tag Clouds *(Altran)*

A tag cloud or word cloud (or weighted list in visual design) is a visual depiction of user-generated tags used typically to describe the content of web sites. Tags are usually single words and are typically listed alphabetically, and the importance of a tag is shown with font size or colour\(^1\). Thus both finding a tag by alphabet and by popularity is possible. The tags are usually hyperlinks that lead to a collection of items that are associated with a tag.

It would be interesting to place the tag cloud at the homepage in order to allow the user to see what kind of contents has the website. Also it would be useful to show a tag cloud for the user’s tags instead of My tags list because by this way the user can see what kind of things are more frequently tagged for him quickly.

In fact, it would be interesting to place a cloud tag instead a list tag always because a cloud tag gives easily more information, as frequency or importance, than a tag list. Two examples of this are resource list and resource preview.

#### 4.1.1 KRSERVICE issues

It would be needed that Fedora could return a list of the most used tags with the number of times that each tag has been used for the whole system and per user, and per resource in case of use a tag list un resource list and/or resource preview.

Fedora doesn’t need a special structure to store the frequency because it can be easily calculated with a SQL query (that is shown below).

---

\(^1\) Martin Halvey and Mark T. Keane, An Assessment of Tag Presentation Techniques, poster presentation at WWW 2007, 2007
There is not documentation about algorithms of famous web 2.0 tools but a tag cloud algorithm can be very easy, just an example of these is:

**Step 1 - Get a list tags, and their frequency**

```sql
SELECT COUNT(tag) AS tagCount, tag
FROM tags
[WHERE (user=userID | resource=resourceID)]
GROUP BY tag
[HAVING tagcount > limit]
```

Note: [] means optional, () means list of possible options (user OR resource)

This query will show all tags for the whole system or filtered by user or by resource, but if there are a lot of tags, it is possible to set the min number of occurrences using a HAVING statement. For example HAVING tagCount > 5

**Step 2 - Find the Max and Min frequency**

In pseudo code:
tagValueArray is an array with the tag values
max has the maximum value from tagValueArray
min has the minimum value from tagValueArray

**Step 3 - Find the difference between max and min, and the distribution**

In pseudo code:
diff = max - min
distribution = diff / 3

It is possible to define the distribution to be more granular if is needed by dividing by a larger number, and using more font sizes below.

**Step 4 - Loop over the tags, and output with size**

In pseudo code (similar to php):
```
foreach $tag in $tags
{
    if $tag["tagCount"] = min
        $class="smallestTag"
    elseif $tag["tagCount"] = max
        $class="largestTag"
    elseif $tag["tagCount"] > (min + (distribution*2))
        $class="largeTag"
    elseif $tag["tagCount"] > (min + distribution)
        $class="mediumTag"
    else
        $class="smallTag"
    end if
```
Step 5 - Add css classes to stylesheet

```css
.smallestTag { font-size: xx-small; }
.smallTag { font-size: small; }
.mediumTag { font-size: medium; }
.largeTag { font-size: large; }
.largestTag { font-size: xx-large; }
```

4.1.2 LearnWeb issues

It would be needed to make a little function to order alphabetically the list of tags and to set the word size depending on its importance, an example of the function to set the word size is shown in the previous point.

4.2 Downloading (SU)

LearnWeb should provide explicit resource URL. This is needed for other tools to make a link to a resource in LearnWeb. For videos from YouTube or TenTube, LearnWeb should also provide embeddable code so the users could easily embed the videos in their own HTML pages.

For images and photos from Flickr, LearnWeb should provide means to download various photo sizes using Flickr API.

4.2.1 LearnWeb issues

The Preview page in LearnWeb should be modified to contain textboxes with explicit resource URL and embeddable code for videos. Also, the user can select the image size for viewing/downloading when previewing photos/images from Flickr.

4.3 Supporting UoL (SU)

Units of Learning (UoLs) will be stored in Fedora as File resources in .zip format. UoLs can be uploaded to Fedora by ReCourse (using the existing KR services) and/or manually from LearnWeb. Searching and retrieving can be done by the existing KR Services. If needed, new services will be developed. The work will be in close collaboration with WP6 and related Task Forces.
4.3.1 KRService issues

The existing KR Services will be explored in order to see to what extent they can handle UoLs. If needed, new services will be designed and implemented. Also, if it is necessary, a model for storing and retrieving relations between UOL and Learning Objectives and Prerequisites will be developed. For example, relations isLearningObjectiveOf and isPrerequisiteOf can be defined in Fedora and corresponding services for adding and querying for such relations can be implemented.

4.3.2 LearnWeb issues

The Preview and Add resource pages of LearnWeb will be adapted to handle adequately UoLs. The Preview page will show also links to Learning Objectives and Prerequisites if available for the UOL. Also, when clicking on the thumbnail of UOL, Sled will automatically open.

4.4 Category management (Giunti)

Categories in LearnWeb are intended to be managed only by an administrator. New functionality on this is required. In particular it is necessary a way to add new categories, remove categories, moving a resource from a category to another. Remove a category could present some issues if the category contains resources. In this case two approach can be used. Assign all resources to parent category or assign all resources to the root category. LearnWeb should present this option to the administrator.

4.4.1 KRService issues

New services are needed:
1. Add new category service
2. Remove category service
3. Resource category reassignment
4. Massive reassignment of resources from a category to another

4.4.2 LearnWeb issues

The application needs an admin section with pages for category management with add, remove category, resource reassignment. It will be also necessary to give to the user the possibility to modify the category of owned resources. This is achieved by modifying the “edit_metadata” page, adding two fields: “current category” and “set new category” with the tree panel.
4.5 Resource groups (UHANN)

The API for the GroupMe! tool for the handling of the resource groups is currently being developed and extended. LearnWeb2.0 will integrate GroupMe through the API as far as it will be available.

4.5.1 LearnWeb issues

The resource Groups are handled as resources. The resource group creation has to be performed on the GroupMe page directly and can than be annotated and added to the LearnWeb2.0 repository “Fedora”. The Group can contain not only resources, but also other resource groups. Currently the API for searching within GroupMe is implemented. We will maintain and improve the integration of the tool through the API.

4.6 Bookmarking (UHANN)

Delicious Web2.0 bookmark tool is used for managing of the bookmarks in LearnWeb2.0. Since Delicious itself does not provide an Authorization API, a wrapper will be created in LearnWeb2.0 that emulates the authorization. User account data will be stored on the server and used for accessing Delicious. The authentication mechanism, that the wrapper will use, is described in Error! Reference source not found. and will correspond to such of the tool Flickr.

4.6.1 LearnWeb issues

4.6.1.1 View bookmarks

By clicking on “My Bookmarks” link in case the user accesses the bookmarks for the first time an authorization process will be started. Otherwise the bookmarks are shown in the list.

4.6.1.2 Bookmark resource

While browsing the Fedora repository the user has a possibility to bookmark a particular resource. In case authentication was not yet performed, the authentication process will be started. Otherwise the page will be displayed with the possibility to add metadata about the resource like title, description and tags. The bookmark will be stored on delicious account directly.

4.6.1.3 Using different Delicious accounts

The user can anytime reset his Delicious account data from his LearnWeb2.0 homepage by clicking on the link “reset account” and create new by completing the authentication process like described in 4.6.1.1 and 4.6.1.2.
4.7 Sharing (UPF + Giunti)

Sharing, in this section, refers to the action that a user does when wants to share a resource with other users. These second users must be informed that someone has request for share a resource with them. In the sharing action we distinguish within two different processes: a) the action of sharing a resource with someone and, b) the action of accept the request from another user for sharing something. In this section, we provide different possibilities for enhancing these two actions in the system.

a) Share a resource with other user:

1. The system should include a social section in which the user is informed about the people that can be interested in the already uploaded resource. For example, if a user is uploading a resource about learning, on the left-hand or bottom part of the window s/he can be informed about some of the people that can be interested on such as resources. The user should be able to select from the list of people those with s/he wants to share the resource with.
   
   The critical aspect here is to determine what “interested” people means. We suggest define “interested people” as people related through tags, categories or rating. This is, when a user uploads a resource, he/she will be showed people interesting on the same resource like:
   
   - People that used the same tags
   - People that used the same category
   - People that commented or rate a resource classified in the same category.

2. the system should allow to send an email to people that are not yet TENC users. The following fields should be visible: “to”, “message”. The email is should contain a direct link to the LearnWeb preview page (example: http://learnweb.it.fmi.uni-sofia.bg/resources/view/resource:949) for allowing the recipient to see the comments, tags and rate.

h3. The system should also include a section called My shared resources in which it can appear the list of people who have shared something with.

b) Accept the request of a share resource:

1. In order to encourage sharing, when user X shares a resource with others, he/she must specify the e-mail address of his/her colleagues. The colleagues will receive a message indicating that user X wants to share a message with them and a direct link to the resource in the LearnWeb Platform. The message will be something like this:

   “Hi,
   John wants to share this resource with you!
   Click here for viewing this resource:
   http://learnweb.it.fmi.uni-sofia.bg/resources/view/resource:949
and enjoy the comments of TENCompetence users. If you are not yet registered, please consider the advantages of entering in the wonderful world of TENCompetence!”

If the colleague is agree, he/she will register to the system and see the resource. This technique will encourage new user to use the tool.

### 4.7.1 KRService issues

The two actions related with the sharing process explained above, have a set of implications on the items implemented in the KRService. We list here the new items that should be implemented in the resources for supporting the above-mentioned functionalities:

- **Share Users on the Resource:** The Resource should contain a new field with the list of users that can have the resource available.
- **Accepted/Rejected Resources:** Each User item should contain a list of ID resources that he/she has accepted or rejected.
- **Friends on the User:** The User should contain a new list of friends in which there are included the friends that he/she has shared with and the ID of the resource that he/she has shared with each of them.

### 4.7.2 LearnWeb issues

The two actions related with the sharing process explained above, have a set of implications on the items implemented in the LearnWeb. We list here the new items that should be implemented in the resources for supporting the above-mentioned functionalities:

- **Accept/Reject function:** A script function for accepting or rejecting the shared resource. This function should update the resource list of the user accepting or rejecting.
- **Invite someone function:** A script function for searching those users that have more commonalities and provide a suggestion list with users to share with. This should include the functionality of inviting someone.

### 4.7.3 LearnWeb issues

The two actions related with the sharing process explained above, have a set of implications on the items implemented in the LearnWeb. We list here the new items that should be implemented in the resources for supporting the above-mentioned functionalities:
- **Accept/Reject function:** A script function for accepting or rejecting the shared resource. This function should update the resource list of the user accepting or rejecting.
- **Invite someone function:** A script function for searching those users that have more commonalities and provide a suggestion list with users to share with. This should include the functionality of inviting someone.

## 5 Integration functionalities

### 5.1 Integration with ReCourse (Giunti)

A user should be able to search a resource in LearnWeb from within ReCourse, download it or link to it. The linking can be done through the URL (or embedded object code) that is provided through LearnWeb. This section is strictly related to Task Force 12.

#### 5.1.1 KRService issues

In principle no new services are needed, but it will be very useful an improved search service, allowing search based on resource type.

#### 5.1.2 LearnWeb issues

This task does not require web application improvement.

### 5.2 Integration with PDP (SU)

For the integration with PDP we need to be able to store, search and retrieve Learning activities (and other related objects if needed) in Fedora. The work within this section is strictly related to Task Force 12 and will be done in close collaboration with related Task Forces.

The integration will be done both on LearnWeb and WebPDP sides. In LearnWeb the “search”, “add” and “preview” page will be modified to handle Learning Activities properly.

In WebPDP facilities for publishing Learning Activities in Fedora, searching and importing Learning Activities from Fedora using the KR Services will be implemented.
5.2.1 KRService issues

A model how to store Learning Activities in Fedora should be developed. For example they can be stored as File Resources in XML format, or as Web Resources pointing to the corresponding Learning Activity in the TENC server.

After investigation, specific services for searching and retrieving Learning Activities will be designed and implemented.

5.2.2 LearnWeb issues

The “add” page will include a new entry “Learning Activity” in the “Type” selection menu.
The “search” page will display “Learning activity” in the “type” column.
The “preview” page will allow to play the Learning Activity by calling automatically PDP when the user clicks on the thumbnail

5.3 Integration with PCM (SU)

The integration with PCM (TENC server) is part of the activities within Task Force 12 and the work will be in collaboration with WP3 and related TaskForce.

The integration includes the following tasks:
- Identify what TENC objects and how they will be stored as resources in Fedora;
- Identify what relations between TENC objects exist and how they can be stored in Fedora;
- Design and implement general or specific KR services for uploading, searching and retrieving TENC objects and relations.
- Implement user authentication (CAS) and authorization (Protune) for KR Services and LearnWeb.

5.3.1 KRService issues

5.3.1.1 TENC objects (SU)
- Identify what TENC objects will be stored as resources in Fedora. Such objects are for example:
  - Attachments
  - Competence
  - Competence Profile
  - Evidences
  - Learning Path
  - Learning Path : objective
  - Learning Path : prerequisite
For each type of TENC objects identify what will be stored in Fedora – metadata only or/and the object itself and build corresponding Digital Object Models in Fedora;
Identify what relations between TENC objects exist and how they can be stored in Fedora;
Design and implement general or specific for each type of TENC object services for uploading, searching and retrieving.
Design and implement general or specific services for creating/deleting relations, searching for related objects.
Implement user authorization for accessing resources at KR Services level using the Protune framework.

5.3.1.2 Authorization services (UHANN)
UHANN is making authorization services available within the TENCompetence infrastructure by providing a Web Service interface to the Protune framework which TENCompetence applications may use for authorization purposes.

<table>
<thead>
<tr>
<th>addResource</th>
<th>Add owner and security level (cf. below) of a (newly-created) resource to the policy knowledge base</th>
</tr>
</thead>
<tbody>
<tr>
<td>deleteResource</td>
<td>Removes a resource from the policy knowledge base</td>
</tr>
<tr>
<td>update</td>
<td>Updates the security level of a resource</td>
</tr>
<tr>
<td>exists</td>
<td>Checks whether a resource is available in the policy knowledge base</td>
</tr>
<tr>
<td>getSecurityLevel</td>
<td>Returns the security level of a resource</td>
</tr>
<tr>
<td>getResources</td>
<td>Returns the resource ids of the resources having a given security level</td>
</tr>
<tr>
<td>selectAllowedResources</td>
<td>Selects the subset of resources a user can access out of a set of given resources</td>
</tr>
<tr>
<td>commit</td>
<td>Commits the last changes eventually performed to the policy knowledge base</td>
</tr>
</tbody>
</table>

Tab. provides an overview of the functionalities that are defined for LearnWeb2.0 and will be supplied by the TENCompetence authorization service. Such functionalities will be thoroughly described and exemplified in the following. Although the examples will show how the authorization service can be used over a GET protocol, the POST protocol is supported as well (and even exploited in the current implementation). In following the interface is described in the current step design.

5.3.1.2.1 addResource

You can invoke method addResource as follows
serviceURL²/addResource?resourceId=resourceId&repositoryId=repositoryId&ownerId=ownerId&securityLevel=securityLevel

where

resourceId: is the id of a resource a user just created
repositoryId: is the id of the repository where the resource is stored
ownerId: is the id of the user who just created the resource
securityLevel: represents the security level of the resource. Can be either 0 (private) or 1 (public) or 2 (only for logged-in users) or 3 (only for members of the same group)

The result will be...

    <ns:addResourceResponse>
    <ns:return/>
    </ns:addResourceResponse>

... if everything went fine or...

    <ns:addResourceResponse>
    <ns:return>
    errorDescription
    </ns:return>
    </ns:addResourceResponse>

... if something went wrong, where “errorDescription” is the description of the error occurred (typically the trace of the thrown exception).

5.3.1.2.2 deleteResource

You can invoke method deleteResource as follows

    serviceURL/deleteResource?resourceId=resourceId&repositoryId=repositoryId

where

resourceId: is the id of the resource to be removed
repositoryId: is the id of the repository where the resource is stored

The result will be...

    <ns:deleteResourceResponse>
    <ns:return/>

² Here and in the following with “serviceURL” http://10c.l3s.uni-hannover.de:9080/axis2/services/AuthService is meant.
5.3.1.2.3 update

You can invoke method **update** as follows

```
    serviceURL/update?resourceId=resourceId&repositoryId=repositoryId&newSecurityLevel=newSecurityLevel
```

where

- **resourceId**: is the id of the resource whose security level must be updated
- **repositoryId**: is the id of the repository where the resource is stored
- **newSecurityLevel**: represents the new security level of the resource. Can be either 0 (private) or 1 (public) or 2 (only for logged-in users) or 3 (only for members of the same group)

The result will be...

```
    <ns:updateResponse>
    <ns:return/>
    </ns:updateResponse>
```

... if everything went fine or...

```
    <ns:updateResponse>
    <ns:return>
    errorDescription
    </ns:return>
    </ns:updateResponse>
```

... if something went wrong, where “errorDescription” is the description of the error occurred (typically the trace of the thrown exception).
5.3.1.2.4 exists

You can invoke method `exists` as follows

```
serviceURL/exists?resourceId=resourceId&repositoryId=repositoryId
```

where

- `resourceId`: is the id of a resource
- `repositoryId`: is the id of the repository where the resource is supposed to be stored

The result will be...

```
<ns:existsResponse>
  <ns:return>
    ‘0’, ‘answer’
  </ns:return>
</ns:existsResponse>
```

... if everything went fine or...

```
<ns:addResourceResponse>
  <ns:return>
    ‘1’, errorDescription’
  </ns:return>
</ns:addResourceResponse>
```

... if something went wrong.

- In the first case “answer” is either `true` or `false`
- In the second one “errorDescription” is the description of the error occurred (typically the trace of the thrown exception)

5.3.1.2.5 getSecurityLevel

You can invoke method `getSecurityLevel` as follows

```
serviceURL/getSecurityLevel?resourceId=resourceId&repositoryId=repositoryId
```

where

- `resourceId`: is the id of a resource whose security level must be retrieved
- `repositoryId`: is the id of the repository where the resource is stored

The result will be...
... if everything went fine or...

    <ns:getSecurityLevelResponse>
    <ns:return>
    ‘0’, ‘securityLevel’
    </ns:return>
    </ns:getSecurityLevelResponse>

... if something went wrong.

- In the first case “securityLevel” is either 0 (private) or 1 (public) or 2 (only for logged-in users) or 3 (only for members of the same group)
- In the second one “errorDescription” is the description of the error occurred (typically the trace of the thrown exception)

5.3.1.2.6 getResources

You can invoke method getResources as follows

    serviceURL/getResources?securityLevel=securityLevel

where securityLevel represents the security level of the resource and can be either 0 (private) or 1 (public) or 2 (only for logged-in users) or 3 (only for members of the same group). The result will be...

    <ns:getResourcesResponse>
    <ns:return>
    ‘0’, ‘resources’
    </ns:return>
    </ns:getResourcesResponse>

... if everything went fine or...

    <ns:getResourcesResponse>
    <ns:return>
    ‘1’, ‘errorDescription’
    </ns:return>
    </ns:getResourcesResponse>
... if something went wrong.

- In the first case “resources” is a comma-separated list of single-quoted (resourceId, repositoryId) pairs, each of which has the format

  \`resourceId\', \`repositoryId\'

- In the second one “errorDescription” is the description of the error occurred (typically the trace of the thrown exception)

5.3.1.2.7 selectAllowedResources

You can invoke method selectAllowedResources as follows

```
serviceURL/selectAllowedResources?resourceIds={resourceIds}&repositoryId={repositoryId}&userId={userId}
```

where

- **resourceIds**: is a comma-separated list of single-quoted ids of the resources a user is trying to access
- **repositoryId**: is the id of the repository where the resources are stored
- **userId**: is the id of the user trying to access the resources

The result will be...

```
<ns:selectAllowedResourcesResponse>
  <ns:return>‘0’, ‘resources’
  </ns:return>
</ns:selectAllowedResourcesResponse>
```

... if everything went fine or...

```
<ns:selectAllowedResourcesResponse>
  <ns:return>‘1’, ‘errorDescription’
  </ns:return>
</ns:selectAllowedResourcesResponse>
```

... if something went wrong.

- In the first case “resources” is a comma-separated list of single-quoted resource ids
In the second one “errorDescription” is the description of the error occurred (typically the trace of the thrown exception)

5.3.1.3 commit

You can invoke method commit as follows

```
 serviceURL/commit
```

The result will be...

```
<ns:addResourceResponse>
<ns:return>
</ns:return>
</ns:addResourceResponse>
```

... if everything went fine or...

```
<ns:addResourceResponse>
<ns:return>
 errorDescription
</ns:return>
</ns:addResourceResponse>
```

... if something went wrong, where “errorDescription” is the description of the error occurred (typically the trace of the thrown exception).

5.3.2 LearnWeb issues

5.3.2.1 TENC objects(SU)

Some changes in LearnWeb may be needed in order to:

- Search for specific TENC objects;
- Preview TENC objects;
- List related TENC objects
- Etc.

5.3.2.2 Authorization services(UHANN)

The ultimate goal of Protune is to use natural language for every interaction with the user, i.e.

1. for policy definition
2. for answering policy-related questions (e.g., for explaining the decisions taken by the Protune engine)
The first item is a current research issue. So far we support relatively complex policies like the following one.

Every user who sends a credential
- that is valid and
- whose type is “creditCard” and
- whose owner is authenticated and
- on which a price is charged
pays the price with “creditCard”.

Nevertheless further investigation is needed to find the ways to support users in the task of defining policies (and especially to constrain them to only define – syntactically – correct policies).

The second item has been already addressed: the Protune framework supports HOW-TO, WHAT-IF, WHY and WHY-NOT natural-language explanations which allow a user to get information about (respectively)

- which actions she has to perform in order to access some resource
- how the Protune engine would behave assuming a given user behaviour
- why the Protune engine allowed her to access some resource
- why the Protune engine did not allow her to access some resource

An example WHY-NOT explanation follows.

I cannot prove that it is allowed to retrieve “myFile” because:
1. Rule [0] cannot be applied:
   - it is not the case that “myFile” is a public document [details]
2. Rule [1] cannot be applied:
   - “myFile” is a protected document [details]
   - entities certified by Visa can access “myFile” [details]
   but
   - it is not the case that a credential with issuer Visa has been received. [details]

The “details” tags are links to more fine-grained explanations, so that the explanation tree can be navigated by clicking on them. E.g., by clicking on the first “details” tag a new page opens showing why “myFile” is not a public document.
5.4 Integration with TenTube (Giunti + Insead)

TenTube will accept the keywords from the user and then activate the LearnWeb Search Web Service that would provide a list of accessible resources. Some of those resources for example video resources would be imported or dynamically linked to TenTube. We would be modifying the video player in order to play the video which are hosted on LearnWeb.

TenTube will also provide the web service for searching the videos from its own repository. The search web service will accept the input search parameters using query string and will return the results in the xml format. Using the search web service, LearnWeb will also be able to dynamically link the video which are hosted on TenTube.

5.4.1 KRService issues
The KR services will make a additional call to TenTube search webservice for searching the videos which are present in TenTube.

5.4.2 LearnWeb issues
The LearnWeb Search page will be returning the results from TenTube. When clicking on a resource from TenTube - the video will be previewed just as the videos from YouTube so code of the Preview page need to be enhanced accordingly.
6 Social functionalities

6.1 Comment rating (Giunti)

Comments are very important to give an added value to resources. Metadata and the resource itself give an abstract description of the media, a comment can add personal view and can allow to see the same resource from different perspectives. Through comments is also possible a kind of communication between users. It is also important for each user to read directly useful comments and discard useless ones. For this reason the idea is to introduce a rating on comments, very similar to resources rating. In this way is very simple for a learner to read only useful information. Comment rating function will be added in comment section in the resource viewer. The user should be able to see the number of positive/negative vote for a comment and can rate comments by pressing up/down buttons showed close to the comment. Only one vote will be allowed.

6.1.1 KRService issues

Comment rating service is already present, but is similar to resource rating. Some modifications are required to meet the different approach. Instead of the average value of votes, KRService need to provide a sum of negative/positive vote.

6.1.2 LearnWeb issues

The Web Application should provide a simple way for a user to rate comments. The best approach is to show the rating value near each comment. It is not necessary for comments to have a scaled value like in resources (from 1 to 5), it will be sufficient to have a yes/no approach; the graphical representation could be something like thumb up/down. Resource page need to be modified. Buttons for rating comment are required. A label showing the rate of the comment needs to be added close to each comment.

6.2 User search (Altran)

LearnWeb is addressed only to Resources and not to Users, because there exist other specific tools allowing user search: TenTube, OverviewTool.

Therefore LearnWeb will not allow user search in an interactive way. Nevertheless a KRService exists for getting user information from PCM URI. This service could be useful for other tools, not for LearnWeb.
6.2.1 KRSERVICE issues
The KRSERVICE for user search is already existing, so nothing has to be implemented.

6.2.2 LearnWeb issues
LearnWeb will not allow user search. It will done by WP8 tools (TenTube, OverviewTool)

6.3 Managing abuses (Giunti)
Abuses are managed at resource level and are specific of a single tag or comment or metadata. Each registered user can report an abuse by clicking a specific button in the preview page and filling a form precompiled with the resource name. The abuse is managed sending an email to the administrator, whose address is stored in a specific table.

The below figure shows the “abuse” button:
6.3.1 KRSservice issues
Abuses are not managed through KRSservice and are not stored in Fedora.
Abuses consists in simple email sent by a LearnWeb php page.
Abuses in other tools (PCM, ReCourse,...) will be managed by the tools themselves.

6.3.2 LearnWeb issues
The only page where the abuse button will be present is the preview page.

6.4 Stimulating users (Giunti)
As already highlighted in ID5.12, LearnWeb is useful only if crowded of comment/tags/rates having a high quality. To achieve this, users should be stimulated for becoming:

- a frequent user
- a good user

The stimulus may be created by giving users gratitude expressed as membership to an excellence class.
The user quality is expressed by mean of 5 classes, represented by an icon aside each occurrence of the user photo or name. Classes are:

1. newbie, represented by or similar
2. member, this is considered “normal” hence no icon is associated to this class
3. silver member, represented by or similar
4. golden member, represented by or similar
5. unfair user, obscured and his resources are hidden

The below figure is an example about how to visualize the membership class.
The class is assigned to a user in the following way:

- **class 1 (newbie):** everybody is in class 1 after registration
- **class 2 (member):** a user enters in class 2 when he has:
  - uploaded/added at least 1 resource (this number is parametric as well as the numbers appearing below)
  - commented at least 1 resource of another user
  - rated at least 1 comment of another user
  - rated at least 1 resource of another user
  - tagged at least 1 resource of another user
  - his feedback balance is above zero. The feedback balance of a user is the sum of all rates received by other users about all his comments
- **class 3 (silver):** a user enters in class 3 when he has:
  - uploaded/added at least 5 resources
  - commented at least 5 resources of another user
  - rated at least 5 comments of another user
  - rated at least 5 resources of another user
  - tagged at least 5 resources of another user
  - his feedback balance is at least 5
- **class 4 (golden):** a user enters in class 4 when he has:
  - uploaded/added at least 10 resources
  - commented at least 10 resources of another user
  - rated at least 10 comments of another user
  - rated at least 10 resources of another user
  - tagged at least 10 resources of another user
  - his feedback balance is at least 10
- **class 5 (unfair):** the belonging to this class is not automatic. A user enters in this class from the authority of the System Administrator. This is generally applied when a user uploads several resources that are considered abuses from several other users.

All the numbers shown above are parametric and stored in a specific configuration file.
6.4.1 KRService issues

No specific web-service will be implemented for this functionality. However Fedora should store the following information for each user:

- A counter of the uploaded-added resources. This counter is incremented at the end of the upload/add function
- A counter of the comments on resources of other users. This counter is incremented at the end of the commenting function
- A counter of the rates on comments of other users. This counter is incremented at the end of the comment-rating function
- A counter of the rates on resources of other users. This counter is incremented at the end of the resource-rating function
- A counter of the tags on resources of other users. This counter is incremented at the end of the tagging function
- The feedback-balance. This balance is incremented/decremented at the end of the comment-rating function
- A counter of the abuses received from other users. This counter is incremented manually by the system administrator
- The membership class. This value is recomputed whenever one of the above counters change.

The web-service returning the user photo should return the user class as well, in order to allow the proper icon visualization.

6.4.2 LearnWeb issues

All the pages where a user photo or name is shown must be modified for adding the icon of his membership class.

7 Conclusions (Giunti)

The above presented functionalities have been well discussed and weighted both from a technical point of view and from the perspective of the usage scenarios.

The described features meet the three main objectives of WP5 in DIP-4:

1. Finalize the web tool and web services with complete functionalities
2. Empowering quality of resources through social functionalities (rating, rewarding, immersive collaboration)
3. Integration of all WP5 tools in TENCompetence framework, limiting the aims to the most important issues and feasible within the timeframe defined for implementation of ID5.16, i.e. LearnWeb v.0.3..

All the remaining functionalities will be targeted in the final version of the system (ID5.21), and described in the final version of the requirements (ID5.20).
8 References


[2] ID5.13 (new requirements), delivered on 28-11-2008,
http://hdl.handle.net/1820/1607

http://docs.google.com/View?docid=ddhtqnsh_663c8h7vtfs

http://docs.google.com/Doc?id=ddsnp3t6_28hcxmxphf


9 Appendix A

9.1 Protune (UHANN)

The Protune policy framework will be used to enforce authorization on top of the TENCompetence infrastructure.

The Protune framework allows users to define and enforce policies. An example authorization policy (in natural language) follows.

John is allowed to read “myFile” each day from 9:00 to 17:00.

While not that meaningful, the example above helps in identifying the different elements a generic policy consists of

**Actor:** the entity (not) allowed to perform some action on some resource (e.g., “John”)
**Action:** which kind of access some actor has (not) on some resource (e.g., “read”)
**Resource:** the entity some action is (not) allowed to be performed on (e.g., “myFile”)
**Environmental constraints:** further conditions which do not depend on actor, action or resource (e.g., “each day from 9:00 to 17:00”)

Protune policies do not need to explicitly mention actor, action, resource and environmental constraints since the Protune language allows to identify them by means of their properties, like in the following example.
Every authenticated user is allowed to access the files contained in “myDirectory” according to her access rights and only during working time.

Again this policy involves actors, actions, resources and environmental constraints, but this time they are not referenced by name but by their properties

**Actor:** every authenticated user  
**Action:** the actions the user has rights for  
**Resource:** the files contained in “myDirectory”  
**Environmental constraints:** during working time

Finally notice that, as the item “action” shows, the expressiveness of the Protune language allows to define complex conditions involving not only either of actors, actions, resources and environmental constraints but also all of them (since access rights differ from person to person and from resource to resource).

In order to enforce the policy mentioned above the Protune engine must (of course) be able to get information about

- whether a user has been authenticated  
- which rights a user has on a resource  
- which are the files contained in “myDirectory”  
- what “working time” means

Retrieving this information may require to perform some actions (e.g., in order to find out which files are contained in “myDirectory” the Protune engine must access the file system). The Protune framework provides a means to smoothly plug actions into it.

The main authorization-related tasks of L3S within TENCompetence are

1. to identify and implement the actions which the TENCompetence scenarios require to support  
2. to provide a user-friendly natural language-based interface for every interaction with the user  
3. to make such services available within the TENCompetence infrastructure

The first point requires support from other TENCompetence partners in order to

- identify the scenarios which need to be supported and extract the requirements from them  
- eventually implement the corresponding actions the Protune engine needs to be aware of

The current state of point 2. (resp. 3.) will be reported in Section 5.3.1.1(resp.5.3.2.1).