Collaborative Project

Responsive Open Learning Environments

European Commission Seventh Framework Project (IST-231396)

Deliverable 4.2

Service Syndication Platform

Editor	*Daniel Dahrendorf*
Work Package	*WP 4*
Status	*Final*
Date	*29 July 2010*

The ROLE Consortium

Beneficiary Number	Beneficiary name	Beneficiary short name	Country
1	Fraunhofer FIT	FHG	DE
2	RWTH Aachen University	RWTH Aachen	DE
3	Technical University of Graz	TUG	AT
4	Katholieke Universiteit Leuven	K.U.LEUVEN	BE
5	University of Koblenz	UNI KO-LD	DE
6	Uppsala University	UU	SE
7	École Polytechnique Fédérale de Lausanne	EPFL	CH
8	University of Leicester	ULEIC	UK
9	Open University UK	OU	UK
10	Vienna University of Economics & Business Administration	WW	AT
11	Festo Lernzentrum Saar GmbH	FESTO	DE
12	imc AG	IMC	DE
13	British Institute for Learning and Development	BILD	UK
14	Shanghai Jiao Tong University, China	SJTU	RPC
15	Zentrum für Soziale Innovation	ZSI	AT
16	U&I Learning	UIL	BE

Document Control

Title:	**Service Syndication Platform**
Author/Editor:	**Daniel Dahrendorf**
E-mail:	**daniel.dahrendorf@im-c.de**

Amendment History

Version	Date	Author/Editor	Description/Comments
0.1	06.05.2010	Daniel Dahrendorf	First draft
0.2	20.05.2010	Daniel Dahrendorf	Integration of the feedback of Nils Faltin based on version 0.1
0.3	08.06.2010	Dr. Nils Faltin, Daniel Dahrendorf	Update of the document structure, added widget preferences
0.4	17.06.2010	Evgeny Bogdanov, Daniel Dahrendorf	Introduced additional recommendations and widget preferences framework
0.5	28.06.2010	Daniel Dahrendorf	Final draft for the peer-review
0.6	22.07.2010	Daniel Dahrendorf	Integrated the feedback of the first peer review from Denis Gillet and Alexander Nussbaumer Integrated the feedback from Martin Wolpers
0.7	22.07.2010	Dr. Volker Zimmermann	Update of positioning and strategy with syndication platform within ROLE
1.0	29.07.2010	Daniel Dahrendorf	First final version

Contributors

Name	Institution
Dr. Volker Zimmermann, Dr. Nils Faltin, Daniel Dahrendorf, Jens Peters	IMC AG
Evgeny Bogdanov	EPFL
Martin Wolpers	FIT

Reviewer

Name	Institution
Alexander Nussbaumer	TUG
Denis Gillet	EPFL

Executive Summary

This deliverable is the documentation of the major research work and outcomes of tasks 4.1 and 4.2. It describes the design and the implementation plan for the components of the service syndication platform, which is based on a systematic requirements analysis and functional specification of the different components.

The “service syndication platform” consists of four components, as described already in the deliverables D3.3 and D4.1:

- The ROLE Widget Store,
- the ROLE Widget Repository,
- the ROLE Widget Preferences Server, and
- the ROLE Space,

These four components together enable a highly user-oriented customizability of the learning environment according to personal or collaborative needs. It is the core technology of a responsive environment according to the project’s definition. All the components of the syndication platform use open standards for interoperability.

This document puts the focus on the ROLE Widget Store. The ROLE Space and links to the Preferences Server will be specified in the deliverables 4.3 “PLE implementations for testbeds” and 4.5 “Generic PLE implementation”, as the ROLE space is one option for end users to create their responsive environment based on recommendations and personal preferences.

This document provides a requirements analysis and an implementation plan for the ROLE Widget Store and related components. To reach this goal, a common vocabulary is created extending the work done in deliverable 3.3 and 4.1 and the dependency between the different components is illustrated.

The deliverable provides also use cases for the service syndication platform. These use cases include the definition and selection of services (widgets) and the aggregation into sets (widget bundles). Reposting, rating, searching and recommendation are key features in the use cases. Further a quality management for widgets is introduced. As an analysis of existing widget stores revealed, none of these platforms match the ROLE use cases.

The use cases are the basis for detailing the requirements and discussion of possible technical solutions. This includes different levels. Possible widget specifications are listed and a first specification is selected. Authentication and basic store features are discussed and a first approach for quality assurance is suggested. For recommendation different approaches which will be included in the store are described. A mechanism for shared widgets is discussed in the widget preferences section. The Interoperability is taken into account by defining interfaces of the ROLE Widget Store and introducing a framework for controlled access to widget services. Also first Mock-ups of the widget store are provided and a last section describes possible additional features which still need to be discussed with the developer community.

An agile implementation approach is used to implement the ROLE Widget Store which allows reacting on possible changes of the requirements. This will also allow extending the widget store with new features like the recommendation engine and the preferences server.

According to the project definition, one main expectation of this approach is: “The ROLE framework will create a market for small software components, augmenting software from big vendors. That will provide, in particular, new opportunities for SMEs”. For that reason the ROLE Widget Store is designed for multi vendor support to provide possibilities for developer to sell

their widgets. As part of the evaluation of the whole system prepared in WP 6 and done by WP 5 the acceptance of the payment system by users will be tested.

The implementation process is planned in several iterations, as described above. Therefore, this is the first version of this deliverable. After each iteration the requirements in this document will be updated and the progress of the implementation will be documented.

Table of Contents

1 The Service Syndication Platform

The idea of the **Service Syndication Platform** is to create a platform for end users where they can

- choose personal learning services according to their learning needs and personal profile out of a repository,
- administrate the selected services in order to use them according to their personal preferences, and
- aggregate or syndicate single services with others in order to create a "larger" service bundle (syndication technology, selection of bundles).

Within ROLE, the term "widget" has been established for services that fulfil the ability to be used within a learning platform or personal learning environment and are being able to be syndicated using mash-up technologies. Therefore the ROLE service syndication platform is composed out of four components:

- The ROLE **Widget Store** (as main platform component for end users to search services and bundle them based on recommendations. The name of the component reflects the market point of view of learning services),
- the ROLE **Widget Repository** (as the community database of all services and service bundles that can be delivered from different software partners and the learning technology community which will be integrated in the ROLE Widget Store),
- the ROLE **Widget Preferences Server** (as a platform component which allows to store service preferences and enable sharing of services and service bundles), and
- the ROLE **Space** (as personal learning mash-up environment for end users).

This deliverable (related to tasks 4.1 and 4.2) presents the concept of the ROLE syndication platform with focus on the ROLE Widget Store. A detailed description of the ROLE Space and the ROLE Preference Server will be given in deliverable 4.3 and 4.5 as these are more linked to the implementation side of a personal learning environment.

The term "widget" is in the following being used equivalent to "service" or "learning service" (please see below the chapter on definitions).

1.1 Scope

This document addresses the following topics:

- Detailed overview over the widget architecture and description of existing widget stores
- Use cases for the store and a requirement analysis
- Description of the implementation process

The description of work (DoW) marks this deliverable as continuously updated until M48. Thus this document is a living document which will be changed if needed as the requirements and technical specification could be changed during the implementation.

1.2 Requirements on the Service Syndication Platform from the Description of Work

From the DoW the following requirement where extracted from task 4.3 for the service syndication platform:

- "Users can describe individual services (PLS) as well define and administrate sets of services that should be syndicated together"

- “A user that defines a set of syndicated Services must also specify other properties such as which kind of participation is allowed, for example open invitation or restricted membership”
- “… the platform will provide support for search, compatibility assessment, state educational value (rate) and mashing-up of the services by using an intelligent mediation and recommendation engine …”
- “… enable the reposting and tracking of adapted, contextualized or repurposed PLS”
- “develop a client side library that simplifies the client side integration of sets of syndicated services”

and from the DoW also stated as general requirement:

- "... The ROLE framework will create a market for small software components, augmenting software from big vendors. That will provide, in particular, new opportunities for SMEs."

These requirements are the base for the use cases in section 3 and are integrated in requirement analysis in section 5.

1.3 Why a Widget Store as Part of the Syndication Platform is Needed

When designing the ROLE syndication platform and having identified that the core component was a services marketplace or “widget store” is needed, different marketplace technologies, shop systems and widget stores have been analyzed and compared in a first step. The results are described in details in section 4. None of the existing stores match the use cases listed in section 3. Main points of the critique were:

- No Store provides recommendations for learning/studying processes,
- existing stores do not aim at providing education widgets or host only a very limited amount of education widgets,
- existing stores often do not provide quality management,
- no web widget store provides payment functionality.

The ROLE project thus needs to create a platform component which fills these gaps. As part of the Service Syndication Platform, the ROLE Widget Store will serve as an open widget repository where learners can get recommendations for education related widgets and get support for creating their own Personal Learning Environment (PLEs). Widget developers will be attracted to submit widgets into the store where the quality is ensured by the community. It will be a place where best practices of PLE will be exchanged and discussed. PLE Developers will be able to connect via provided APIs to the ROLE Widget Store and add ROLE based features to their PLEs.

2 Definitions and Technical Baseline

The following definitions are as general as possible but are also compatible with definitions from previous deliverables.

2.1 Definitions for PLE and Widget Related Terms

Channel

A channel is a listing in a widget store which contains widgets belonging to one developer or one topic (e.g. English learning). A channel is managed by users in contrast to a category which summarizes related widgets. All widgets from the channels are also displayed in the general widget store catalogue.

Personal Learning Environment (PLE)

Personal Learning Environment (PLE) is an environment that helps learners take control of and manage their own learning. PLE normally comprises more than one device or platform and includes learning resources in electronic as well as non-electronic format. PLE enables a learner to create and maintain a user profile, which facilitates them to set learning goals, identify resources, and communicate as well as collaborate with people in a community of interest or practice. It also enables the learner, with or without the use of tools, to evaluate their learning progress using feedback on their work from abler others, peers and personal reflection. PLE also supports the delivery of recommendations from trustful sources to learners with respect to their dynamic profile and goals. [34].

Personalized Learning Management System (PLMS)

A Personalized Learning Management System (PLMS) combines the “traditional” learning management system (LMS) approach with personal learning environments (PLE). The aim of PLMSs is to fulfill business requirements. (LMS + PLE = PLMS) [36].

Widget

Widgets are PC or mobile micro-applications [18]. In Web 2.0 this term is used for small independent programs displayed in a window [17]. A widget container is needed to render and execute a widget. Some Widgets are able to communicate via the widget engine with the widget container on which the widgets are running. For the developing of widgets often web technologies like DHTML, JavaScript, Flash and Ajax are used.

Within ROLE, the term “widget” has been established for services. A learning service can be implemented as learning widget.

Widget Bundle

A widget bundle is a set of widgets, which is prepared by a user and stored e.g. in a widget store. A bundle can contain layout information and (default or modified) widget preferences for the contained widgets. Further the owner of a widget bundle can share the bundle with the public or with specific users. A widget space which is running in a container can be saved as widget bundle and a widget bundle can be used to create a widget space in a container.

Widget Container

A widget container provides a well defined environment for executing, view management, communication, etc. It uses the widget engine to render the widgets. Further, it might also take the form of a more user friendly environment where the learner can organize widgets visually, set preferences, contact the widget store for additional widgets, etc. (see Deliverable 3.3).

Widget Engine

A widget engine is a server application that generates embeddable widget instances by combining widget definitions from a widget repository with widgets preferences from a widget preference server. The widget engine may also expose additional services useful to widgets, for example proxy functionality for accessing remote services and support for authentication/authorization (see Deliverable 3.3).

Widget Instance

A widget instance is a widget "put at work" into a widget engine for a single user or a group of users; in particular it is associated with widget preferences and/or state data (see Deliverable 3.3).

Widget Kit

A widget kit is simple set of widgets which does not contain any other information.

Widget Preferences

Widget Preferences are used to configure widget instances. These preferences are stored at a widget preferences server, or at other locations selected by the users for privacy issues. They can be changed at runtime by the users and default values can be provided.

Widget Preferences Server

The widgets located in the widget repository are tied with widget preferences that have default values. When a widget is instantiated or later modified by a widget engine (on behalf of a user), those preferences can be overridden. A common approach is to store the widget preferences in the widget container (e.g. OpenSocial [27]). By storing these widget preferences in a widget preference server (separate from the widget engine) one achieves portability and sharing (see Deliverable 3.3).

Widget Repository

A widget repository is a physical storage that hosts the code of widgets. The widgets in the widget repository cannot be executed directly. They must be rendered with a widget engine to be instantiated into a widget instance. Strictly speaking, a widget repository is not needed, e.g. the iGoogle content directory just accepts any URL that points to a Google gadget XML file (see Deliverable 3.3).

Widget Space

A widget space is made of several widgets which are combined together, either graphically and/or through some kind of data-flow/event wiring. A widget space is always running in a widget container and can be an instance of a widget bundle. Thus it contains also layout information and widget preferences. A space can be used by only one user or if the container supports it the space can be shared and used by selected users. In current widget containers (e.g. iGoogle, Netvibes, Pageflakes) this is done by putting widgets in different tabs. A space can correspond to a course, a project etc. (compare [40]).

Widget Space Configuration

A widget space configuration can describe how to compose widgets of a space (layout, graphical theme, dataflow, etc.). The configuration also include widgets, the widgets preferences (default or modified) and users of the space (including their rights) [40].

Widget Store

A widget store should provide a user friendly interface to a widget repository that simplifies widgets discovery by using both basic metadata of the widget and potential recommendations from other users. The functionality of the widget store includes listings of widgets, categorizations, and search by keyword, etc. Further users can add widgets from a widget store to supported PLE systems. From a social media point of view, a store could also be the place to collect and share user tags, comments and ratings.

Today, there are several widget stores available (see Deliverable 3.3). There will be no strict distinction between web and desktop widget as the line between them is blurred (consider Air [1] or Apple Dashboard [6] widgets).

2.2 Technical Baseline for Widgets

Figure 1 shows the interaction between the different components. The illustrated widget architecture generalizes existing systems.

The sources of the widgets are stored in a widget repository. These widgets are cataloged in a widget store. Different providers can submit their widgets to the store where the widgets are listed in provider and topic channels and in the general catalogue.

A user can select widgets from the widget store and add these widgets to her preferred widget container inside a PLE. When the user wants to use the added widgets in her widget container, the widget container sends a render request to the widget engine with the location of the widget which should be rendered.

The widget engine gets the sources (from the widget repository) of the widget and renders the widget in a form the widget container can process. The widget running in the container which is used by the learner is called widget instance.

In the most cases there is no strict separation between the platforms. A common approach is to combine the widget store and the widget repository to a single platform. Further, for widgets which are deployed as a single ZIP file, the workflow will be different as the widget engine does not need to contact the widget repository. Most of the widget container will have an integrated widget engine.

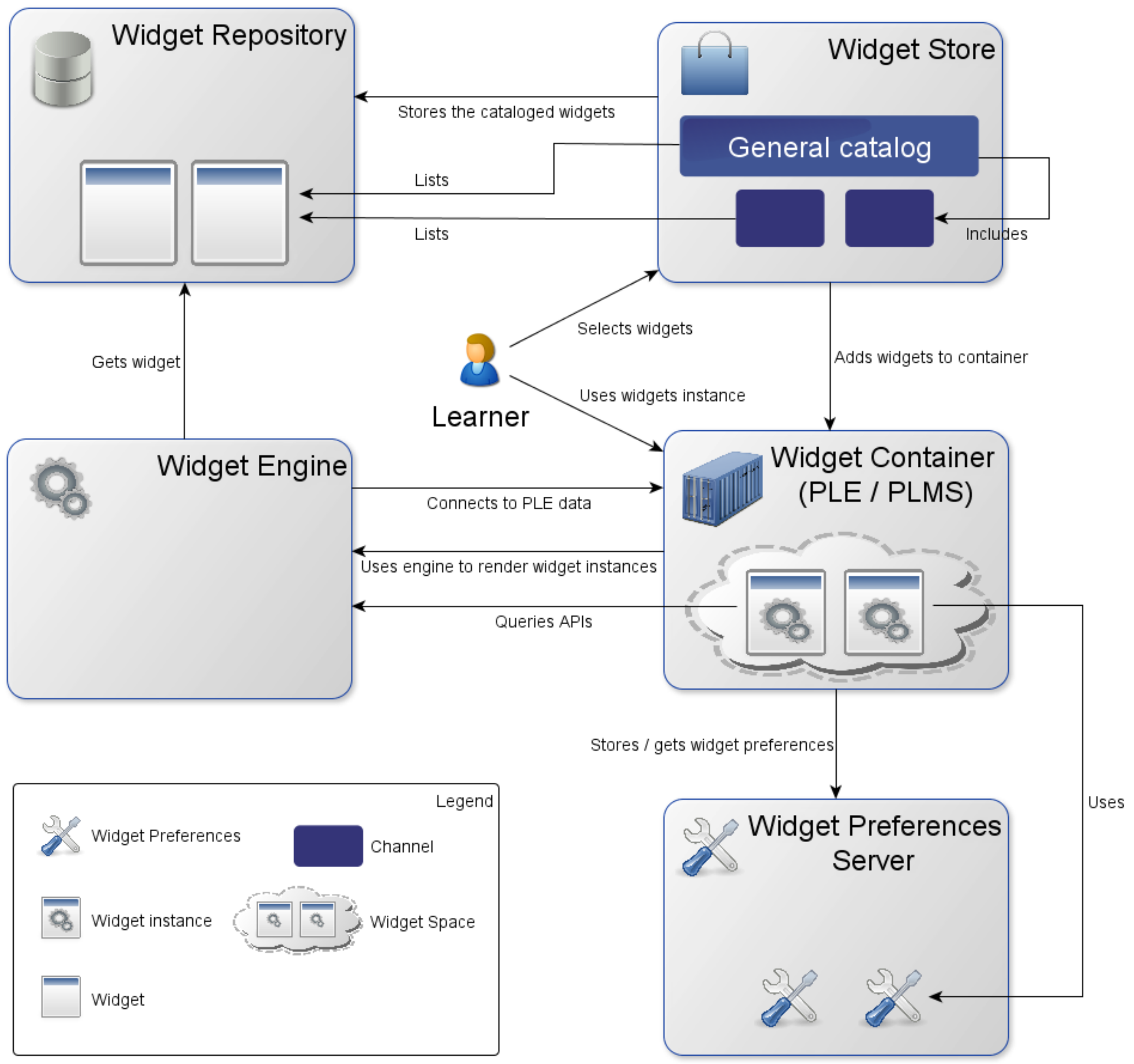


Figure 1: Technical widget architecture

Some widget technologies, like OpenSocial, allow widgets to connect via a defined API to the engine which is connected to the backend of the widget container. OpenSocial widgets are able to connect to the social graph (if existing) of the widget container.

A learner is not limited to use only one widget store as resource of widgets for her PLE. Figure 2 shows the case where a learner adds widgets from different stores to her PLE. The limitation is that the widget store and the PLE need to support the same technology and external widgets can be added to the PLE.

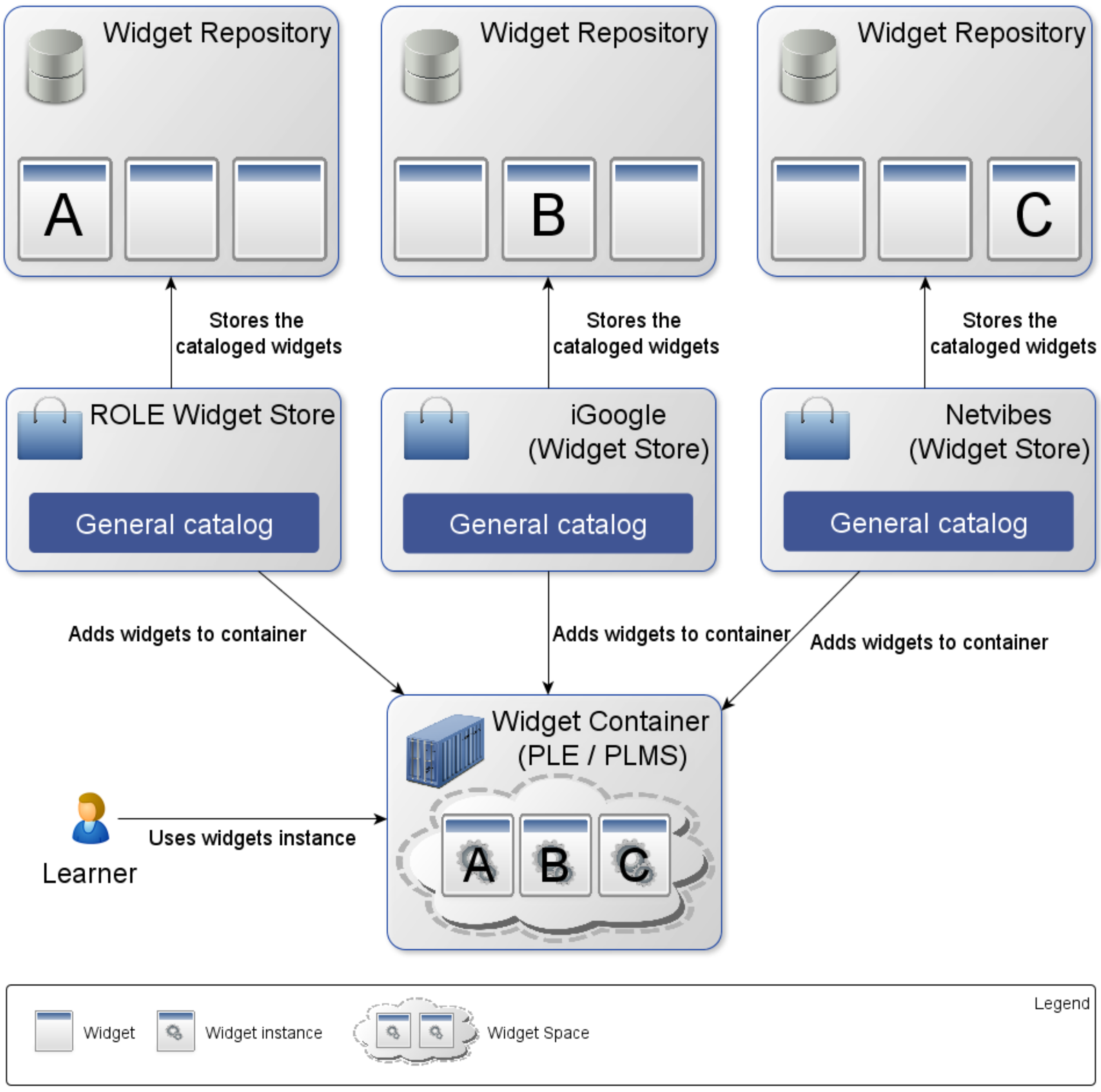


Figure 2: A PLE containing widgets from different widget stores

3 Use Cases

This section describes the use cases for the ROLE Widget Store. The use cases did build the basis for the requirements identification and design process of the syndication platform with focus on the widget store.

For the description process of the use cases a learner is being considered who uses the ROLE Widget Store to add widgets to her PLE. The widgets in the store are submitted by different vendors and passed the quality assurance which is done by quality managers from the community.

The use cases were created based on discussions with the testbeds and the developer community. The use cases are taking the description of the ROLE Widget Store in Deliverable 4.1 and the requirements from task 4.3 into account.

3.1 Actors

The following actors are involved in the use cases:

- Learner: The actor who uses the system to search, obtain and manage her obtained widgets.
- Vendor: The actor who offers widgets in the shop.
- Instructor: The actor which may support learners and prepare learning activities pages
- Quality manager: The actor who judges the quality of submitted widgets and publishes or sends them back for improvement.
- External Widget Store: A widget store which has an API which allows the importing of widgets.
- PLE / PLMS: A Personal Learning Environment system as defined in section 2.1. The learner can add widgets to the PLE and the PLE can access the API of the system.

3.2 Description of the Use Cases

In the following section the use cases supported by the ROLE Widget Store will be described. Figure 3 shows an overview of the existing use cases.

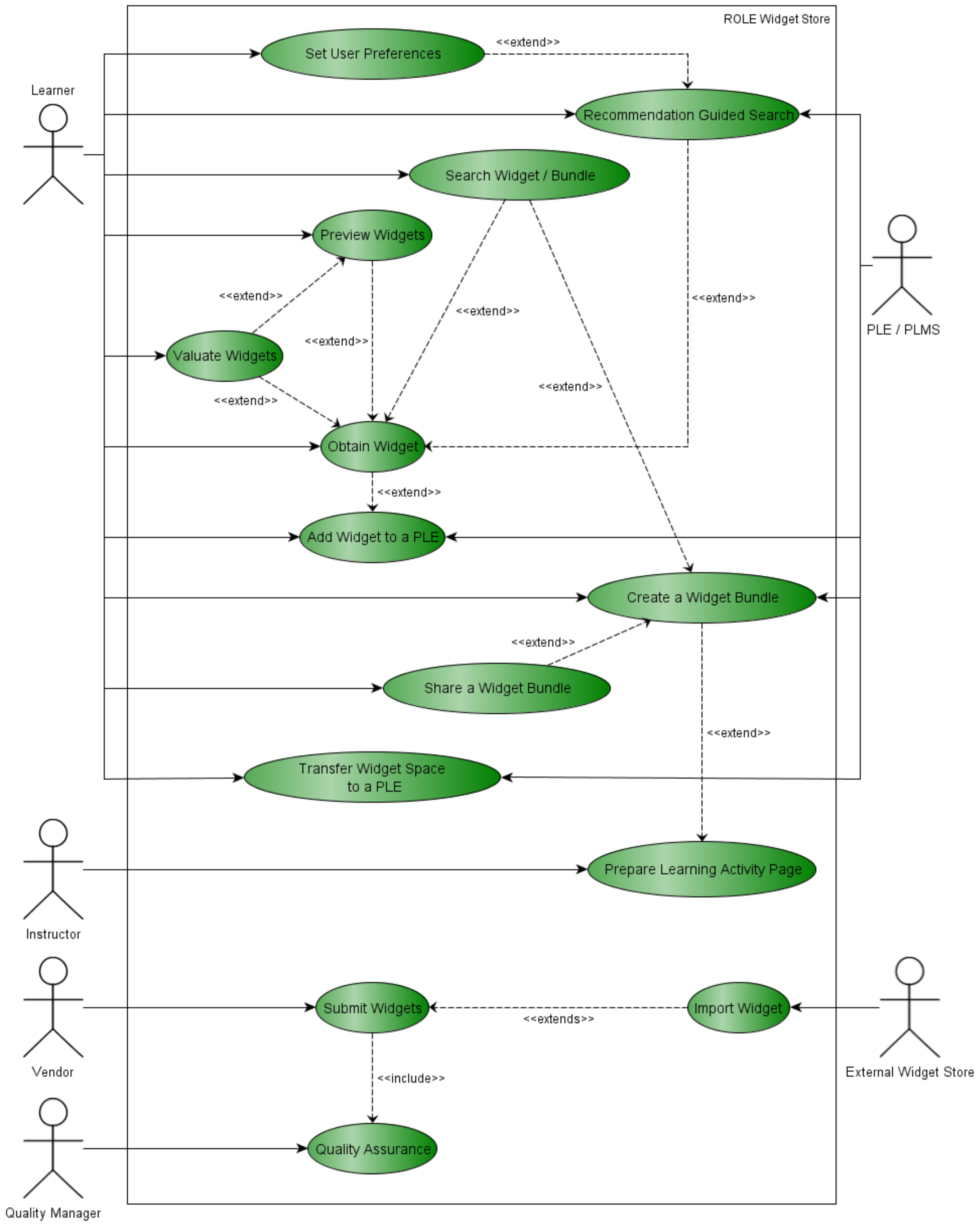


Figure 3: Overview of the ROLE Widget Shop use cases

3.2.1 Use Case: Add Widget to a PLE

3.2.1.1 Preconditions

- The learner has an account and is logged into the ROLE Widget Store.
- The learner has an account at her PLE

3.2.1.2 Postconditions

- The learner will have added a widget to her PLE.

3.2.1.3 Main Scenario

1. The Learner opens her list of already obtained widgets in the store [Alternate Scenario B: Add widget from inside the PLE]
2. The learner selects a widget from her personal widget list [Extension Point: Obtain Widget]
3. The learner chooses a PLE where she wants to add the widget
4. The learner chooses to add the widgets via the “add to PLE button” in the ROLE Widget Store [Alternate Scenario A: Add widget via copying the URL]
5. The system forwards the learner to her PLE
6. The learner logs into the PLE
7. The learner confirms the adding of the widget in the PLE
8. The use case ends

3.2.1.4 Alternate Scenario A: Add Widget via Copying the URL

A1. The learner chooses to add the widget by copying the URL to her PLE

A2. The learner copies the URL to her system clipboard

A3. The learner logs into the PLE

A4. The learner uses the PLE mechanism for adding a widget via URL

A5. The use case ends

3.2.1.5 Alternate Scenario B: Add Widget from Inside the PLE

B1. The Learner logs into her PLE

B2. The Learner opens the list already obtained widgets from the store

B3. The PLE queries the system via the API which widgets are available for the user

B4. The system returns the list of obtained widgets from the user

B5. The user chooses a widget and presses the add button

B6. The PLE adds the requested widget

B7. The use case ends

3.2.1.6 Alternate Scenario C: Add Widget via an Aggregation Bookmark

C1. The Learner logs into her PLE which supports aggregation bookmarks

C2. The Learner creates an aggregation bookmark

C3. The Learner logs into Widget Store

C4. The Learner opens a widget that she is interested in

C5. The Learner clicks on the bookmark and presses the add button

C6. The PLE parses the URL of the widget via the aggregation bookmark

C7. The PLE adds the widget

C8. The use case ends

3.2.2 Use Case: Obtain Widget

3.2.2.1 Preconditions

- The learner has an account and is logged into the ROLE Widget Store.

3.2.2.2 Postconditions

- The learner will have obtained a widget.

3.2.2.3 Main Scenario

1. The Learner goes to the home of the widget store
2. The learner visits the catalogue and selects a widget [Extension Point: Search Widgets]
3. The learner reads the description [Extension Points: Preview Widgets, Valuate Widgets]
4. The Learner adds a free widget to her personal widget list [Alternate Scenario A: Buy Widget]
5. The use case ends

3.2.2.4 Alternate Scenario A: Buy Widget

A1. The learner adds a paid widget to her shopping cart

A2. The learner open the payment page

A3. The learner executes the payment process

A4. The learner waits for confirmation

A5. The system adds the bought widget to the personal widget list of the learner

A6. The use case ends

3.2.3 Use Case: Evaluate Widget

3.2.3.1 Preconditions

- The learner has an account and is logged into the ROLE Widget Store.

3.2.3.2 Postconditions

- The learner will have valuated a widget.

3.2.3.3 Main Scenario

1. The leaner opens a widget description page
2. The learner goes to the rating section of the page
3. The learner rates the widgets using the 5 Stars feature [Alternate Scenario A: Comment Widget]
4. The use case ends

3.2.3.4 Alternate Scenario A: Comment widget

A1. The learner comments the widget using the prepared form

A2. The use case ends

3.2.4 Use Case: Preview Widget

3.2.4.1 Preconditions

- None

3.2.4.2 Postconditions

- The learner will have previewed a widget.

3.2.4.3 Main Scenario

1. The learner goes to a widget description page
2. The learner goes to the preview section of the page
3. The learner tests the available functionality of the demo widget [Extension Point: Use Case Valuate Widget]
4. The use case ends

3.2.5 Use Case: Search Widget / Bundle

3.2.5.1 Preconditions

- None

3.2.5.2 Postconditions

- The learner will have selected a widget / bundle.

3.2.5.3 Main Scenario

1. The learner starts the search feature
2. The learner chooses if she want to search for widgets or bundles
3. The learner searches for a keyword [Alternate Scenario A]
4. The system displays the appropriate widgets / bundles in a list
5. The learner selects a widget / bundle
6. The use case ends

3.2.5.4 Alternate Scenario A

A1. The learner starts the extended search where she can choose to search for vendors, categories and tags

A2. The use case continue at step 3 in the main scenario

3.2.6 Use Case: Recommendation Guided Search

3.2.6.1 Preconditions

- The learner has an account and is logged into the ROLE Widget Store.
- The learner has connected her PLE / PLMS to the ROLE Widget Store

3.2.6.2 Postconditions

- The learner will get several widget recommendations.

3.2.6.3 Main Scenario

1. The learner starts the recommendation tool [Extension Point: Use Case Set User Preferences; Alternate Scenario A: Get Recommendation in the PLE]
2. The learner chooses the type of recommendations
3. The system displays a list of recommended widgets / bundles
4. The use case ends

3.2.6.4 Alternate Scenario A: Get Recommendation in the PLE

A1. The learner logs into her PLE

A2. The learner uses her PLE

A3. The PLE is querying the store for recommendations via the API and is transferring the context of the PLE

A4. The learner sees some recommended widgets / bundles for his context

A5. The use case ends

3.2.7 Use Case: Create a Widget Bundle

3.2.7.1 Preconditions

- The learner has an account and is logged into the ROLE Widget Store.

3.2.7.2 Postconditions

- The learner will have created a widget bundle

3.2.7.3 Main Scenario

1. The learner goes to the widget bundle page [Alternate Scenario B: Create a Widget Bundle Inside a PLE]
2. The learner chooses to create a new widget bundle [Alternate Scenario A: Clone a Widget bundle]
3. The learner chooses to add a widget from her personal widget list to the widget bundle [Extension Point: Use Case Search Widget / Spaces]
4. The learner adds widget to the widget bundle
5. The learner adjust the widget preferences [Extension Point: The learner shares the widget bundle]
6. The use case ends

3.2.7.4 Alternate Scenario A: Clone a Widget Bundle

A1. The learner open a shared widget bundle Extension Point: Use Case Search Widget / Bundles]

A2. The learner chooses to clone the widget bundle

A3. The system creates a clone where the learner is the owner

A4. The use case continue at step 3 in the main scenario

3.2.7.5 Alternate Scenario B: Create a Widget Bundle Inside a PLE

B1. The learner logs into a PLE which supports widget spaces and widget bundles

B2. The learner organizes his widgets in a space

B3. The learner chooses to store the configured widget space as widget bundle in the ROLE Widget Store

B4. The PLE requests the system via a API to store the widget space as widget bundle

B5. The Learner logs into the system

B6. The system stores the widget bundle

B7. The use case end

3.2.8 Use Case: Share a Widget Bundle

3.2.8.1 Preconditions

- The learner has an account and is logged into the ROLE Widget Store.
- The learner is the owner of a widget bundle

3.2.8.2 Postconditions

- The learner will have shared the widget bundle

3.2.8.3 Main Scenario

1. The learner goes to the widget bundle page
2. The learner opens a widget bundle from her widget space list
3. The learner chooses to make the bundle readable for all learners [Alternate Scenario A: Choose User for Write Rights for a Bundle]
4. The use case ends.

3.2.8.4 Alternate Scenario A: Choose User for Write Rights for a Bundle

A1. The learner opens the list of her buddies

A2. The learner chooses buddies which will get write rights for the bundle

A3. The use case ends

3.2.9 Use Case: Transfer a Widget Bundle to a PLE

3.2.9.1 Preconditions

- The learner has an account and is logged into the ROLE Widget Store.
- The learner has a account at a PLE which support the creation of widget spaces by widget bundles

3.2.9.2 Postconditions

- The learner will have created a widget space in her PLE

3.2.9.3 Main Scenario

1. The learner goes to the widget bundle page
2. The learner chooses a widget bundle
3. The learner transfer a widget bundle to the PLE
4. The learner logs into the PLE
5. The PLE creates the widget space from the widget bundle
6. The use case ends

3.2.10 Use Case: Set User Preferences

3.2.10.1 Preconditions

- The learner has an account and is logged into the ROLE Widget Store.

3.2.10.2 Postconditions

- The learner will have set her user preferences

3.2.10.3 Main Scenario

1. The learner goes to the user preferences page
2. The learner sets her user preferences
3. The use case ends

3.2.11 Use Case: Prepare Learning Activity Page

3.2.11.1 Preconditions

- The teacher has an account and is logged into the ROLE Widget Store.

3.2.11.2 Postconditions

- The teacher will have created a learning activity page

3.2.11.3 Main Scenario

1. The teacher goes to the create a learning activity page
2. The teacher opens a widget bundle [Extension Point: Use Case Create a Widget bundle]
3. The teacher sets a user limit and a validity period
4. The teacher invites the learners to use the learning activity page [Alternate Scenario A: Include Learning Activity page to a PLMS]

5. The learner enters the prepared course page in her PLE / PLMS system (The learner does not need to have ROLE Widget Store Account)
6. The learner enter her name into the widget to activate it
7. The learner is using the widget
8. The use case ends

3.2.11.4 Alternate Scenario A: Include Learning Activity Page to a PLMS

A1. The teacher includes the learning activity page into a PLMS

A2. The use case continues at step 5 of the main scenario

4 Related Systems and Case Studies

4.1 *Widget Stores*

This chapter describes selected widget stores which are available on the internet and compares the functionality of them. The focus lies on stores which are targeting web widgets. The Apple Appstore which provides apps for the iPod / iPhone / iPad platform is shown as an example store which has payment functionality.

4.1.1 Appstore

The appstore from Apple [8] does not provide web widgets but sells apps for the iPhone / iPad / iPod framework. As there is no platform where web widgets are sold, the appstore is added as prominent store example for selling small applications. There exist similar approaches from other mobile phone companies. The appstore provides around 140.000 apps and has a strong quality assurance. A developer gets 70% of revenues from the store.

4.1.2 iGoogle

The personalized starting page from Google [15] is using the open Google Gadget / OpenSocial specification for its widgets. In the iGoogle directory a user can get gadgets which are categorized and rated from the community. Also information about the number of installations is provided. The iGoogle page provides the user with a simple recommendation of other widgets. There exist many clones of different widgets and iGoogle has no quality assurance. The whole service is free for users.

4.1.3 Netvibes

More than 180 000 widgets are available for the personalized starting page Netvibes [23]. The user has private and public pages and can connect her pages to other users. The user is only allowed to add the widgets which are hosted at Netvibes to her page. The widgets are categorized and rated. All users can comment and tag the widgets. It provides for businesses three different services. First a business can add a widget to the store by a cost per install model. The second service is that businesses are able to create their own branded dashboards. Further Netvibes offers to create enterprise spaces either as intranet, extranet or portal solution. The Netvibes widgets are based on the UWA API and uses JavaScript/Ajax and JSON. The API also provides OpenSocial support. Netvibes also provides an ecosystem API [22]. Using the Netvibes ecosystem [21], a user is able to export the Netvibes widgets to different platforms (iGoogle, Apple Dashboard, Opera, Windows Vista and Windows Live). There are no costs for the user to use Netvibes.

4.1.4 Springwidget

Springwidget [41] provides an own widget platform for windows where the user can add the provided widgets to her desktop. Another possibility is to add widgets per HTML embedded code in blogs or profile pages of social networks. The widgets are flash based and a developer kit is provided by Springwidget. Each submitted widget will be reviewed and afterwards published in the widget store. There is no payment system available.

4.1.5 Widgetbox

Widgetbox [45] is a widget store with the feature to create widgets via a web interface. The user can create either a widget from the scratch by using flash files, HTML pages or existing Google Gadgets or use a assistant which creates widgets based Twitter [42], Youtube [51], etc. content.

If the user has a premium account, she is able to create pro widgets like polls and slideshows and gets add free widgets. The widget store can be browsed by tags and the widgets can be rated, customized and exported to Facebook, Twitter, Ning, iGoogle, etc.

4.1.6 Widgetop

The personalized page Widgetop [46] is a combination of a widget store and a container. It provides 290 different own widgets without any fee and the user has the possibility to add Google gadgets, YourMinis Widgets and RSS Feeds to the personalized starting page of Widgetop. They use HTML, CSS, JavaScript and Flash based on the Apple Developer Guidelines [7] as technologies. Widgets are submitted to a review before they are added to the page.

4.1.7 XING

The business orientated social network Xing [48] does not aim to be a widget store. However it hosts 18 widgets based on the Open Social technology which can be used by members to add functionality to the core system. Xing does not allow to add any others widgets than the own hosted ones thus it has a high quality assurance. The widgets are usable without any fee.

The following table compares the different widget repositories.

	appstore	iGoogle	Netvibes	Springwidget	Widgetbox	Widgetop	Xing
#widgets	140.000 (apps)**	59875 (over netvibes)*	180.000*	4771*	197,577*	290*	18*
Tagging system	Yes	No	Yes	Yes	Yes	No	No
personalized page	No	Yes	Yes	No	No	Yes	Yes
SDK	Yes	No, but developer tools	No	Yes	No, but Webcreator which creates widgets from flash, HTML and Google Gadgets	Apple Dashboard	No
API	Yes	Google-Gadget, OpenSocial	UWA API, REST API, Ecosystem API, OpenSocial API (integrated in UWA)	Yes, for Flash 7+	No	Apple Dashboard (requires changes)	Google-Gadget, OpenSocial
Used Standards	No	Google-Gadget, OpenSocial	Google-Gadget, OpenSocial	No	Google-Gadget, OpenSocial	Google-Gadget, OpenSocial	Google-Gadget, OpenSocial
Technology		JavaScript, AJAX, Flash	JavaScript, AJAX, Flash	Flash	JavaScript, AJAX, Flash	HTML, CSS and JavaScript	JavaScript, AJAX, Flash
Categorization	Yes	Yes	Yes	Yes	No	Yes	No

	appstore	iGoogle	Netvibes	Springwidget	Widgetbox	Widgetop	Xing
Quality Assurance	Yes	No	?	Yes, but seems to accept any widget	No	?	Yes, strong
Rating	Yes	Yes, 5 Stars, Number of Votes	Yes, 5 Stars, Number of Installations	Yes, 5 Stars, Number of Votes	Yes, 5 Stars, Number of Installations	No	No
Preview	Yes, Slideshow	Yes	Yes	Yes	Yes	Thumbnail, Screenshot	No
Recommen-dation	Yes	Yes	No	No	Yes, related widgets, widgets from the same developer	No	No
System for selling widgets	Yes	No	No	No	No	No	No
Promoting widgets from companies for money	Yes	No	Yes	No	Yes	No	No
Add foreign Widgets	No	Yes, XML files which are Google Gadgetss	No	No	No	Yes, Google Gadgets, YourMinis Widgets	No
Export	No	Yes, embed code	Yes, iGoogle, Apple Dashboard, Opera, Windows Vista, Windows Live	Yes, embed code	embed code, Facebook, MySpace, Ning, iGoogle, Netvibes, etc	No	No

4.2 Summary of the different Approaches

The different widget stores described in the previous section have different foci. While Xing just provides a few widgets for their own system, Springwidget and Widgetbox provide widgets which can be embedded in other systems. Widgetop, Netvibes and iGoogle act as personal starting page and provide also export functionality (except for Widgetop). The Appstore only provides apps for Apples platforms and does have neither an import nor export functionality.

Nearly all web widget stores support the OpenSocial specification either as export or import functionality. This confirms the importance of this specification.

Searching, categorization and rating are common features in these stores. In contrast nearly no recommendation features are available. Exceptions are recommendations based on related widgets and widgets from the same developer.

Also a common feature is the ability to show previews of the widgets. This makes sense as the most widgets are not based on personalized web services (currently).

The Appstore is the only store where developer are able to sell their widgets but also able to submit free widgets.

As none of the described stores match the use cases described in section 3 the ROLE Project has to develop an own platform which fits the needs of the project.

5 Requirements

Based on the use cases and the aims to be fulfilled with the ROLE syndication platform, the requirements from Deliverable 3.3 and the Deliverable 4.1 were extended in this section. This section does not only repeat these requirements but also discusses possible technical solutions.

5.1 Widget Specifications

The ROLE Widget Shop should be based on common widget specifications.

For web widgets there are different specifications available. Deliverable 3.3 contains a summary of selected widget specifications and widget engines. Two of the specifications are the W3C widget specification [43] and the OpenSocial specification [27] which is based on Google Gadgets.

- OpenSocial is used in many widget containers (iGoogle, Linkedin, HI5, etc) and there exists a free implementation of a widget engine for OpenSocial [27] (Apache Shindig [3]) which is available for JAVA and PHP. An important feature of OpenSocial is the extendable specification. It was decided in deliverable 3.3 and deliverable 4.2 to focus first on OpenSocial.
- W3C Widgets are specified by the World Wide Web Consortium (W3C). Implementations of widget engines are available from the Palette Project [31] and Apache Wookie [4]. W3C Widgets are also supported by different Mobile Vendors e.g. Opera in Cooperation with Vodafone [16].

A long term goal is to support also other specifications for widgets. Candidates are desktop widget approaches like Adobe Air [1], Dashboard (Apple) [6], Google Desktop Gadgets [12], Microsoft Gadgets [47], Opera [29] and Yahoo! Widgets [49]. Netvibes [23], a starting page service, has its own Widget API (UWA [20]). These widgets can be included in different web and desktop widget containers.

5.2 Authentication

The access to the ROLE Widget Store should be easy and secure.

Besides creating a user account by username and password users should be able to use common services which provide same credentials. Services which provide such features are OpenID [26], Facebook connect [9] or Sign in with Twitter [39]. Using such services will lower the acceptance threshold and password stress for the learner. The ROLE Widget Shop will at least support OpenID.

Further there are ambitions to connect the ROLE Widget Store with existing PLE / PLMS systems from the ROLE consortium using a Single Sign-on mechanism.

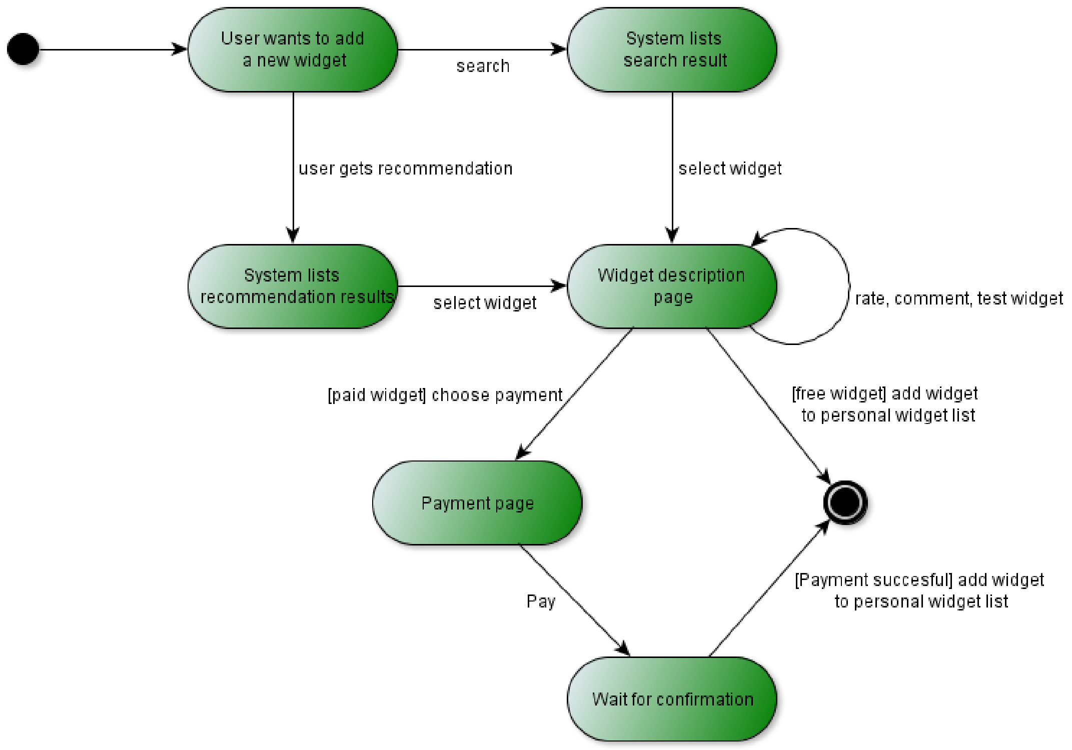
5.3 Basic Widget Store Requirements

The ROLE Widget Store should provide standard web shop features.

Most of the learners will be familiar with different web shop system in the internet. The ROLE Widget Store will provide some standard functionality of a web shop system.

5.3.1 Acquisition Process

The ROLE Widget Store should provide the learner with a simple mechanism to obtain widgets. The workflow should be similar to the workflow of common web shops which are well known by the most of the learners. The acquisition workflow which will be implemented in the ROLE Widget Store is shown in Figure 4.


User wants to add a new widget
search
System lists search result
user gets recommendation
select widget
System lists recommendation results
select widget
Widget description page
rate, comment, test widget
[paid widget] choose payment
[free widget] add widget to personal widget list
Payment page
Pay
[Payment succesful] add widget to personal widget list
Wait for confirmation

.

Figure 4: Acquisition Workflow

5.3.2 Digital Products

As widgets are digital products, the shop will support this type of product. This feature is provided by most shop systems, but the ROLE widget shop needs an additional requirement. The widget file must be accessible all the time a widget engine needs it to render the widget. Thus the URL of the widget may not change as it is implemented in different web shop systems for security reasons. Further it should not be possible to access the file if the user has not the necessary rights.

As the files of the widgets are stored (at least a part of it) in the store, the ROLE Widget Store acts as widget repository.

5.3.3 Multi Vendor Support

The ROLE widget store must provide the functionality that different vendors can provide widgets. Each vendor will be able to manage the widgets she uploaded. The widgets of the different vendors are appearing in a common store catalog.

5.4 Community Features

The ROLE Widget Store should provide social community features

The ROLE Widget Store will be enriched by a strong community. The community is needed to rate and comment the different widgets and bundles to assure a first quality assurance. Further the ROLE Widget Store is the place where user could come in contact with developer and report bugs and requests features. Also for the discussion about best practices for designing widget bundles the store is the perfect platform.

To support the community the ROLE Widget Store will provide basic social network features like profiles, buddy management, messages and a forum.

5.5 Quality Management

The ROLE Widget Store should provide a quality management to ensure the high quality of education related widgets.

One result of the research of different widgets stores is that there exists many “just-for-fun-widgets” and only a few e-learning related widgets in the stores (see Figure 5).

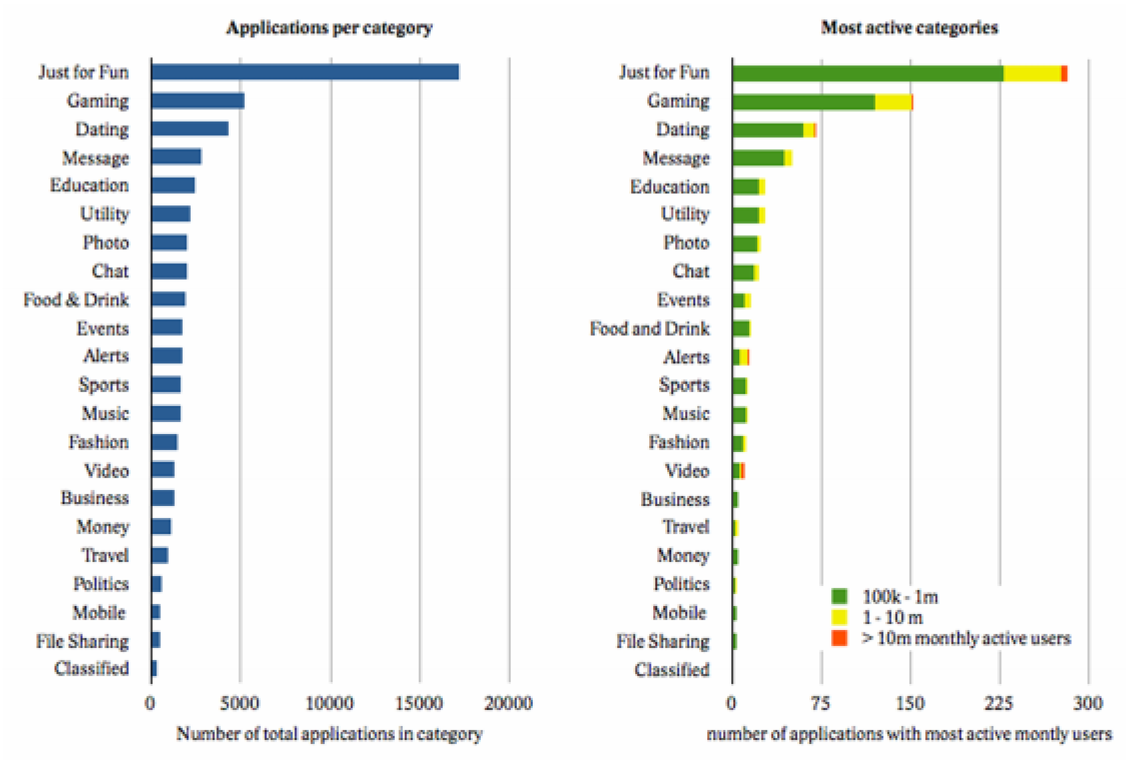


Figure 5: Facebook application categories and user activity from AppData (23.3.2009) [19]

Further there are many clones and poor quality widgets available in the widget stores. Thus to ensure the quality of the widgets a controlled publishing mechanism is needed which allows the publishing of a widget in the shop only if the quality was assessed.

5.5.1 Rating and Commenting

User should be able to rate and comment widgets. This web 2.0 crowd sourcing approach is an important feature for users to estimate the quality of a widget. It was also shown in [18] that rating (21%) is after description (27%) the second most important reputation element by choosing web widgets from a catalogue. Users will be able to rate and comment widgets in the widget store.

5.5.2 Preview and Testing of Widgets

People who are looking for widgets should be able to preview and test the widgets in the store. It will be not enough to provide a screenshot of the widget. The ROLE Widget Store will provide a preview and test mechanism directly in the store. To use this feature reasonable the vendors should provide demo accounts for widgets with personalized web services. Technically this means that the ROLE Widget Shop serves also as of Widget Container

5.6 Connection to other Widget Repositories

The ROLE Widget Store should provide the feature to catalogue external widgets

As shown in section 4.1 there exist already several widget stores. Different repositories (e.g. iGoogle and Netvibes) provide an API to access their content. Adding selected widgets from their content would be an added value as learner are able to manage their already used widgets in the ROLE infrastructure and the number of the widgets in the ROLE Widget Store could be easily increased.

There are two possibilities to provide such a feature. First the catalogue of other widget repositories could be integrated in the ROLE Widget Store. This approach would mean to access different stores over a provided API and list the content of these stores in the ROLE Widget Store. Second the ROLE Widget Store could provide import functionality as shown in Figure 6. Here only specific widgets would be imported on user demand.

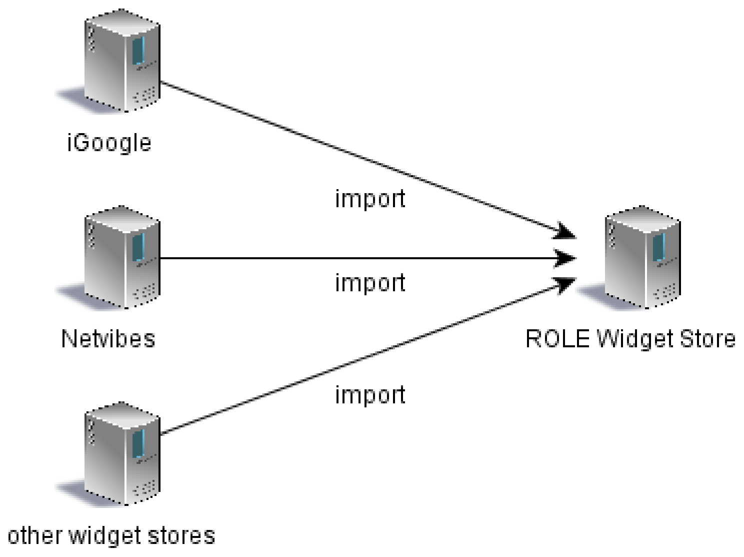


Figure 6: Importing widgets form other repositories

The advantages of the import functionality would be a better quality assurance as importing would pass by the quality management of the ROLE Widget Store. Thus the ROLE Widget Store will provide import functionalities of widgets from other widget stores.

5.7 Recommendation

The ROLE Widget Store should provide recommendations

Recommendation is a key feature in ROLE. As the ROLE Widget Store is the part of the project where people select its tools (widgets) they should get guidance by recommendation from the store. The innovative aspect of ROLE is to provide recommendations not only for single widgets but also for widget kits and widget bundles. Some recommendation features will be integrated into the core store system, other will depend on external services.

5.7.1 Who is using this Product is also using that Product

Automatic recommendation based on the view and adding history of a user is a common feature in state of the art web shop systems. These recommendations will be among others displayed directly on a widget description page.

5.7.2 Recommend Widget Kits based on Vendor Configurations

Vendors who are developing widgets which interact together should be able to recommend the using of widget kits. Thus vendors will be able to submit kits of widgets to the catalogue. These kits will be handled in the store system like single widgets.

5.7.3 Recommend Widgets Based on a User Driven Tag System

Learner and vendors should be able to tag widgets in the store system and get recommendation based on these tags. The tags will be used to recommend related widgets. These recommendations will be also displayed directly on a widget description page.

5.7.4 Filter Widgets by User Preferences

Users have different preferences in learning activities. The system should provide the learner with a possibility to set her personal learning preferences. These preferences will be used by the learner to filter search results and support the recommendation. A possible preference slider is displayed in Figure 7.

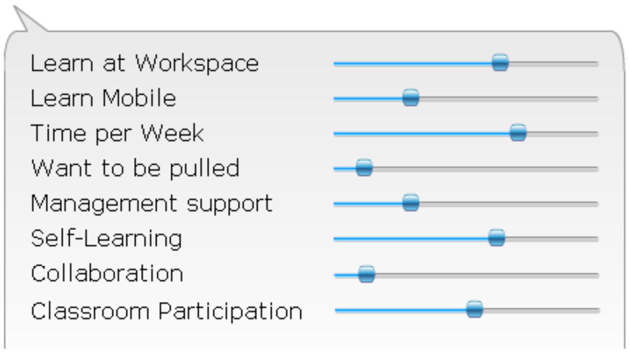


Figure 7: Learning Preferences Slider

5.7.5 Recommend Widgets by using the Theory Driven Approach from WP2 / WP6

Work packages 2 and 6 will provide a theory driven recommendation approach. The diagram (Figure 8) shows that recommendations are made based on user model information (skills, preferences and goals), domain information, and context data. In the first place learning activities are recommended. Then according to the recommended learning activities, tools/widgets are recommended. So there is a relation needed between widgets and learning activities. Also 'tool skills' should be taken into account which reveals if learners are able to perform certain learning activities with specific widgets. Consequently, there is a need that widgets are assigned with learning activities meaning that it is indicated which learning activities can be performed with which widget. This has probably rather be done in the publishing process and not by the community. There is taxonomy of learning activities defined in WP2 and WP6.

For this approach different user, domain and context data will be processed. At least the context data has to be provided by the PLE / PLMS of the user. The store will provide an API where the PLE / PLMS systems will request recommendations by providing the necessary data. The store will use an external service developed by partners to implement this kind recommendation. It could also be possible to import the necessary data from PLE / PLMS systems to use this recommendation inside the store.

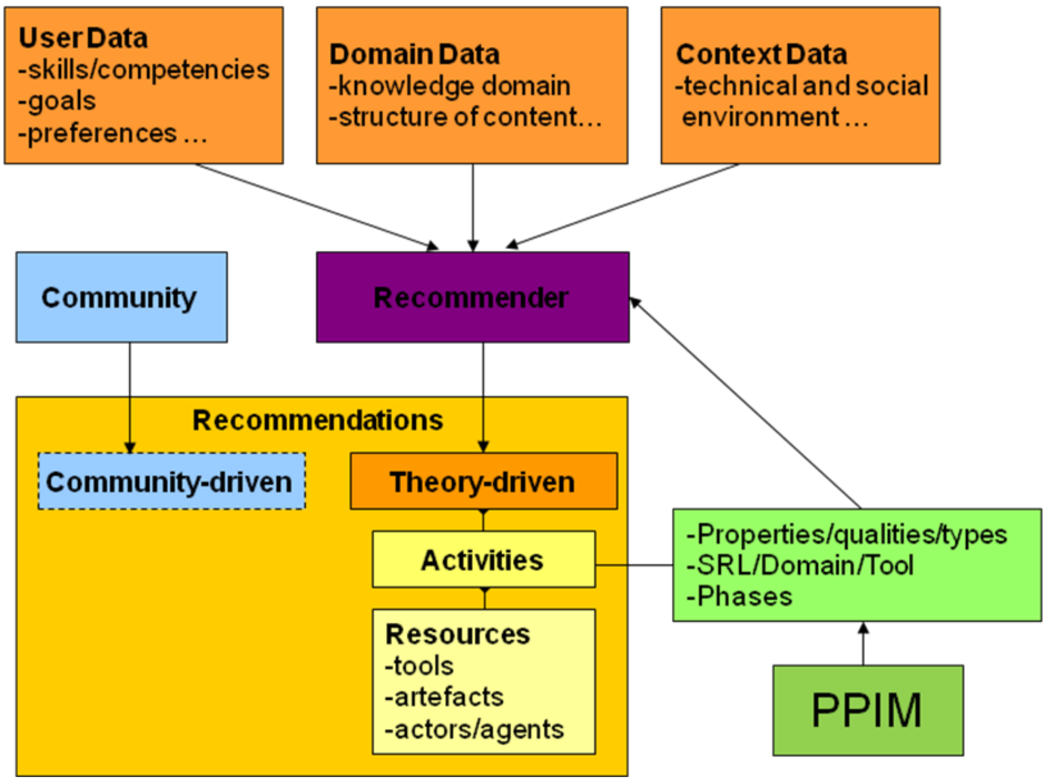


Figure 8: Recommender Model of WP2

5.7.6 Recommend Widget Bundles

It is important that the ROLE project provides also the recommendations of widget bundles. This includes the user driven recommendation as well as other outlined approaches.

Further ROLE should investigate the possibilities of auto generating bundles for learners. In order to do so the recommendation engine has to be able to assess the compatibility among widgets, cluster widgets by similarity/dissimilarity and combine them into bundles according to user learning goals.

5.8 Support of Learning Activity Page Creation for Teachers

The ROLE Widget Store should support teacher by creating learning activities pages for their students

Teacher should be able to prepare learning activity pages where students do not have to activate widgets using user credentials. The learning activity pages are special widget bundles. The ROLE Widget Store will provide the creation of special widget bundles where teacher are able to set member limits and validity period.

Further the teacher will be able to invite learners to add the learning activity page to a supported PLE of their choice. The invited learners do not need a ROLE Widget Store account and will not have to activate the provided widgets as the activation will be done by the teacher while she is creating the learning activities page.

5.9 Sharing of Widget Preferences

The ROLE framework should provide the learner with a storage functionality of widget preferences which offer independence from containers and allow collaboration.

The need of a ROLE Widget Preference server was already described in deliverable 3.3 and deliverable 4.1. At such a server two different kinds of preferences should be stored: first widget preferences which store configuration of a single widget instance and second widget space configurations which store the configuration of widget spaces. The configuration of widget spaces includes the widgets in the space, the layout of the space, users which are allowed to use the space and the widget preferences of the widgets. Possibilities to store space configurations are described in the paper Towards Collaborative Portable Web Spaces [40].

As the ROLE Widget Store is a central platform of the ROLE infrastructure it will be connected to a preference server and act as mediator between the different widget container and the preference server. Figure 9 shows the architecture.

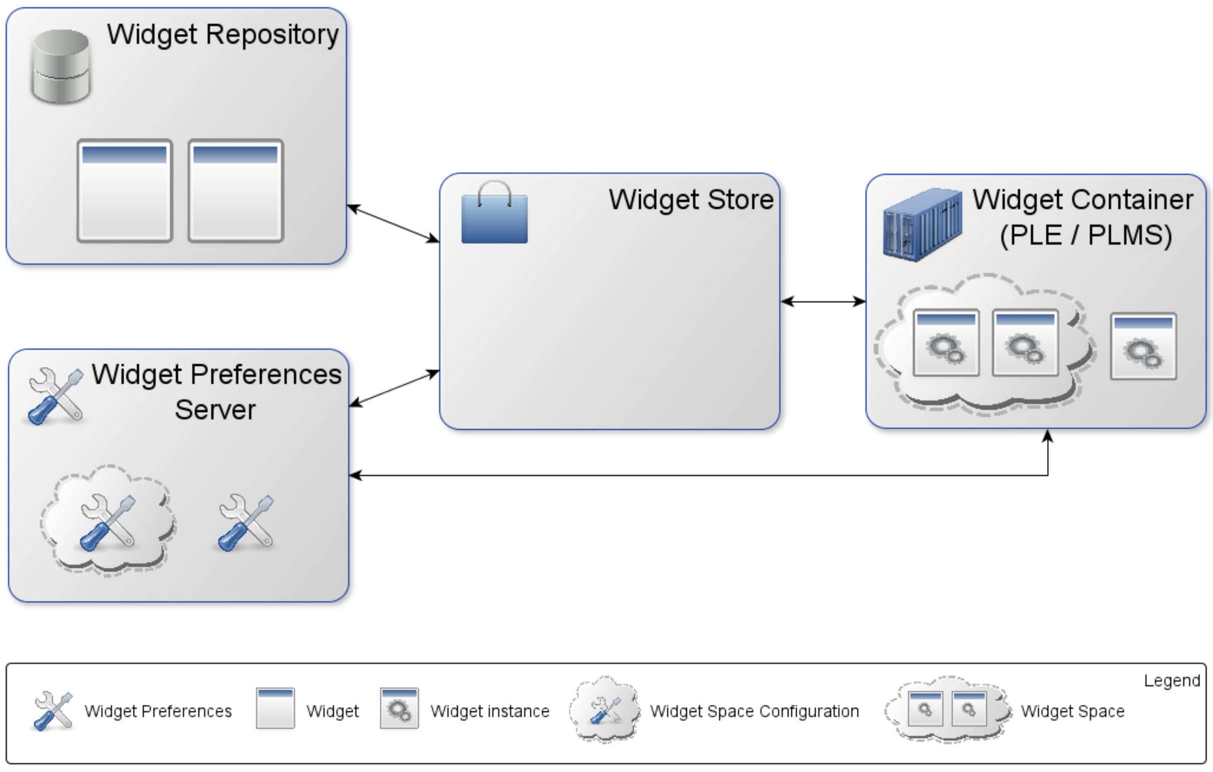


Figure 9: Widget preferences architecture

There are different purposes both for widget preferences and space preferences:

- Synchronization on different containers for one user.
- Collaborative use of widgets or spaces for different users (at different containers)
- Preparing a widget or space and publish the preferences at the ROLE Widget Store.

The first two purposes have different requirements as the third one. For synchronization widget instances need to have access to the preferences stored on the preference server. For the third one the preferences will be exported to a container and will not be modified. First focus will be on exporting space preferences and later support widget preferences and widget space configuration synchronization.

In the future the widget/space preferences server can turn into a federated combination of servers. Thus the widgets do not depend only on one widget preferences server but every widget container can use the technology and can interact with other preferences servers.

5.10 Interfaces

The ROLE Widget Store should be an open system which is part of the ROLE infrastructure and can be easily access by internal and external tools.

To fulfill this requirement the ROLE Widget Store will provide different APIs based on common technologies like REST [10]. For authorization OAuth will be used which is supported by common REST implementations and Apache Shindig [24].

The next sections describe the APIs which will be implemented.

5.10.1 API for Adding Widgets to PLE / PLMS systems

As described in the use case “Add Widgets to a PLE” a PLE should be able to add Widgets which are obtained by a learner to the learners space without visiting the ROLE Widget Store. For this reason the ROLE Widget Store will provide an API which can be used by PLE Developer to get the list of obtained widget for a learner. By using the information provided by the API the PLE can allow its learners to add widgets directly.

5.10.2 API for Requesting Recommendations

A learner should get recommendations directly in her PLE. The ROLE Widget Store will provide a API which processes the context of the PLE and returns recommendations based the submitted context. This API could be used by PLE as well as widget developers.

5.10.3 API for Storing Widget Spaces

For learners it could be more intuitive to create widget spaces in their PLE instead of creating widget bundles in the ROLE Widget Store. For that reason the store will provide an API to store created widget spaces directly from the PLE in the store as widget bundles. This will be done by providing an XML specification which describes the widget space. Sire et al. [40] describing such a specification by extending the unofficial Google GagdetTabML specification.

5.11 Single Sign On through Service Access Control Framework

The ROLE Widget Store should provide a framework for single sign on.

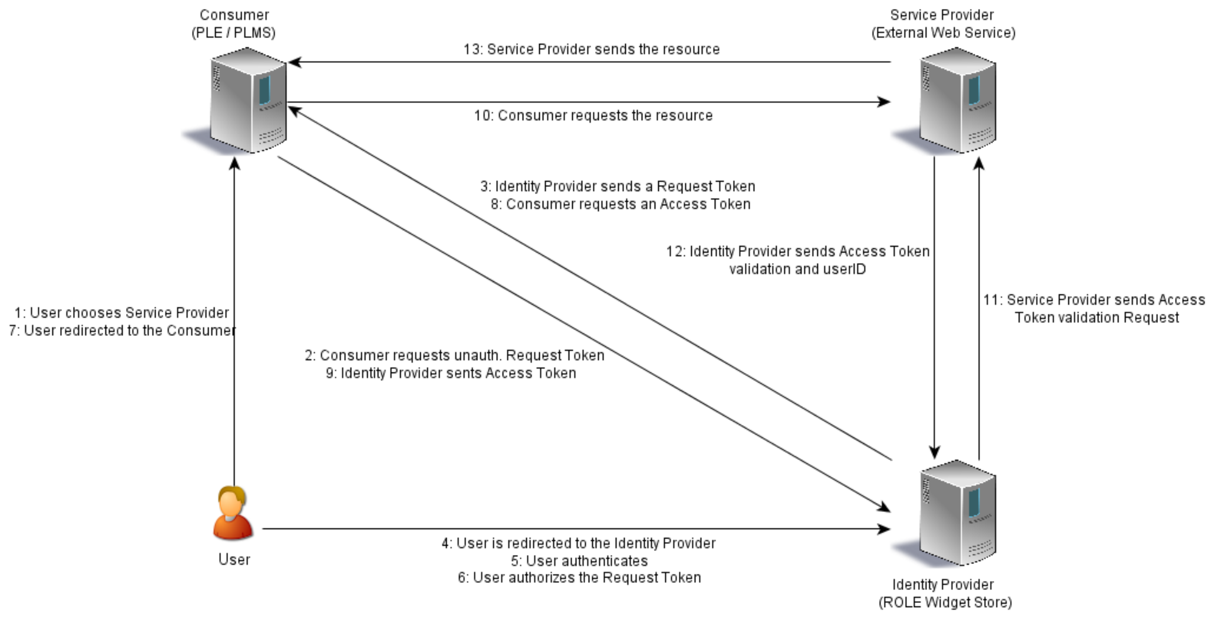
Both users and widget developers currently suffer from not having a common (single) sign on system for widgets.

Most widgets which serve personalized data are based on web services which require user credentials. The current state of the art is that widget developers create local solutions for their widgets. The learner has to create an account for each service and has to log in if she wants to use it. Identity services (e.g. OpenID) and authorization services (e.g. OAuth) are helping to create a user friendly environment.

An added value of the Widget Store will be creating an easy and accepted authentication system which can be used to replace the current local solutions of the different existing widget vendors.

Gonález et al. describe in [11] different initiatives to achieve delegated authorizations in e-learning systems. They mention OAuth [35], Delegation Permits [13], Shibboleth [38] and OpenSSO [28]. None of these approaches satisfy their requirements. Thus they created Reversed OAuth.

Also for service access control in the ROLE infrastructure none of these approaches is directly applicable. A first suggestion based on ScalableOAuth [35] which is an extension of OAuth is outlined in Figure 10. The ROLE Widget Store will serve as central identity provider for all web services which are using the framework.



Consumer (PLE / PLMS)
Service Provider (External Web Service)
13: Service Provider sends the resource
10: Consumer requests the resource
3: Identity Provider sends a Request Token
8: Consumer requests an Access Token
12: Identity Provider sends Access Token validation and userID
11: Service Provider sends Access Token validation Request
1: User chooses Service Provider
7: User redirected to the Consumer
2: Consumer requests unauth. Request Token
9: Identity Provider sents Access Token
4: User is redirected to the Identity Provider
5: User authenticates
6: User authorizes the Request Token
User
Identity Provider (ROLE Widget Store)

Figure 10: Authentication via a central widget store

This approach has the advantage that users only have to log in to the widget shop and vendors do not have to care about user management. It will be possible to manage the access of widget services using the ROLE Widget Store.

The ROLE Widget Store will host three different kinds of widgets:

- Widgets which do not use web services
- Free Widgets which uses web services
- Paid Widget which uses web services

For the first kind of widgets the Service Access Control Framework is neither needed nor possible to use. This includes, that it is not possible to have a secure copy protection for widgets without web services. For the second subset of widgets there is no need to use the service but developer can get much benefit by using it. The third kind of widget requires the framework.

Based on this approach the supporting of creating learning activity pages described in section 5.8 will be realized.

To attract this approach for developers JavaScript libraries for the widget development will be created as well as libraries on the server side (e.g. JAVA and PHP).

5.12 User Interface Requirements

The ROLE Widget Shop should provide an easy to use interface

In this section first mock-ups of the user interface will be presented. The mock-ups are based on best practices of common web shops and existing widget stores.

Figure 11 shows the starting page of the ROLE Widget Store. Users which are not logged in can browse the widget catalogue, search for widgets and get information and a tour about the widget store. Also information about widget development and the ROLE infrastructure are provided.

By browsing the catalogue they are able to see basic information of the widgets. Rating, used by, number of comments, a short description, the developer name and a greater thumbnail of the widget are shown. As noted before description and rating are important reputation elements [18]. A details button allows user to get more information about the widget as described below.

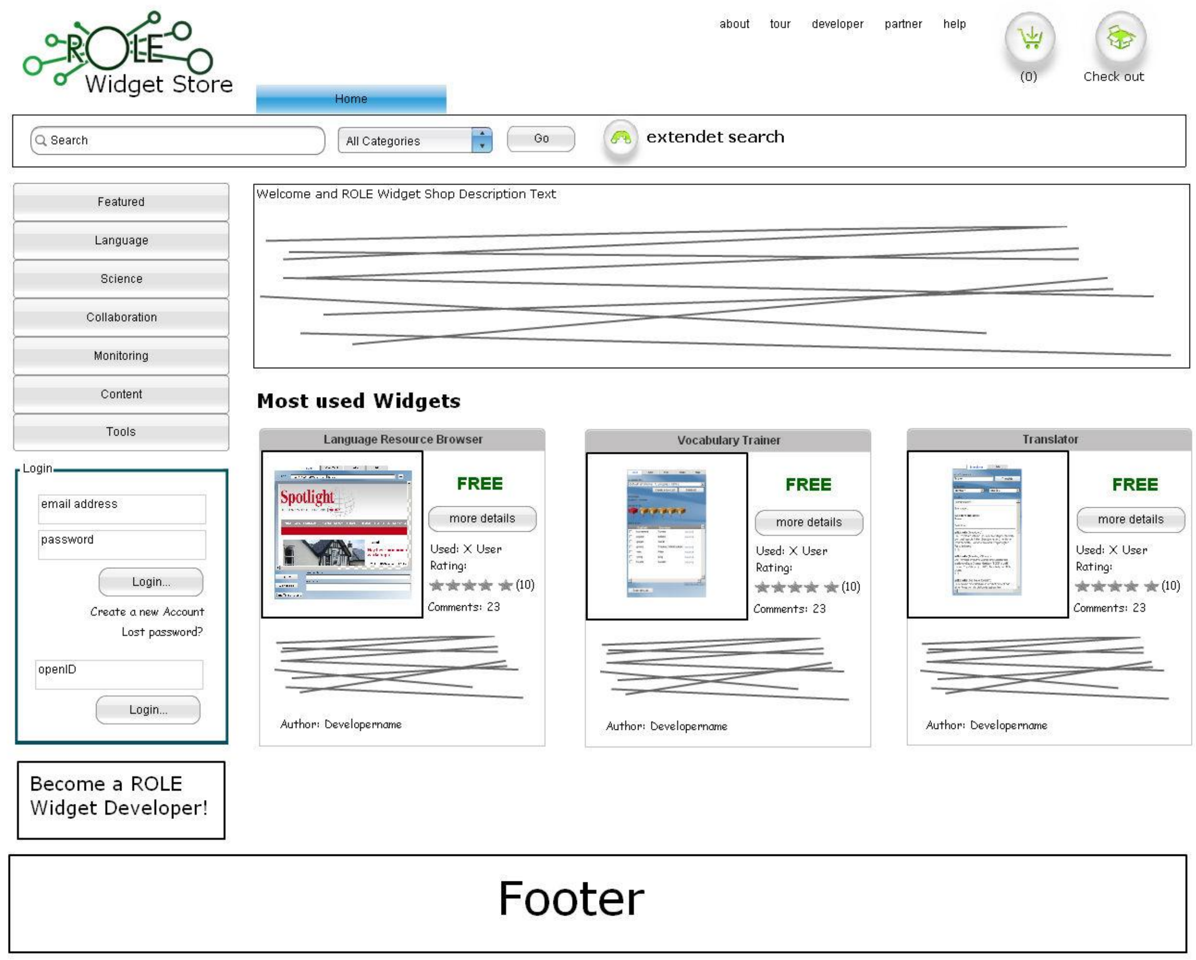


Figure 11: Home

Users which are logged in get recommendations on the home page (Figure 12). They are also able to navigate to their already obtained widgets, their widget bundles and their own account. As their user information is now available they can also get personalized recommendations.

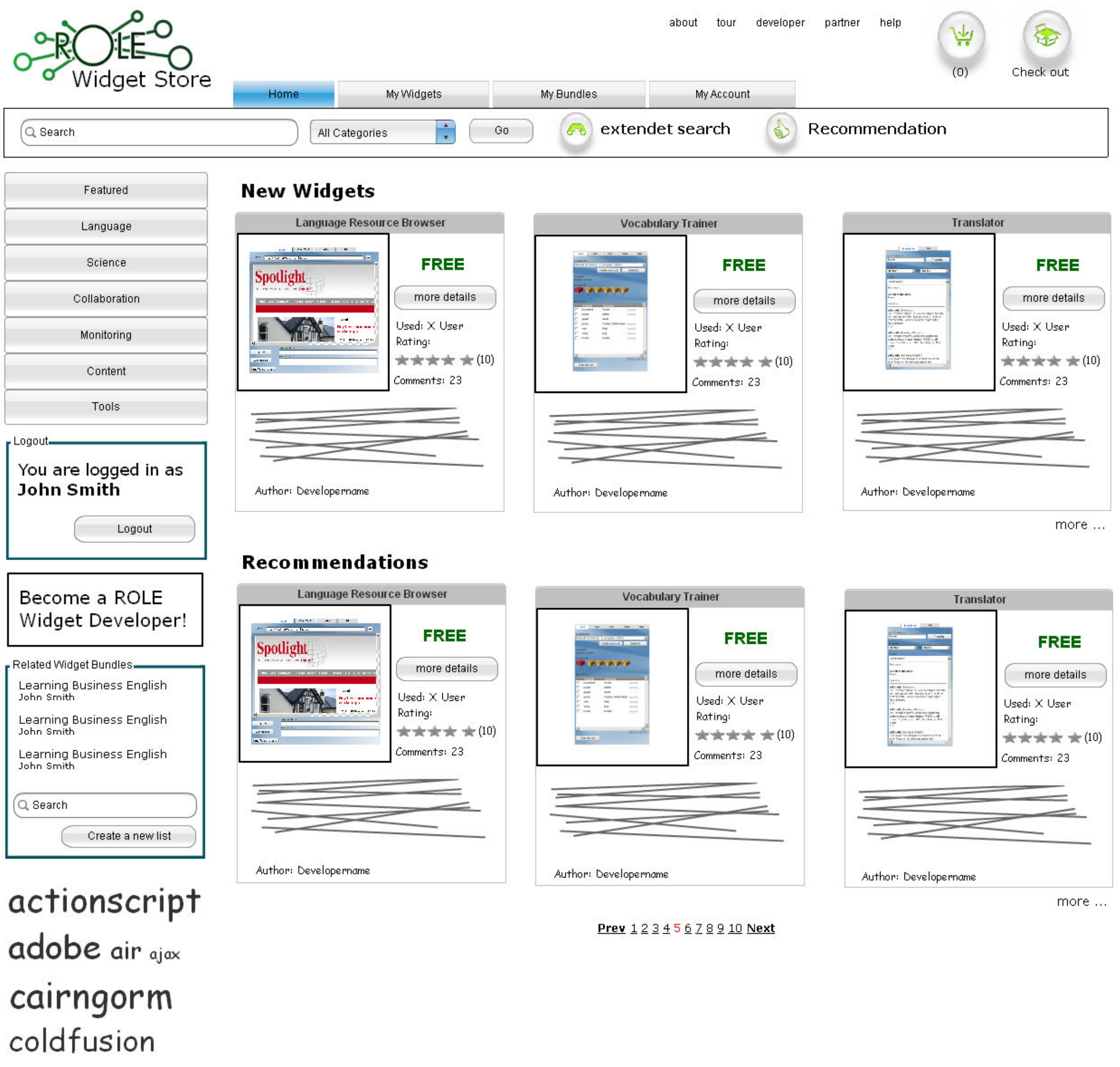
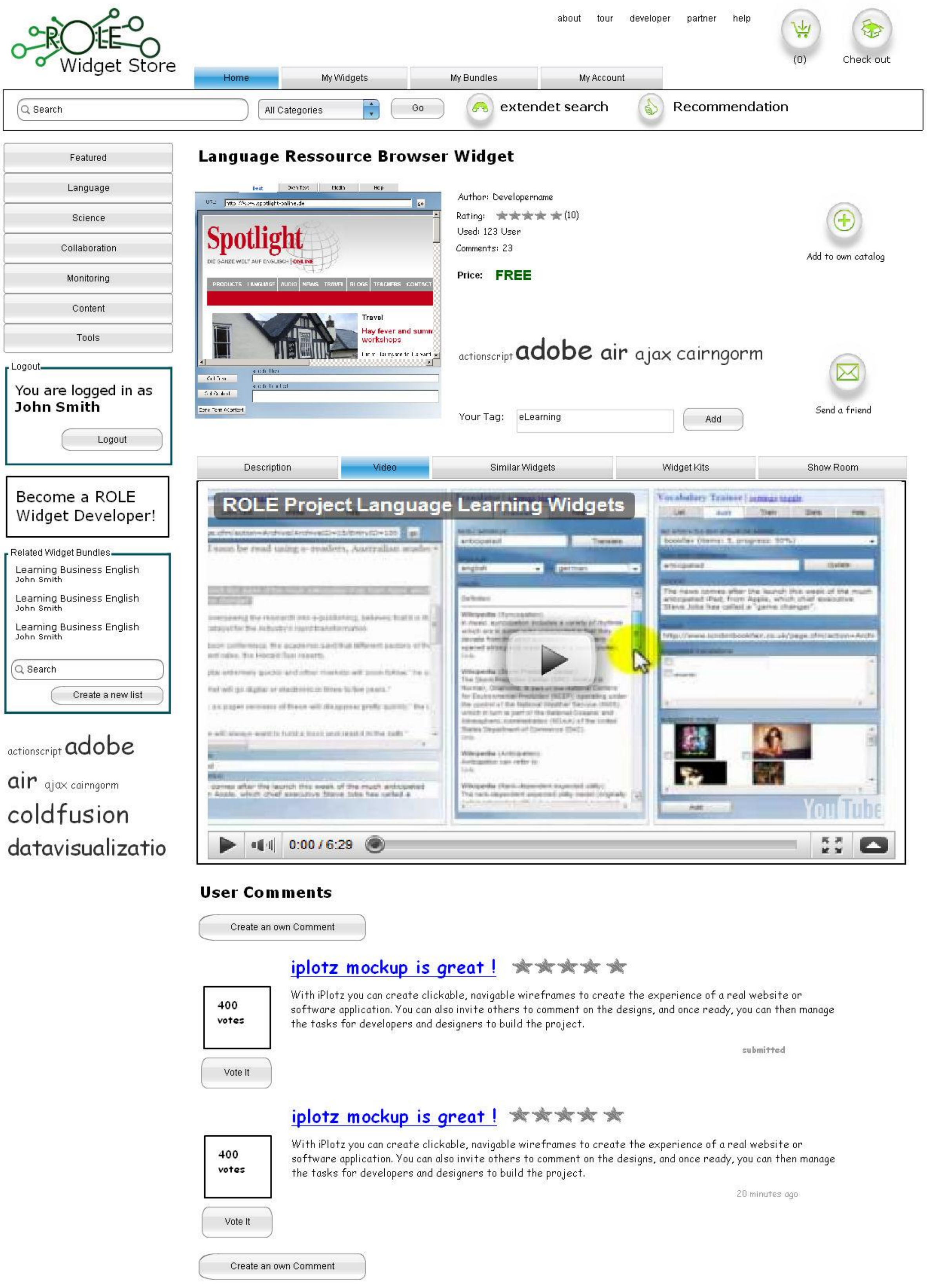


Figure 12: Home – logged in

Figure 13 shows the detailed widget description. In this view users are able to tag widgets, leave comments and rate the widgets. Here they start the obtaining process outlined in section 5.3.1. In the centre they can choose to read the full description, watch videos (if available), look for similar widgets or widget bundles which include the current widget. Further they have the possibility to use the showroom which provides a live demo of the widget.

By using the “My Widgets” Tab (see Figure 14) the user can manage their obtained widgets. Here the ROLE Widget Store provides the feature to insert widgets directly into supported PLEs.



ROLE Widget Store

about tour developer partner help

(0) Check out

Home | My Widgets | My Bundles | My Account

Search | All Categories | Go | extendet search | Recommendation

Featured
Language
Science
Collaboration
Monitoring
Content
Tools

Logout
You are logged in as **John Smith**
Logout

Become a ROLE Widget Developer!

Related Widget Bundles
Learning Business English
John Smith
Learning Business English
John Smith
Learning Business English
John Smith
Search
Create a new list

actionscript adobe air ajax cairngorm coldfusion datavisualizatio

Language Ressource Browser Widget

Author: Developername
Rating: ★★★★★ (10)
Used: 123 User
Comments: 23
Price: **FREE**

Add to own catalog

actionscript adobe air ajax cairngorm

Send a friend

Your Tag: eLearning | Add

Description | Video | Similar Widgets | Widget Kits | Show Room

ROLE Project Language Learning Widgets

0:00 / 6:29

User Comments

Create an own Comment

iplotz mockup is great ! ★★★★★

400 votes

Vote It

With iPlotz you can create clickable, navigable wireframes to create the experience of a real website or software application. You can also invite others to comment on the designs, and once ready, you can then manage the tasks for developers and designers to build the project.

submitted

iplotz mockup is great ! ★★★★★

400 votes

Vote It

With iPlotz you can create clickable, navigable wireframes to create the experience of a real website or software application. You can also invite others to comment on the designs, and once ready, you can then manage the tasks for developers and designers to build the project.

20 minutes ago

Create an own Comment

Figure 13: Widget descrption

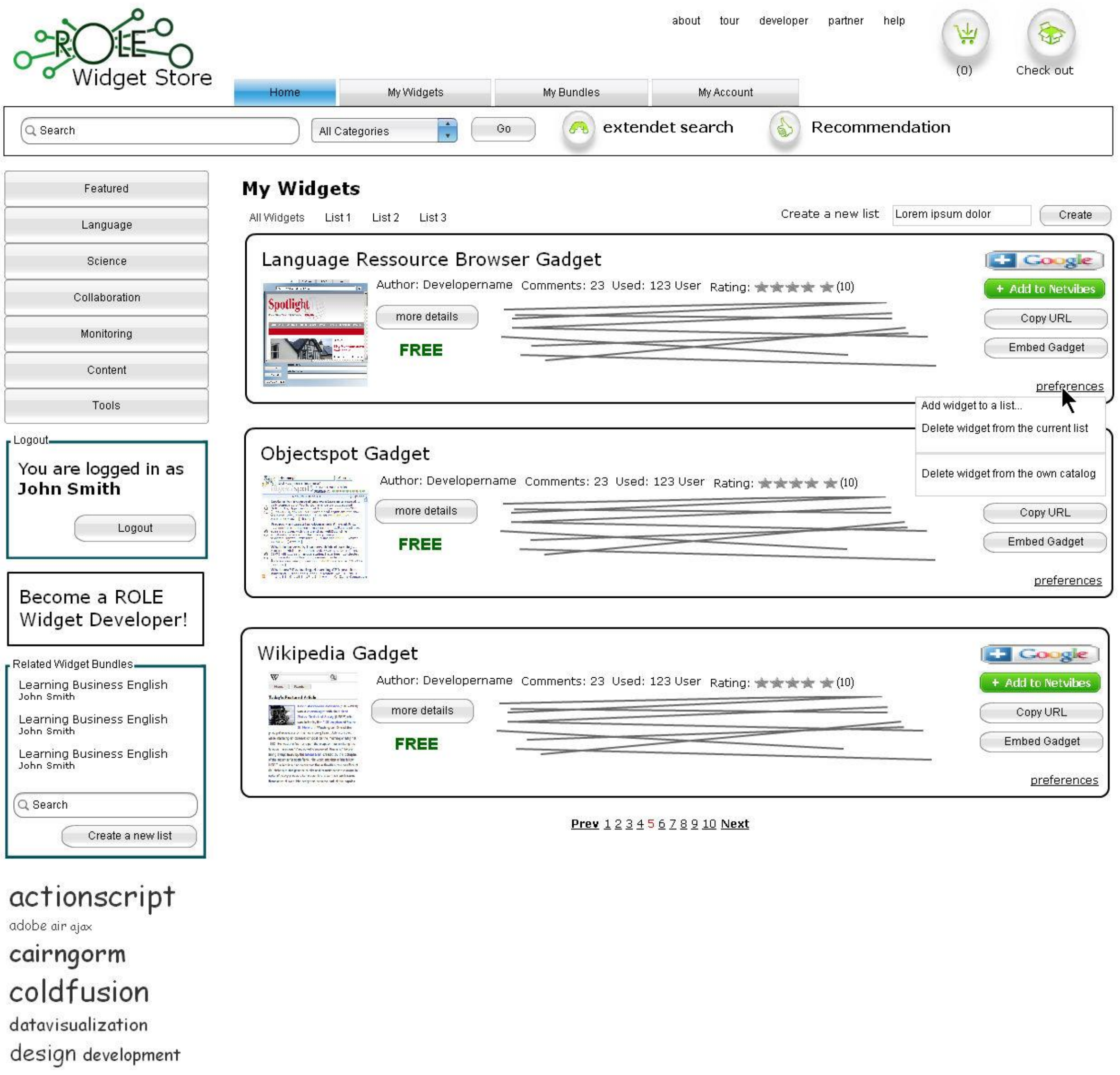


Figure 14: My widgets

5.13 Requirements which are under Discussion

5.13.1 Recommendation Related Requirements

5.13.1.1 Recommend taking into account targeted containers

In the current version of the deliverable the focus is on the OpenSocial specification (See section 5.1). In later versions it will be also possible to handle widgets for other specifications. Thus The ROLE Widgets Store will take the targeted containers into account for the recommendations of widgets.

5.13.2 Quality Assurance Related Requirements

5.13.2.1 Reputation of people should be taken into account

Reputation and privacy will be taken into account for rating and commenting. A relevant mechanism has still to be defined. It will be provided in the next versions of this deliverable.

5.13.2.2 Controlled Publishing

Only rating and commenting is not enough to ensure the quality of widgets. For that reason a controlled Quality Assurance will be developed and included in the next version of the deliverable.

A possible solution is the introduction of a Quality Manager (QM) role for the ROLE Widget Store. Widgets have to pass an editorial workflow process (Figure 15). A vendor can submit a widget and then apply for publishing. This widget will not be available to the public before it is approved by a quality manager. If the quality is sufficient, the QM will publish it, which makes it available for normal users. If the quality is not sufficient, the QM will reject the widget with comments on how to improve it. In this case the widget developer can update the widget according to the comments and submit it again. Quality managers will be trusted users of the ROLE community.

A further possibility would be creating an automatic mechanism.

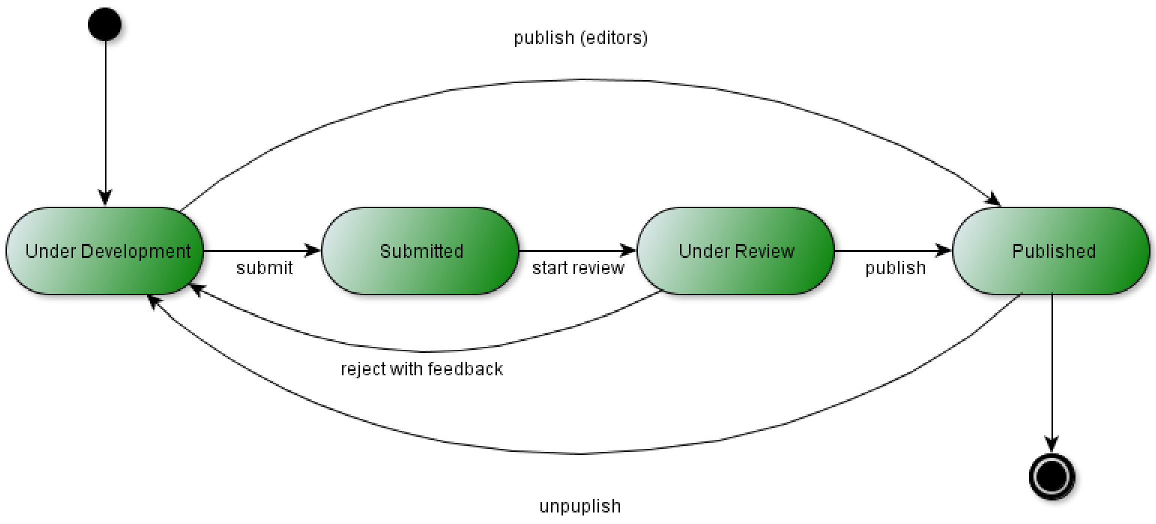


Figure 15: Quality Management Workflow

6 Implementation

As shown in chapter 3, existing systems do not fulfill the requirements of ROLE (see also chapter 5). In addition, the syndication platform with the ROLE Widget Store is one of the most important components in order to enable a personal, responsive and open learning environment approach. Therefore, there is a strong need within ROLE to develop the specific components.

In the following, the implementation and development plan is being described. The development follows an agile approach, where small sets of features are designed, implemented, tested and evaluated iteratively. To minimize the development effort, the new system will be based on an existing open source software platform (i.e. a web shop system).

6.1 Implementation Plan

For implementation an agile approach is chosen. Like in the agile software development method SCRUM [37] development iterations are defined. These iterations are based on the requirements in this document. After each development iteration, a new prototype will be available, ready to be demonstrated. Further the next iteration is planned when the last is finished. Using such an approach the ROLE project will be able to react if the requirements will be changed.

6.1.1 Efforts of the implementation partners

Task 4.2 is basically responsible for this deliverable. As mentioned the ROLE Space will not be part of this deliverable.

The core implementation will be done by IMC. For different key features there will be a strong cooperation with EPFL, TUG, UU and WW.

6.1.2 Features that will be implemented until M48

As this deliverable is a living document the list of features could change as features will be removed or added. Figure 16 gives an overview over the different functionality packages and priorities (”0” is the highest priority and “9” the lowest)). The detailed functionalities are listed by iterations in section 6.1.3.

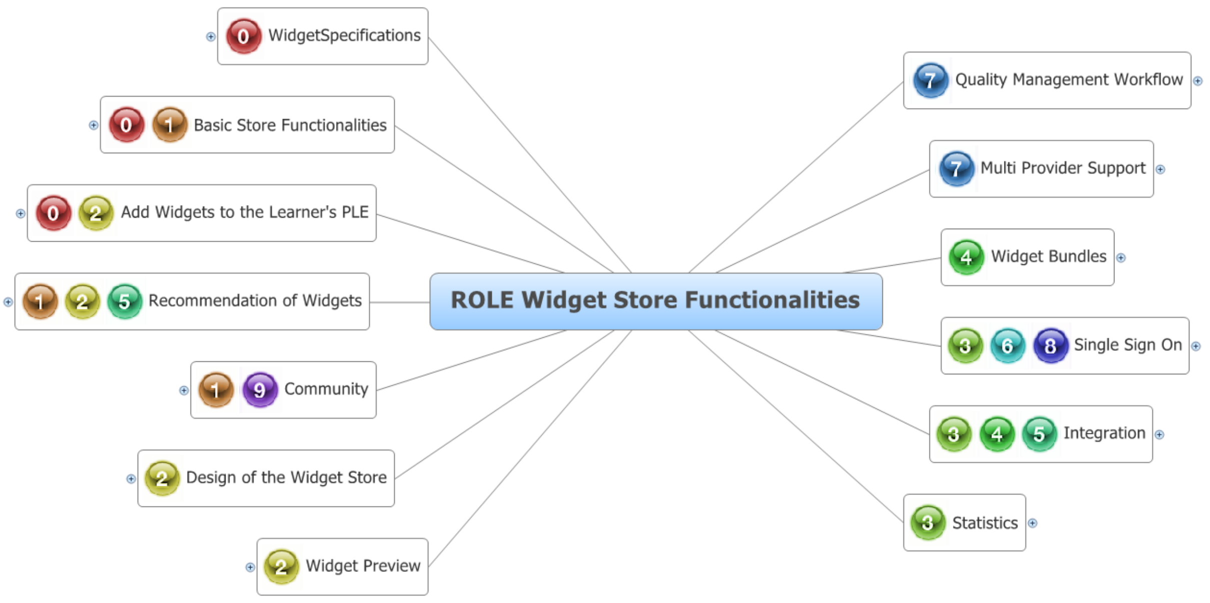

ROLE Widget Store Functionalities
0 WidgetSpecifications
0 1 Basic Store Functionalities
0 2 Add Widgets to the Learner's PLE
1 2 5 Recommendation of Widgets
1 9 Community
2 Design of the Widget Store
2 Widget Preview
7 Quality Management Workflow
7 Multi Provider Support
4 Widget Bundles
3 6 8 Single Sign On
3 4 5 Integration
3 Statistics

Figure 16: Functionality Packages and Priorities

6.1.3 Iterations

6.1.3.1 Baseline Iteration "Standard Shop" (Nr 0, till end of M18 [July 2010])

- *Store administrator is able to add widgets:*
  User belonging to the store administration group will be able to add free widgets to the ROLE Widget Store general catalogue.
- *OpenSocial widgets are supported:*
  The OpenSocial specification for Widgets will be supported. The widget XML file will be hosted at the widget store.
- *Widgets are listed in categories:*
  Different categories will be created and widget will be categorized. Users can browse such a catalogue.
- *User is able to create an account:*
  New Users are able to create a user account without interaction of the system administrator.
- *User is able to search for widgets via search terms:*
  The user can use the search function to search for widget via search terms.
- *User is able to add widgets to their own widget list:*
  Each user will have an own list of obtained widgets. Using the search or the catalogue they will be able to add new widgets to this list.
- *User is able to copy the XML URL from widget store to PLEs:*
  PLEs which supports OpenSocial mostly supporting the adding of widgets by inserting the URL of the widget XML. The ROLE Widget Store will provide this URL for obtained widgets.

6.1.3.2 Feedback Possibilities (Nr 1, till end of M19 [August 2010])

- *Store provides basic payment functionalities:*
  Store administrators will be able to add paid widgets to the store. For users a simple payment system will be provided.
- *User is able to tag widgets:*
  User will be able tag widgets on the description page. Further user will able to search for widgets via tags.
- *User is able to comment widgets:*
  User will be able to comment widgets on the description page.
- *User is able to rate widgets:*
  Users will be able to rate widgets on the description page. The five star rating will be used for the rating feature.
- *Vendors are able to create widget bundles:*
  Vendors will be able to create widget bundles by combining several of their widgets together in a bundle. These bundles will be handled as a single widget.

6.1.3.3 Simple Recommendation of Widgets and Design (Nr 2)

- *User is able to get recommendation based on community tags:*
  A user will get recommendation based in the user tags on the home page, in the recommendation feature and on the widget description pages.
- *User is able to get recommendation based on other user behaviour:*
  A user will get recommendations based on the tracked user behaviour on the home

page, in the recommendation feature and on the widget description pages. This feature will be an equivalent to the “who this bought, bought also that” feature of common web shops.

- *The store has an own design:*
  An own design will be created for the ROLE Widget Store and the required templates will be developed.
- *User initiates via the store the installation of the widget into the PLE:*
  A user will be able to insert widgets via a simple button into her PLE if her is supporting this feature.
- *User is able to preview widgets:*
  User will be able to test widgets on the widgets description page. For this an OpenSocial widget engine will be embedded in the store.

6.1.3.4 Integration with PLEs and Repositories (Nr 3)

- *User gets statistics about widgets:*
  Users will be able to get information about widgets statistics. Statistics include number of user which use a widget, number of downloads and the development over time.
- *PLE is able to get a List of widgets for a user over an API:*
  The ROLE Widget Store will provide an API for PLE developers which can be used to get the list of obtained widgets for a user of the PLE.
- *Vendors can import and describe widgets from external stores:*
  Vendors (or store administrators until vendors are introduced) will be able to import widgets from external widget stores.
- *User is able to log into the store using OpenID:*
  The ROLE Widget Store will support login with OpenID.

6.1.3.5 Widget Bundles (Nr 4)

- *User is able to create own widget bundles:*
  The ROLE Widget Store will support the creation and management process of widget bundles
- *User is able to share her widget bundles:*
  A User will be able to share her created widget bundles with the community
- *User is able to comment and rate widget spaces:*
  A User will be able to comment and rate widget bundles like widgets.
- *User is able to add a whole widget space to her PLE:*
  The ROLE Widget Store will provide the functionality to create widget spaces from widget bundles in PLEs which support this process.

6.1.3.6 Advanced Recommendation of Widgets (Nr 5)

- *User can set her user preferences:*
  A User can set user preferences as described in section 5.7.4.
- *User can filter result using user preferences:*
  User can filter the result of searches and recommendations by user preferences.
- *User is able to get recommendation based on the theory driven approach from WP2 / WP6:*
  The ROLE Widget Store will support recommendation features based on the work of WP2 / WP6.

- *PLE is able to request recommendation over an API:*
  The ROLE Widget Store will provide an API for PLE developer to access recommendations.

6.1.3.7 Single Sign-on (Nr 6)

- *Service Access Control:*
  The Service Access Control framework described in section 5.11 will be implemented.

6.1.3.8 Multi Vendor Support (Nr 7)

- *User is able to apply for a quality manager account:*
  A user can apply for a quality manager account. Quality managers have to be verified.
- *Vendor is able to submit widget for publishing:*
  Vendors are able to submit their widgets into the quality assurance process.
- *Quality manager is able to accept or decline widget for publishing:*
  Quality managers are able to accept qualitative widgets which will be added to the store and also decline low quality widgets which will return to the submitter.
- *User is able to register for a vendor account:*
  A User is able to register for a vendor which allows her to submit widgets to the ROLE Widget Store.
- *Vendor is able to publish widgets (for free):*
  A vendor is able to publish free widgets which can be simply added to a user PLE.
- *Vendor is able to publish widgets (for a fee):*
  A vendor is able to publish paid widgets which can be added to a user PLE after the user paid the onetime fee.
- *User is able to comment and rate providers:*
  A user will not only be able to rate widgets but also providers / developers of widgets.

6.1.3.9 Subscription (Nr 8)

- *Payment for subscriptions:*
  A users can subscribe to a widget for a period of time.

6.1.3.10 Community Support (Nr 9)

- *The ROLE Widget Store provides social network features:*
  The ROLE Widget Store will provide credential social network feature like a profile and messaging.
- *User is able to contact providers / developer:*
  The ROLE Widget Store will provide a contact mechanism for users to contact developers.
- *User is able to report bugs:*
  The ROLE Widget Store provides the functionality for users to report bugs in widgets.
- *User is able to create feature requests:*
  The ROLE Widget Store provides the functionality for users to request new features in widgets.
- *The store provides a forum:*
  The ROLE Widget Store will include a forum where users will be able to discuss Widget and PLE related topics.

6.2 Implementation Technologies

6.2.1 Platform for the Widget Store

ROLE is an open project and aim to attract external developer and open source communities. This underlines the decisions to choose an open source platform for the development of the ROLE Widget Store.

The "Open-Source E-Commerce Guide" rated in December 2009 the ten most famous open source web shop solutions by performing a Boolean Web search [25]. The result is show in Table 1.

Table 1: Ten most famous open source web shop solutions (Dec 2009)

Rank	December 2009
1	OsCommerce
2	Zen Cart
3	Magento
4	XT:Commerce
5	Cube Cart
6	X-Cart
7	PrestaShop
8	Quick Cart
9	Ubercart
10	CRE Loaded

One of the requirements derived above is the multi vendor support (section 5.3.3). This feature was found for the following two platforms as an extension or plugin (no platform supports it in the base version):

- For OsCommerce there exists a multivendor project from osCDevShed [30].
- For Ubercart [32] there exists an additional module which provides multi vendor support.

Beside these two solutions there are different commercial shop systems which fulfil this requirement and also proprietary developed extensions for different platforms listed on table 1.

The shop system Ubercart was chosen which is based on Drupal as platform for the ROLE Widget Store for several reasons:

- The last version osCDevShed was published in June 2008 and OsCommerce is loosing popularity.
- Drupal has an active developer community
- It is based on an extendable modular system
- It is mostly well documented
- It is Opensource
- Ubercart is a well-engineered web shop module which provides

    - Selling items in a virtual economy
    - Automatic account generation
    - Integrated payment system (Cyber Source, Authorize.net, PayPal, etc)
    - Simple order processing
    - Configurable product catalogue
    - Extendable with modules for
        - Multi vendor support
        - Rating
        - Recommendation

These advantages will compensate the disadvantage of more configuration work compared with an out-of-the-box system.

6.3 Status of the ROLE Widget Store Prototype Development

This section describes the ongoing development of the ROLE Widget Store Prototype. The actual prototype is hosted by IMC AG under the following URL:

http://widgetstore.role-project.eu/

The version of the ROLE Widget Store which will be hosted under this URL will be current stable version of the development. After each finished iteration the demo server will be updated.

6.3.1 2010-06-28 Start of the Development

- The sub domain was configured and prepared for the development.
- A SVN server was set up.
- Based on selection of the platform for the ROLE Widget Store the baseline iteration (section 6.1.3.1) is currently under development and will be deployed on the ROLE demo server.

6.3.2 2010-07-22

- The first Iteration is finished and the stable version was uploaded to the demo server.

7 Concluding Remarks

The document will be continuously updated with the current state of development as already stated above. Also changes which will be made at the requirements will be inserted in the deliverable. After each successful iteration the new prototype will be installed under the URL described under section 6.3.

8 References

[1] Adobe Air http://www.adobe.com/products/air/ [19.05.2010]

[2] Analyst: There's a great future in iPhone apps. http://digital.venturebeat.com/2008/06/11/analyst-theres-a-great-future-in-iphone-apps/ [19.05.2010]

[3] Apache Shindig. Online available: URL: http://shindig.apache.org/ [19.05.2010]

[4] Apache Wookie. Online available: URL: http://incubator.apache.org/wookie/ [23.06.2010]

[5] Apple appstore Online available: URL: http://www.apple.com/ipad/app-store/ [19.05.2010]

[6] Apple Dashboard Widgets. Online available: URL: http://www.apple.com/downloads/dashboard/ [19.05.2010]

[7] Apple Developer Guidelines. Online available: URL: http://developer.apple.com/documentation/AppleApplications/Conceptual/Dashboard_ProgTopics/index.html [19.05.2010]

[8] Apple iPhone Apps. Online available: URL: http://www.apple.com/iphone/apps-for-iphone/ [19.05.2010]

[9] Facebook-Connect Online available: URL: http://www.facebook.com/advertising/?connect [19.05.2010]

[10] Fielding, R. T.(2000) Architectural Styles and the Design of Network-based Software Architectures Online available: URL: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm [19.05.2010]

[11] Gonzalez, J., F., Rodriguez, M.,C., Nistal, M.,L., Rifon, L., A. (2008), Reverse OAuth: A solution to achieve delegated authorizations in single sign-on e-learning systems, Computers & Security, Volume 28, Issue 8, Pages 843-856, ISSN 0167-4048, DOI: 10.1016/j.cose.2009.06.002.

[12] Google Desktop. Online available: URL: http://desktop.google.com/ [19.05.2010]

[13] Hasan, R., Winslett, M., Conlan, R., Slesinsky, B., and Ramani, N. (2008). Please Permit Me: Stateless Delegated Authorization in Mashups. In Proceedings of the 2008 Annual Computer Security Applications Conference (December 08 - 12, 2008). ACSAC. IEEE Computer Society, Washington, DC, 173-182. DOI= http://dx.doi.org/10.1109/ACSAC.2008.24

[14] HTTP Authentication Online available: URL: http://tools.ietf.org/html/rfc2617 [19.05.2010]

[15] iGoogle. Online available: URL http://www.google.com/ig/ [19.05.2010]

[16] Introduction to W3C Widgets. Online available: URL: http://www.quirksmode.org/blog/archives/2009/04/introduction_to.html [23.06.2010]

[17] ITWissen. Online available: URL: http://www.itwissen.info/definition/lexikon/Widget-widget.html [23.07.2010]

[18] Karvonen, K., Kilinkaridis, T., and Immonen, O. 2009. WidSets: A Usability Study of Widget Sharing. In Proceedings of the 12th IFIP TC 13 international Conference on Human-Computer interaction: Part II (Uppsala, Sweden, August 24 - 28, 2009). T. Gross, J. Gulliksen, P. Kotzé, L. Oestreicher, P. Palanque, R. O. Prates, and M. Winckler, Eds. Lecture Notes In Computer Science, vol. 5727. Springer-Verlag, Berlin, Heidelberg, 461-464. DOI= http://dx.doi.org/10.1007/978-3-642-03658-3_50

[19] Mathis, K., (2009), E-Learning in European Web-based Social Networks Using Widgets, Bachelor Thesis, FH Furtwangen

[20] Netvibes Developers. Online available: URL: http://dev.netvibes.com/doc/ [19.05.2010]

[21] Netvibes eco system Online available: URL: http://eco.netvibes.com [19.05.2010]

[22] Netvibes UWA API Online available: URL: http://dev.netvibes.com/doc/uwa [19.05.2010]

[23] Netvibes. Dashboard Everything. Online available: URL: http://www.netvibes.com/ [19.05.2010]

[24] OAuth. Online available: URL: http://oauth.net/ [19.05.2010]

[25] Open Source E-Commerce Guide Online available: URL: http://www.ecommerce-guide.com/essentials/shopping_carts/article.php/3855641 [19.05.2010]

[26] Openid Online available: URL: http://openid.net/ [19.05.2010]

[27] OpenSocial. Online available: URL: http://code.google.com/intl/de/apis/opensocial/ [19.05.2010]

[28] OpenSSO. Online available: URL: https://opensso.dev.java.net/ [25.06.2010]

[29] Opera Widgets. Online available: URL: http://widgets.opera.com/ [19.05.2010]

[30] oscdevshed Online available: URL: http://www.oscdevshed.com [24.05.2010]

[31] Palette Project: Online available: URL: http://palette.ercim.org/ [23.06.2010]

[32] Papadongonas, G., and Doxaras, Y., (2010), Drupal e-commerce with Ubercart 2.x, Birmingham: PACKT Publishing

[33] Renzel, D., Höbelt, C., Dahrendorf, D., Friedrich, M., Mödritscher, F., Verbert, K., Govaerts, S., Palmér, M. and Bogdanov, E. (2010). Collaborative Development of a PLE for Language Learning. International Journal of Emerging Technologies in Learning (iJET), Vol 5 (2010). Online available: URL: http://online-journals.org/i-jet/article/view/1196

[34] ROLE Glossary, Online in the Internet: URL: https://ilias.uni-koblenz.de/repository.php?cmd=frameset&ref_id=947 [20.07.2010]

[35] ScaleableOAuth. Online available: URL: http://wiki.oauth.net/ScalableOAuth [24.06.2010]

[36] Schmidt, M., (2009), Personal Learning Environments in Human Resource Development, 4th European Conference on Technology Enhanced Learning (EC-TEL), Nice, France, September 29 - October 2, 2009. Online available: URL: http://www.role-project.eu/wp-content/uploads-role/2009/10/personal-learning-environments-in-human-resource-development2.pdf [16.06.2010]

[37] scrumalliance Online available: URL: http://www.scrumalliance.org/ [24.05.2010]

[38] Shibboleth. Online available: URL: http://shibboleth.internet2.edu/ [25.06.2010]

[39] Sign in with Twitter Online available: URL: http://apiwiki.twitter.com/Sign-in-with-Twitter [24.05.2010]

[40] Sire, S, Bogdanov, E., Palmér, M., and Gillet, D. (2009), Towards Collaborative Portable Web Spaces. 4th European Conference on Technology Enhanced Learning (EC-TEL) - Workshop on Mash-Up Personal Learning Environments (MUPPLE'09), Nice, France, September 29 - October 2, 2009. Online available: URL: http://infoscience.epfl.ch/record/140943/files/mupple-portable-ple.pdf

[41] SpringWidgets. Online available: URL: http://www.springwidgets.com [19.05.2010]

[42] Twitter Online available: URL: http://www.twitter.com/ [22.06.2010]

[43] Widget Packaging and Configuration. Online available: URL: http://www.w3.org/TR/widgets/ [23.06.2010]

[44] Widget Packaging and Configuration. Online available: URL: http://www.w3.org/TR/widgets/ [19.05.2010]

[45] Widgetbox. Online available: URL: http://www.widgetbox.com/ [19.05.2010]

[46] WidgetTop. Online available: URL: http://www.widgetop.com [19.05.2010]

[47] Windows Live Gadgets. Online available: URL: http://gallery.live.com/home.aspx [19.05.2010]

[48] XING Online available: URL: www.xing.com] [19.05.2010]

[49] Yahoo! Widgets. Online available: URL: http://widgets.yahoo.com/ [19.05.2010]

[50] Yourminis. Online available: URL: http://www.yourminis.com/ [19.05.2010]

[51] Youtube. Online available: URL: http://www.youtube.com [19.05.2010]