Lessons Learned from the Development of the ROLE PLE Framework

Sten Govaerts, Katrien Verbert, Evgeny Bogdanov, Erik Isaksson, Daniel Dahrendorf, Carsten Ullrich, Maren Scheffel, Sarah Léon Rojas, and Denis Gillet

Abstract Within the ROLE European research project, an interoperability framework has been developed to support self-regulated learning and to enable learners and teachers to create personal learning environments (PLEs). This framework enables learners to assemble tools, services and resources together to create their own custom learning environment. This chapter discusses the overall architecture, the specific components of this architecture and the platforms in which we have integrated the ROLE framework. Additionally, we share the lessons learned from the design and development. Furthermore, we discuss our experience with the

S. Govaerts (✉) • E. Bogdanov • D. Gillet
Ecole Polytechnique Fédérale Lausanne (EPFL), REACT, Station 9, Lausanne CH-1015, Switzerland
e-mail: sten.govaerts@epfl.ch, http://react.epfl.ch; evgeny.bogdanov@epfl.ch, http://react.epfl.ch; denis.gillet@epfl.ch, http://react.epfl.ch

K. Verbert
Technische Universiteit Eindhoven, Information Systems WSK&I, Postbus 513, 5600 MB, Eindhoven, The Netherlands

Katholieke Universiteit Leuven, Department of Computer Science, Celestijnenlaan 200A, 3001 Heverlee, Belgium
e-mail: katrien.verbert@cs.kuleuven.be

E. Isaksson
KTH—Royal Institute of Technology, Stockholm, Sweden
e-mail: erikis@kth.se

D. Dahrendorf
IMC Information Multimedia Communication AG, Altenkesseler Str. 17/D3, Saarbruecken 66115, Germany
e-mail: daniel.dahrendorf@im-c.de, http://www.im-c.de

C. Ullrich
Department of Computer Science and Engineering, Shanghai Jiao Tong University, 1954 Huashan Road, Shanghai 200050, China
e-mail: ullrich_c@sjtu.edu.cn

M. Scheffel • S.L. Rojas
Fraunhofer Institute for Applied Information Technology FIT, Schloss Birlinghoven, Sankt Augustin 53754, Germany
e-mail: maren.scheffel@fit.fraunhofer.de, http://www.fit.fraunhofer.de; sarah.leon.rojas@fit.fraunhofer.de, http://www.fit.fraunhofer.de

S. Kroop et al. (eds.), *Responsive Open Learning Environments*,
DOI 10.1007/978-3-319-02399-1_8

ROLE development infrastructure and our collaboration within the ROLE development team and with several open-source projects.

Keywords Development • Interoperability • Best practices • Lessons learned • Collaboration • Open source • Widgets • Web apps • Framework • Personal learning environment • PLE • Informal learning • Self-regulated learning • Social media platforms

Introduction

The proliferation of Web 2.0 technologies (e.g. wikis and social networks) has impacted the way users retrieve and use information and how they interact with each other (Maness 2006; Ullrich et al. 2008; Ashley et al. 2009). The abundance of Web-based tools and content creates many opportunities for Technology Enhanced Learning (TEL).

The ROLE project aims to exploit Web-based tools and technologies to empower learners to construct their own personal learning environments (PLEs). The overall goal is to create flexible, Web-based, open technologies for the federation and mash-up of learning services to empower the learner to build her own responsive learning environment. Responsiveness is defined as the ability to react to the learner needs—i.e. through recommendation, adaptation or visual analytics services that support the learner to be aware of and reflect upon her own learning process (Fruhmann et al. 2010). The project also targets critical transition stages of lifelong learning, e.g. due to shifts in learner interests or when leaving the university and entering a company. Chapter 1 elaborates more on the ROLE vision on PLEs.

Learning management systems (LMSs) such as Moodle, CLIX and Blackboard primarily focus on distributing learning content, organising the learning process and serving as interface between learners and teachers. Dalsgaard (n.d.) notes that in LMSs generally different tools, such as discussion forums, file sharing, whiteboards and e-portfolios, are integrated in a single system that bundles all tools necessary to manage and run courses. In contrast to PLEs, LMSs place a strong emphasis on how to centralise and standardise the learning experience (Guo et al. 2010). Learning activities in an LMS-based course are organised within a centrally managed system, which is driven by the needs of the institution. On the other hand, a PLE takes a more natural and learner-centric approach and is characterised by the free-form use of a set of services and tools that are controlled and selected by individual learners.

In recent years, research on mash-ups has been elaborated, for example widget mash-ups have been deployed at Graz University of Technology (Ebner and Taraghi 2010). In addition, researchers have focused on augmenting traditional LMSs with widgets to provide live-updating and flexible applications. Wilson et al. (2009) have implemented widget support for Moodle. Their big challenge is logging student activities with the widgets, as there is no communication between the widgets and the LMS.

The ROLE framework builds on this existing work, but incorporates additional core technologies such as inter-widget communication (IWC), automated user activity tracking, collaborative spaces and authentication and authorisation services to protect data. This is the basis to enable real-time communication between widgets and users, and to automate user activity tracking from tools and services. The analysis of such data and IWC provides the basis to develop responsive systems that can react to learner needs in a coordinated way.

Within the time span of the ROLE project, a new Apache project, called RAVE,[1] emerged with the aim to provide an extensible mash-up platform for using, integrating and hosting widgets with personalisation, collaboration and content integration features. The features of Apache RAVE and the ROLE project are quite similar, as confirmed by recent research that has been applying RAVE in educational contexts (Pierce et al. 2011; Chudnovskyy et al. 2012). Since the RAVE project started during the development of the ROLE framework, ROLE did not adopt Rave, but rather contributed components to the RAVE project (which is discussed in more detail in section 'Contributing ROLE Software to Open-Source Projects').

This chapter presents the ROLE interoperability framework, which is a technical platform to assemble widgets within responsive open learning environments. The framework allows the assembly of widget bundles with communication channels, authentication and authorisation mechanisms and services for activity tracking and analysis. The framework ensures that the widgets have access to the necessary information to react to learner needs. Furthermore, the platforms, on which the interoperability framework has been integrated, are discussed and the lessons learned from the design and development of the framework components are presented, as well as on the technical collaboration within the ROLE project and with open-source projects.

This chapter is organised as follows. First, the overall architecture of the framework is presented in section 'The Interoperability Framework', after which each component is discussed in more detail. Section 'ROLE Platforms' elaborates on the different platforms that integrate the ROLE infrastructure and the repository of widgets. Afterwards, the organisation of the ROLE developer community and our contributions to open-source projects are discussed. Finally, the achieved results are summarised and their dissemination is discussed.

The Interoperability Framework

The purpose of the ROLE interoperability framework is to support assembly of different widgets in responsive open learning environments. The architecture supports communication between widgets, authentication and authorisation

[1] Apache RAVE, http://rave.apache.org/

mechanisms, services for activity tracking and analysis and widget spaces, which manage widgets, resources and users. All these services can be accessed via open and if possible standardised interfaces. These are necessary for third-party developers who want to create applications based on ROLE technology. The next section details the overall architecture.

The Architecture

The ROLE architecture is illustrated in Fig. 1. IWC (see section 'Inter-widget Communication') is used and managed by spaces, but is also an autonomous component. It depends on JavaScript and the XMPP (Saint-Andre 2004a, b) protocol to provide a user-, community- and space-centred remote IWC. This allows developers to build powerful collaborative real-time learning tools and learners to assemble them easily in responsive open learning environments.

Tracking of activities is done via the Contextualised Attention Metadata (CAM) framework (see section 'Contextualised Attention Metadata'). An event-based schema was developed to model user behaviour in learning environments. Events are tracked and sent to either a central or container-specific repository. IWC is used to track such events. The data is stored and retrieved via an REST API.

As the CAM service contains sensitive data, an authorisation and authentication framework has been developed to protect this data (see section 'Authentication and Authorisation'). It is also needed for other ROLE services that handle personalised data. One of the main goals of this framework is to reduce the amount of user interaction by providing a Single Sign On (SSO) authentication mechanism.

Finally, widget spaces (see section 'Spaces') allow learners and instructors to create portable collaborative learning environments. Spaces consist of learners, configurable services and sharable resources, within a learning context. The space features can be provided either by a (OpenSocial) container itself or by a special widget. Such an approach guarantees container independence. Furthermore, widget

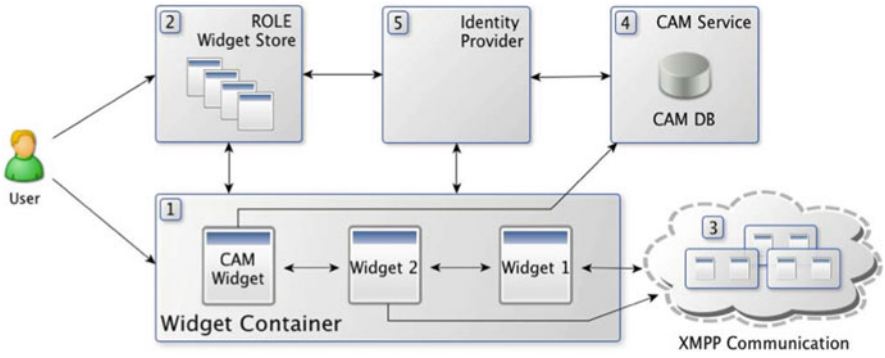

Fig. 1 The ROLE architecture

spaces provide a simplified single point of access to the other background services via an extended OpenSocial API.

Inter-widget Communication

IWC enables event-based communication between widgets following the Publish–Subscribe communication pattern (Birman and Joseph 1987; Eugster et al. 2003). We employ both local inter-widget communication (LIWC) within a PLE and remote inter-widget communication (RIWC) among different users, computers and PLEs.

LIWC is realised in the OpenApplication Event API (Isaksson and Palmer 2010, n.d.) using the HTML5 Web Messaging standard (Hickson 2011) available in most major browsers, including backwards compatibility for the Google Gadget PubSub mechanism. Instead of 'hard-wiring' widgets with each other (Sire et al. 2009), all widgets within a PLE are notified of all events and then decide autonomously to react accordingly. If the widget acts upon the received event, a receipt is sent back. Supporting containers that receive such a receipt can inform the user, e.g. by highlighting the tool that sent the receipt. The event payload format is designed for partial semantic interoperability, i.e. developers use a combination of established vocabularies in a simplified format with name-spaced properties (e.g. Dublin Core (DCMI Usage Board 2006)). In practice, this means that when an event is broadcasted, the originating widget does not indicate what receiving widgets should do (only the past action is specified, e.g. select). If the originating widget had to specify the intent (i.e. the desired future action), it would need to have buttons or menu items for every conceivable proposable action in every conceivable widget (e.g. add to portfolio, share with friends, search in Wikipedia). With events, we instead choose to split the job: events should be broadcasted for as many user actions as possible within each widget, without concern for what receiving widgets ought to do, and receiving widgets provide the affordances (e.g. buttons) for their own proposed further actions.

RIWC enables communication among widgets in different browsers and on different machines in order to foster real-time remote communication and collaboration functionality. RIWC is realised with the Extensible Messaging and Presence Protocol (XMPP) (Saint-Andre 2004a, b), an open standard for real-time communication. The power of XMPP lies in its built-in federation capabilities and extensibility through XMPP Extension Protocols (XEPs), such as for Publish/Subscribe (Millard et al. 2010) and Multi-User Chat (Saint-Andre 2008) as applied in responsive and collaborative learning scenarios (Friedrich et al. 2011). Since no JavaScript XMPP library with PubSub support was available, ROLE extended the dojo XMPP library by a set of common PubSub operations. Users can discover nodes, retrieve subscriptions, create, configure and delete nodes, subscribe and unsubscribe nodes and publish/receive IWC events in an XML-based payload format across a federated network of XMPP servers. However, current libraries

using XMPP over BOSH (Paterson et al. 2010) are not applicable in public containers such as iGoogle due to cross-domain issues. Furthermore, they are rather unstable and unreliable (Friedrich et al. 2011). Our experiments showed that the upcoming Web Socket API (Hickson 2009) for XMPP (Moffit and Cestari 2010) outperforms BOSH with considerable performance and stability improvements and availability in all containers.

IWC enables more responsive, collaborative environments with real-time notifications and richer user experience, although attention to usability is required (Isaksson and Palmer 2010).

Contextualised Attention Metadata

Tracking of user interactions with widgets is an essential part to enable responsiveness in learning environments. User interaction data can be used for data analysis and the computation of personal, social and contextual information about users and applications. Additionally, such data of the actual usage of ROLE services in real-world settings was used to evaluate the framework.

A variety of attention metadata formats exist. These formats differ in scope, expressiveness, scalability and context awareness. Butoianu et al. (2010) provide a survey of the following formats: TaskTracer (Dragunov et al. 2005), Swish (Oliver et al. 2006), CAM (Wolpers et al. 2006), the User Interaction Context Model (UICO) (Rath et al. 2009), the Context Modelling Language (CML) (Henricksen and Indulska 2006) and WildCAT (David and Ledoux 2005). TaskTracer and Swish are least flexible and expressive. UICO, CML and WildCAT are very expressive; however, the available frameworks using the formats do not scale well and some are focused on a specific application (Butoianu et al. 2010). On the other hand, CAM supports scalability and context awareness very well, but is less expressive. Other examples are the ActivityStreams specification[2] and the Experience API (Glahn 2013). The latter took inspiration from ActivityStreams and only became available in the last years of the ROLE project. ActivityStreams are less focused on contextual information. In ROLE, CAM is used because of scalability and context awareness.

The CAM schema (Schmitz et al. 2011) can be used to describe computer-related activities of one or several users—i.e. which objects attract user attention, which actions users perform and what the user contexts are. CAM was developed to describe as many types of attention metadata as possible. Therefore, CAM records of a user cannot merely describe user foci of attention, but rather her entire computer usage behaviour. Collections of CAM records can be exploited for generating diverse kinds of profiles like user profiles and object profiles (item profiles). CAM records represent user–computer-related foci of attention and actions and thus can instantly constitute profiles of individual usage behaviour.

[2] The ActivityStreams specification, http://activitystrea.ms/specs/json/1.0/

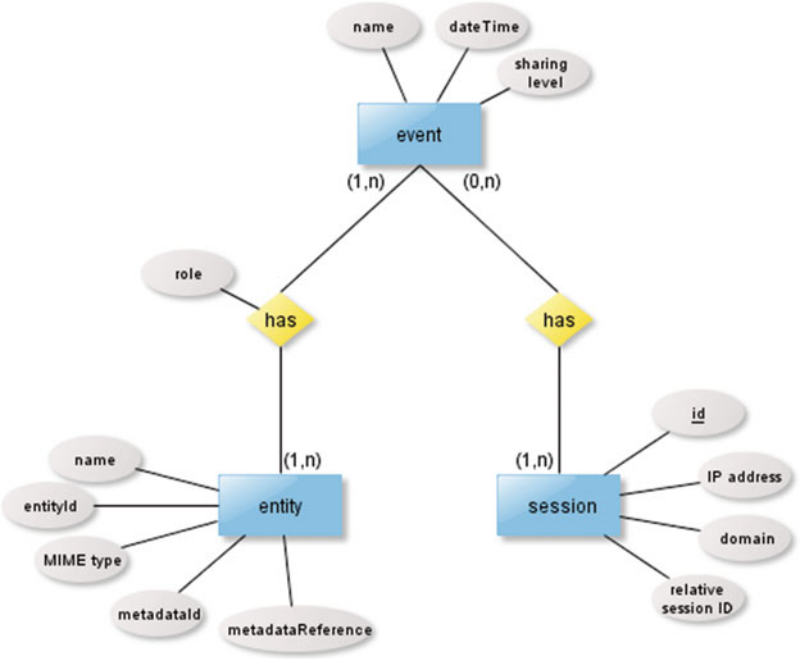

Fig. 2 Structure of the CAM schema

CAM records of different users can be exploited for generating attention and usage-based object profiles.

CAM records can be used to support self-reflection. For instance, visualisation widgets can support a user to recapitulate what she did and generate a picture of her competences. Furthermore, statistical metrics can be employed to aggregate and evaluate CAM records of different users. By this, general trends, for instance in computer usage, data consumption and communication behaviour, can be detected. Aggregated CAM records entail information on the behaviour of average users and on the behaviour of user groups. They also entail usage information on data objects, such as how often a certain object was used, and by which kinds of users and in what contexts it was used. In addition, they can reveal in which respect a user deviates from the average, whether her behaviour conforms to general trends or not, etc.

The CAM schema has been developed to provide a unified schema for monitoring data across system boundaries (Wolpers et al. 2007; Schmitz et al. 2011). The CAM schema has been transformed from a once user-centred version to an event-based version (see Fig. 2) that is better suited for evaluating and analysing user observations over time.[3]

[3] Information about the schema is available at https://sites.google.com/site/camschema/ and the CAM API: http://sourceforge.net/projects/camapi/

One major goal of ROLE is to provide personalisation, recommendation and self-reflection mechanisms. To achieve this, users can be monitored while interacting with their learning environment. The collected CAM information is used to generate different patterns and statistics, such as discovering learning trends and detecting what is currently happening in the learning environment. For an easy integration of the CAM monitoring into different learning environments, the monitoring architecture was divided into a client and a server component. The client component can be considered as a data collection element, responsible for accumulating and transforming the information into CAM, while the server component is responsible for the persistence management and data access control.

Figure 3 shows the CAM architecture applied in ROLE and how it works together with other ROLE components. The picture shows a platform, which uses ROLE technology by integrating ROLE widgets into their learning environment

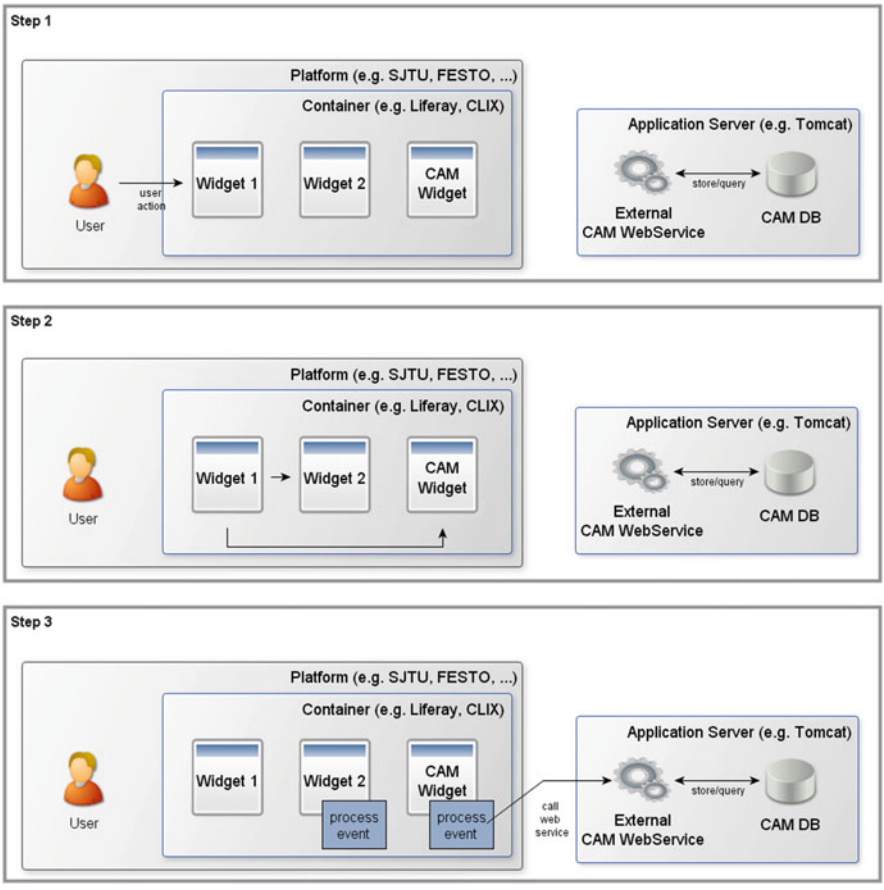

Fig. 3 The CAM architecture used in ROLE

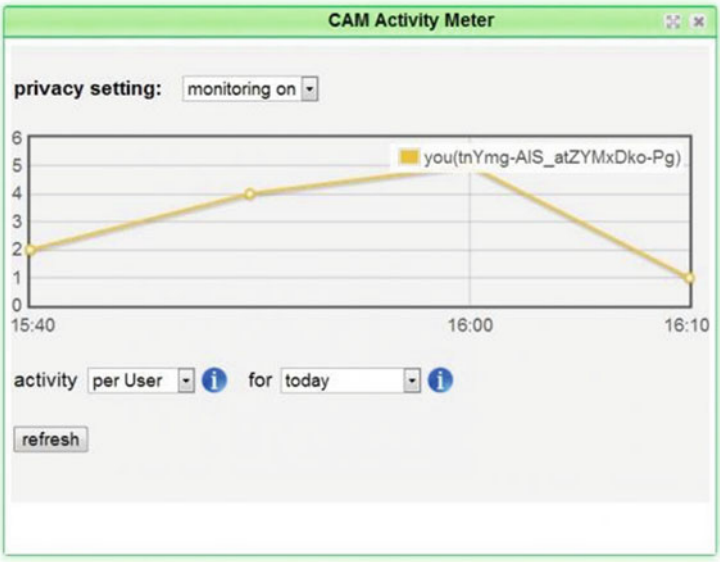

Fig. 4 Screenshot of the ROLE CAM Monitoring widget

(container). Furthermore, the picture illustrates that the CAM monitoring components must not be integrated into the platform since they are running on an external application server. First, a user performs an action on a widget (step 1), e.g. clicking a button. Since this widget supports IWC (see section 'Inter-widget Communication') this action causes an event, which is published via the OpenApp mechanism (Isaksson and Palmer 2010; Isaksson and Palmer n.d.). Thus the event is broadcasted (step 2) and can be received by all other widgets in the user's learning environment (local IWC, see section 'Inter-widget Communication'). Each receiving widget can process and react on the event (step 3). The CAM widget can thus receive all events sent through IWC. Afterwards the CAM widget identifies the event and forwards all required information to the CAM Web service, which is responsible for the CAM mapping and persistence. To offer the user an overview over the activities in the learning space, it contains a graph where past events from the users of the space can be displayed using different configurations, i.e. one or several users and several dates (see Fig. 4). As already mentioned, the picture illustrates that the Web service is not integrated into the platform, but is a standalone service that can be installed for a specific platform.

In addition to storage, the CAM Web service provides a query method, which consumes an arbitrary SQL select statement and returns the results in the JSON format. Using arbitrary SQL select statements ensures maximum freedom for the

developers to experiment with the data. Access to the monitoring data causes privacy issues. These issues have not been completely resolved in the ROLE project. Some steps have been taken though. For instance, data privacy can be supported through a stand-alone installation of the CAM Web service, e.g. by installing the CAM service in the intranet of an institution. The CAM widget also allows users to disable tracking (see Fig. 4). Additionally, the CAM Edit and Share widget[4] (see Fig. 5) allows the user to filter and export her CAM data from a specific space into an SQL file. This widget provides the user with full access and larger control over her CAM records, and she can share her events with others or analyse them herself. Finally, since the Web methods for querying and storing CAM are publicly available, the CAM service requires password-based authentication.

From our experience developing the CAM service and using it in real-world settings various new insights have been gained. One of the main benefits of the approach is that developers do not have to specifically write code to track user events, as the CAM widget will collect the IWC events automatically. On the other hand, this can also limit the data collection since not all important events might be requiring IWC. This tracking method also allows developers to be agnostic about the CAM schema and the implementation of the CAM Web service, since they do just need to send out IWC events. One of the strengths of the OpenApp IWC is the openness of its data format, which has no mandatory fields and enables developers to transmit any kind of JSON data from one widget to another. This freedom makes the mapping of the OpenApp events into CAM harder as such mapping cannot rely on certain elements to be present. Defining a subset of fixed mandatory IWC fields and fixing taxonomies of event types would make the CAM mapping more easy and robust.

Another flexibility issue is due to the high abstraction level of the CAM schema to allow all kind of events to be stored. This can generate a large number of different CAM event mappings, which can make it harder to analyse and compare the CAM records. The different IWC events stored information in different CAM fields, making the data analysis more complex without knowing the details of the intrinsic mapping. The ActivityStreams specification provides an extensible, common vocabulary list of actions that would aid this and when applied properly could provide more portable data (Vozniuk et al. 2013).

Currently, user activities are only tracked when the CAM widget is added to the learning environment. Hence, this enables users to have full control over where their activities are tracked, but also causes that nothing is recorded if the user forgets to add the CAM widget. This problem could be circumvented by integrating CAM monitoring directly into the platform.

[4] http://www.role-widgetstore.eu/tool/cam-edit-and-share-widget

Fig. 5 Screenshot of the ROLE CAM Edit and Share widget

Spaces

In the ROLE framework, a PLE can consist of various spaces. A space is an abstract concept that materialises the user's context and aggregates people, resources, applications and other subspaces. All these artefacts belong to the same activity that a person (or a group of them) is working on to achieve a common goal. This common goal is the purpose to create the space. Various people can participate in a space and might have different access rights and roles within this space, where they share resources and applications that they need to achieve their goal. A space might have subspaces that help hierarchical organisation of resources, applications and people. A space can be seen as a PLE unit. On the one hand, a space is a way for users to give shape to their PLEs by aggregating information. On the other hand, a space allows users to share their PLEs with others by inviting them to collaborate.

The space concept exists in all ROLE platforms (see section 'ROLE Platforms'): the ROLE SDK (see section 'ROLE SDK'), Graasp[5] (Bogdanov et al. 2012a) (see section 'Graasp'), the OpenSocial Moodle plugin (Bogdanov et al. 2012b) (see section 'OpenSocial Moodle Plugin') and the Widget Store (see section 'The ROLE Widget Store'). Every platform internally implements this concept in a different way. In order to allow widgets to use the information about the space and to enable integration and portability of spaces between the platforms and beyond, we applied two approaches: Linked Data and OpenSocial.

For the first approach, we created a space ontology for Linked Data. Linked Data provides a very powerful and extensible way of describing data in a machine-understandable way. It targets the discovery and integration of data originating from different sources. Due to its design, it has limitations. First, Linked Data and SPARQL require a rather steep learning curve, which is a disadvantage compared to simple RESTful APIs that are used by many Web developers.[6] The second limitation is the performance. Since the data is located on different servers, many HTTP requests have to be issued to retrieve the complete data. Moreover, the SPARQL engine requires traversal of a graph, which is much slower than retrieving data from a relational database. The authors Health and Bizer (n.d.) (see section 6.3) foresee the use of data crawling for real-time Linked Data applications, rather than on-the-fly URI dereferencing.

The alternative approach we used is OpenSocial. The OpenSocial specification consists of three main parts. The first part describes the widget standard. The second part standardises the model for social network elements (i.e. Person, App and Document) and relations between them. The third part standardises a set of common REST and JavaScript APIs to retrieve data from a social platform. Since the space concept did not exist in OpenSocial, we introduced it into the OpenSocial specification. The OpenSocial Space extension standardises the space model (namely a list of fields that a space can contain), and the REST and JavaScript

[5] Graasp, http://graasp.epfl.ch

[6] Linked Data API, http://code.google.com/p/linked-data-api/wiki/API_Rationale

APIs to work with spaces (Bogdanov et al. 2011). Through this extension, widgets can retrieve information about their containing space and its content via OpenSocial APIs. For example, the Person API can be used to retrieve all the members of the space. The widget can retrieve a list of the resources and widgets available in the space via the Document and App APIs, respectively.

The main disadvantage of OpenSocial is that the social model cannot be easily and arbitrarily extended as with Linked Data. The new extensions require going through the process of standardisation, which can be quite cumbersome. On a positive note, OpenSocial provides easy-to-use REST APIs with a JSON-based data representation. The data format and APIs are standardised, which enables interoperability when data is accessed and processed. OpenSocial does not target data discovery (as Linked Data) but rather data retrieval and exchange. Since the data is often centralised in one institution, it is very fast to retrieve the data compared to the SPARQL engine.

Authentication and Authorisation

CAM contains sensitive and personal data protected by law. Additionally, users might prefer to keep the content of their spaces private. The data access has to be trusted and allowed by the users. The data communication occurs at two different levels: service-to-service and widget-to-service communication.

Service-to-service communication can occur when for example a recommendation service requires CAM relevance feedback on resources. Data can be transferred across institutions and countries with different laws. Thus, we decided to leave the decision of service-to-service authentication and authorisation (A&A) up to the service developers.

Widget-to-service communication occurs when for example a self-reflection widget wants to query the CAM service. This has been implemented as follows. A user is authenticated as being the owner of a particular personal space. The user first authenticates as being the owner of a separate identity, to which the personal space is linked (or a new personal space will be created), which thereby implies that the user is the authenticated owner of the space. The personal space then functions as the identity of the user.[7] Authentication is typically done via OpenID, which is the standard for decentralised authentication on which we have focused, but other protocols may be supported as well (one test bed, at Uppsala University, has implemented support for CAS while also keeping support for OpenID authentication).

Delegated authorisation is used by widgets that access collaborative and personal spaces. Furthermore, such delegated authorisation may also be used by third-

[7] The identifier of the personal space, i.e. a URI, is used as the identifier of the user, cf. WebID (see http://www.w3.org/wiki/WebID)

party services. The standard for delegated authorisation that we have focused on, i.e. OAuth, lets the currently authenticated user choose whether to authorise the widget or service. After authorisation, the service is provided with a token granting access to spaces on the user's behalf. For widgets, the token is managed by the OpenSocial widget container, which allows widgets to perform requests through the engine's OAuth proxy that first applies OAuth and then forwards the request.

Currently, OAuth endpoints for OpenSocial widgets must be hard coded in the widget's source XML. Therefore, spaces implement a rewriting of the XML so that the proper endpoints are included. Otherwise, it would be necessary for widget developers to maintain separate XML files of their widgets for every server where the widget is deployed. Widgets using the rewritten XML files, however, can be added to any widget container, such as Liferay[8] (Yuan 2009), while still maintaining the connection to their respective spaces.

ROLE Platforms

The ROLE interoperability framework has been integrated in various platforms. This section describes these platforms where users create and use their PLEs and search for widgets.

The ROLE SDK

The ROLE SDK is a collection of software and tools, which allows trying out ROLE technology and developing new widgets for mash-up PLEs. In total, ten versions of the ROLE SDK were released, each one packaging the implementation outcomes of one milestone of an iterative development process (see section 'The ROLE Developer Community').

The central core of the SDK is the reference implementation of a sample PLE, which allows using ROLE technology in practice and developing new widgets at the same time. Within the ROLE SDK, a learning space functions as a collaborative context for learning, consisting of a bundle of widgets, along with a list of participants. Widgets can interact with other widgets, and participants can interact with other participants, by using widgets and built-in features of the ROLE SDK (e.g. chat functionality).

A personal space is defined as a personal context that consists of a person's user model. One representation of the user model is a user profile; another representation, also based on the user model, is a bundle of (personally chosen) widgets. In the

[8] Liferay, http://www.liferay.com/

ROLE SDK, the learning space and personal space are combined in one user interface.

Additionally, there is a social context. The social context offers access to the communities of which the user is a member. The specific community that is accessed typically depends on the website where the widget is currently being used, which may be a different website than that of the PLE. For instance, if a widget is being used on a social networking website, the community would be that of the website. Collaborative contexts (i.e. learning spaces) can transcend social contexts. A widget may be part of a learning space, and at the same time be used on a social network for inviting people from that network to participate in the space. OpenSocial standardises APIs for access to what is defined as the social context here.

Furthermore, the concept of activities was introduced. Activities can be defined as purposes for which the user structures her learning context and assembles widgets. In the ROLE SDK, activities are displayed as one group of visible widgets at a time. In the GUI, activities can also be tabs or pages. However in the ROLE SDK, the additional semantics that the term 'activities' offers is covered. The term hints to the user that the groupings should be used for focusing on one activity at a time (such as training English vocabulary or searching for Web resources), using the tools that are appropriate to that activity, without being distracted by what is unrelated to the activity.

As mentioned before, the ROLE SDK relies on the concept of spaces. While a space is, at its most basic level, simply a bundle of resources such as widgets, there are several aspects that contribute to its usefulness as the basis for a PLE:

- *Aggregation*: Widgets (or more generically, tools) are bundled with any other kind of resources that contribute to the space's goals. The model enables a very flexible use of spaces, without requiring modifications in the model or its implementation.
- *Contextualisation*: A space forms a context for its contained resources. Widgets can be made context aware, and are then able to interact with the space and its resources. In addition to being in the context of a space, widgets can be contextualised further by being given configuration that is specific to their instantiation within the space.
- *Participation*: People can join a space, which means that they become members of that space. Members are notified of the presence of other members, and can interact with them both asynchronously and synchronously.
- *Personalisation*: Spaces offer a level of customisability, so that users are able to personalise the environment according to their needs.

These design requirements are realised within the ROLE SDK. Being a sample implementation of a PLE platform, the space user interface (see Fig. 6) is a Web application composed of four parts. First and on top, the header element is implemented as a top-aligned bar. It provides elements for controlling the Web application as a whole, such as signing in and out, and navigating to other parts of the application. Secondly, the sidebar element is a narrow, fixed-width section,

Fig. 6 The ROLE SDK user interface of a learning space

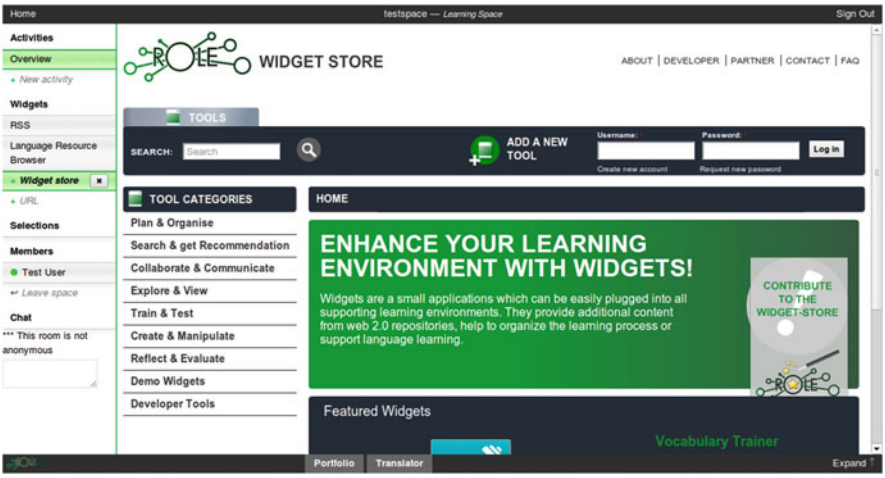

Fig. 7 The ROLE Widget Store, embedded in the ROLE SDK

running along the left side. It provides elements for controlling the space, such as switching between activities, and adding widgets.

Thirdly, the container element comprises the central area of the user interface, not covered by the other parts. This is where the main content is located, typically the space's widgets. The container can also be used as an embedded browser, which is how the ROLE Widget Store (see section 'The ROLE Widget Store') is integrated (see Fig. 7).

The fourth and last element is the dashboard, a bottom-aligned bar when collapsed. Expanding the dashboard displays the widgets on the personal space

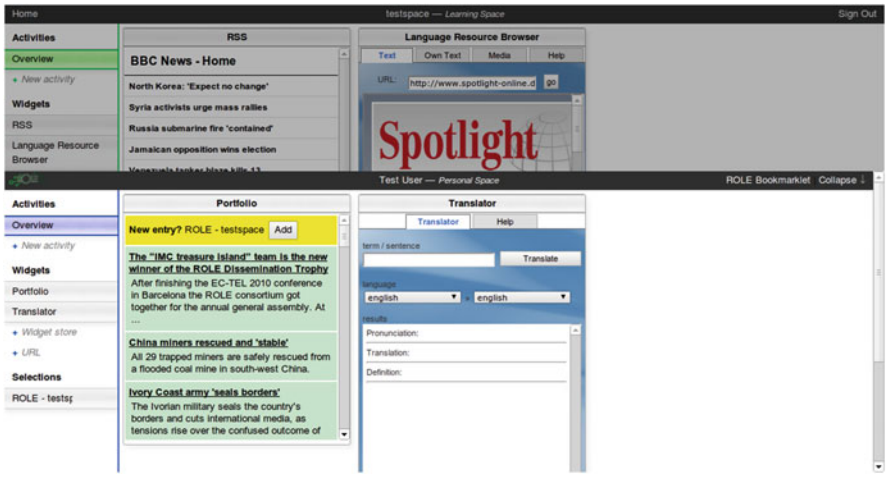

Fig. 8 The personal space, inside the expanded dashboard

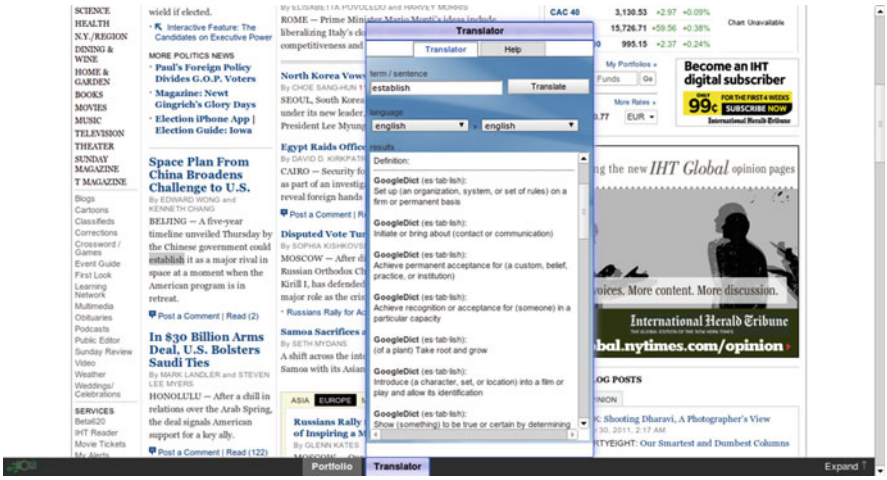

Fig. 9 The personal space, being accessed on a third-party website

right above the container, as illustrated in Fig. 8. The dashboard provides access to the user's personal space, which is a space that is private to the authenticated user. It is available from any other space (and other pages of the Web application) as well as from third-party websites via a bookmarklet (cf. Fig. 9).

In the case that the space itself is embedded (e.g. on the course page of an LMS), it is intended that the parts can be hidden or moved (e.g. the Header), because their functionality (e.g. sign in) can be already provided by the LMS or to adhere to another design.

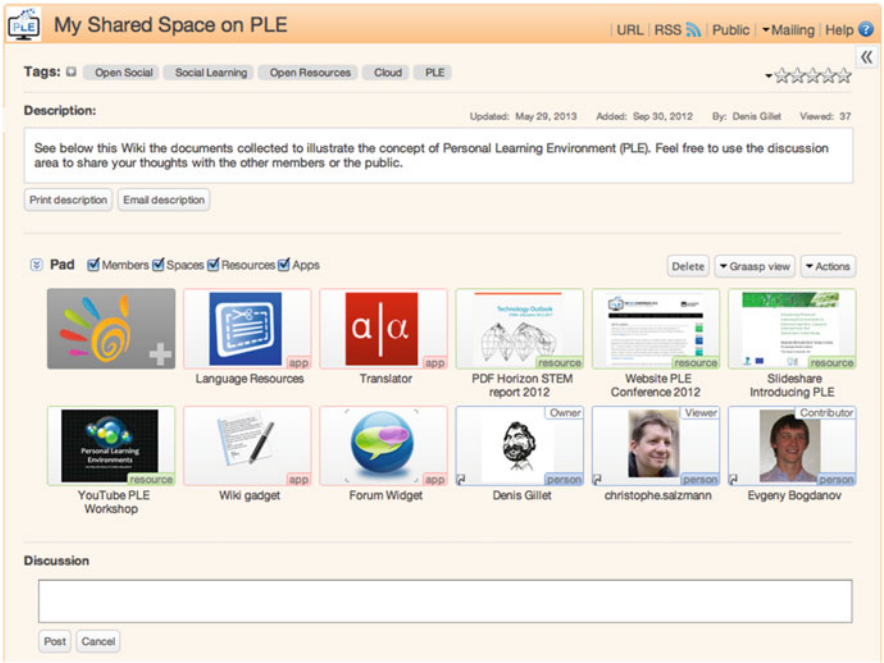

Fig. 10 A shared contextual space created in Graasp that integrates resources gathered from the Cloud, such as YouTube videos, SlideShare presentations, OpenSocial Widgets, Web pages or PDF documents with previews

Graasp

Graasp[9] (Bogdanov et al. 2012a) is a social media platform for collaborative learning and knowledge management (see Fig. 10). Graasp implements the OpenSocial space specification (see section 'Spaces'), which enables the creation of spaces shared between people belonging to different communities and networks. Embedded shared resources are gathered across institutional and corporate boundaries. Unlike dominant social media, Graasp enables a fine definition of the audience, as well as the associated rights and roles to ensure trust and privacy enforcement. In Graasp, people map their personal and shared projects, interests and activities into public or private contextual spaces integrating invited members, relevant resources and necessary apps which can be tagged and rated. Any space or resource in Graasp integrates its own discussion thread to enable contextual interaction. Graasp allows learners to construct and manage their own PLEs. Users can create a PLE for each learning objective, populate it with various resources and tools, personalise it and share it with others (Bogdanov et al. 2012a).

[9] Graasp, http://graasp.epfl.ch

The space concept is at the core of Graasp. A space can represent a PLE and can contain four types of entities: people, resources aggregated and used within the space, apps added to the space to extend its functionality and subspaces to organise the space content in a hierarchical structure. Graasp enables users to manage their spaces.

The resources and apps can be aggregated into a Graasp space from both local and remote locations. First, users can easily drag and drop files directly from their desktops into their spaces. Second, remote resources from the Cloud can be easily aggregated via an aggregation mechanism called GraaspIt!. Whenever a user encounters an interesting page, she can simply click on the GraaspIt! bookmarklet in the browser and the resource will be added into a space (Gillet and Bogdanov 2012). These collected resources can be aggregated as URLs, embed tags or Web page screenshots. In addition to resources, Graasp allows users to aggregate widgets into their spaces. Currently, the OpenSocial widget standard is supported, though other standards (e.g. W3C widgets) can be added in a similar way. Such widgets either can be added manually or can be aggregated from existing widget repositories. For example, when the user is browsing through widgets in a widget repository (e.g. the ROLE Widget Store, see section 'The ROLE Widget Store'), a widget of interest can be added by just clicking on GraaspIt!. The second way is to add widgets from the ROLE Widget Store by exploiting the widget repository search mechanism provided within Graasp.

Once a space is created and populated in Graasp, the core part of the interface (see Fig. 10) enables users to interact with the aggregated content and can be further personalised with the concept of functional skins (Bogdanov et al. 2011). A functional skin is a client-side plugin for a space that can retrieve space data via the OpenSocial APIs and provides users with visual and functional features different from Graasp and tailored to specific needs. For example, in addition to the standard view of Fig. 10, Graasp offers two built-in functional skins: the resource view and the app view. The resource view displays a list of all resources that exist in a space and provides download links and presents resource previews. The app view displays all widget instances from a space as a visual mash-up. In this view, widgets can be resized and their order can be modified through drag and drop. The possibilities to personalise the space are extensive and through functional skins the users can further adapt their spaces for their own professional/personal tasks.

Graasp implements several mechanisms to share and exchange spaces and widget bundles (sets of widgets combined together for a specific purpose). An app bundle can be extracted from the existing space and exported as an OMDL file.[10] The OMDL file can be imported into another platform (or reused in Graasp) or shared at the ROLE Widget Store. Additionally, a space created in Graasp can be shared with other people and with other platforms. The space can be extracted from Graasp as a secret URL. This URL can be given to other users and allows them to collaborate anonymously. Alternatively, the space can also be embedded into

[10] Open Mashup Description Language, http://omdl.org

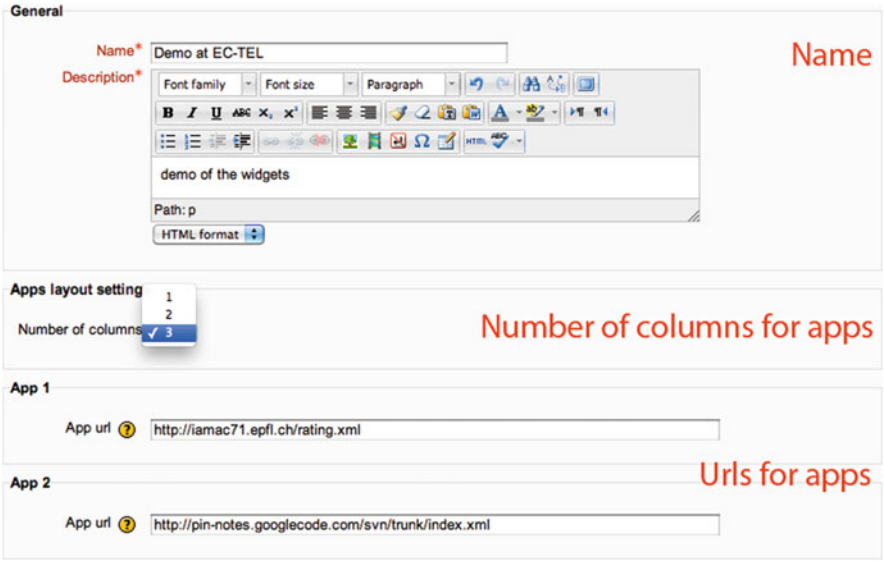

Fig. 11 A teacher creates a space with widgets for a course

another Web platform. Chapter 5 provides more information on how Graasp was used and evaluated for formal learning.

OpenSocial Moodle Plugin

The OpenSocial Moodle plugin (Bogdanov et al. 2012b) enables the use of OpenSocial widgets within the Moodle LMS[11] to create PLEs. By providing support for PLEs in an existing LMS, the disruption often caused by providing users with completely new environments decreases. By integrating PLE features in an existing LMS, users can still continue to use the features of the familiar LMS, but can personalise their learning environment with widgets.

The OpenSocial plugin for Moodle exists in two variations. The first version adds a new module to Moodle, which is displayed in the central area in the Moodle UI.[12] The module allows a teacher to add a 'widget space' to the Moodle course page, specify a set of widgets and choose the widget layout on the Moodle page (see Fig. 11). After this configuration, students can work with several widgets simultaneously (see Fig. 12) in the Moodle course.

[11] Moodle, http://www.moodle.org

[12] OpenSocial Moodle module, https://github.com/vohtaski/shindig-moodle-mod

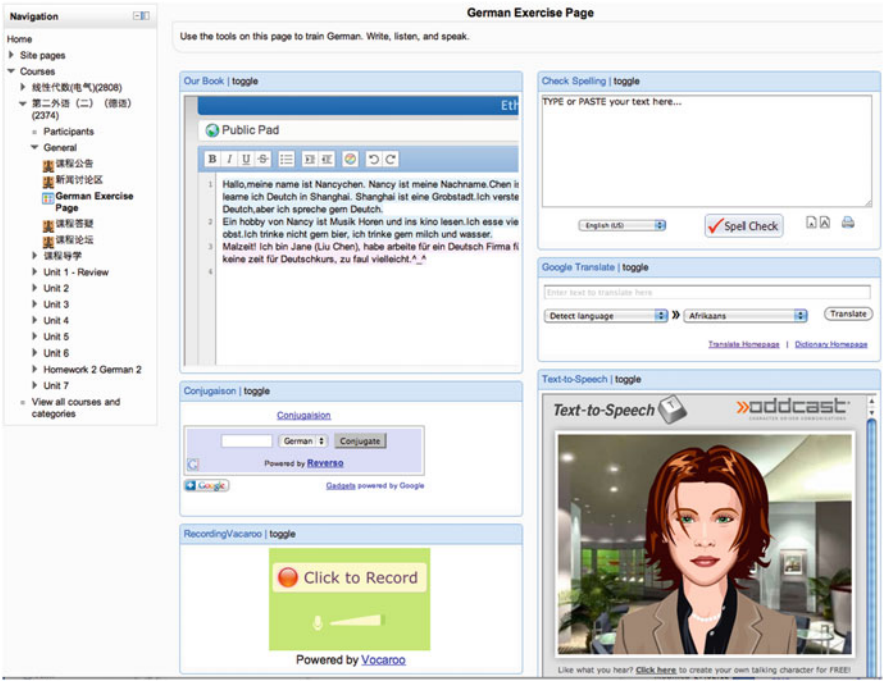

Fig. 12 OpenSocial widgets displayed within Moodle

The second version[13] of the OpenSocial plugin adds a new block in the right column of the Moodle UI. With this Moodle block, the teacher can add widgets to the right column of existing Moodle pages. Both versions of the Moodle plugin make use of the Apache Shindig engine,[14] which provides an open-source implementation of the OpenSocial specification, to render and manage widgets.

One of the main benefits of these plugins is that they enable teachers to easily extend Moodle with new features and services provided by widgets. Consequently, once the OpenSocial plugin is installed in Moodle, a teacher can append the required functionality herself, without the intervention of system administrators. The plugin enables the flexibility of selecting the resources and tools required for a specific course. Additionally, the plugins enable reuse of existing educational resources and tools. Furthermore, teachers and students can continue to operate in the learning environments they are familiar with but gain the mash-up features of PLEs. Naturally, the components of the ROLE architecture are compatible with the Moodle plugin. For instance, IWC and CAM are fully operational in the Moodle plugin. By extending widely used LMS with PLE features, we aim to achieve a faster adoption of the PLE paradigm among institutions. Further details on how this

[13] OpenSocial Moodle block, https://github.com/vohtaski/shindig-moodle-block

[14] Apache Shindig, http://shindig.apache.org

Moodle plugin was put to use for formal learning and evaluated are available in Chap. 4.

The ROLE Widget Store

The ROLE Widget Store allows users to search and browse for widgets and compilations of them. The store addresses the issue of categorisation, browsing, searching and recommending by providing various widgets categorised based on functionality, learning phases and learning domains. Further, the Widget Store enables sharing of platform-independent PLE and templates composed of learning tools and artefacts (or the so-called Widget Bundles). Via these mechanisms, the Widget Store fosters the development of a community of practice to exchange learning tools. Regarding the widget bundles, the store provides features to apply and share bundles across different learning platforms. This section further discusses different recommendation strategies and the interfaces that enable interoperability are specified (LMS/PLE system integration). Figure 13 presents an overview of the Widget Store architecture.

The main focus of the store is to provide a catalogue of widgets by supporting two commonly used widget specifications: the W3C widget specification (Caceres n.d.) and the OpenSocial specification (Mitchell-Wong et al. 2007). Developers can post either their self-developed widgets or widgets based on licenses which allow further distribution. Where possible metadata are automatically extracted from the

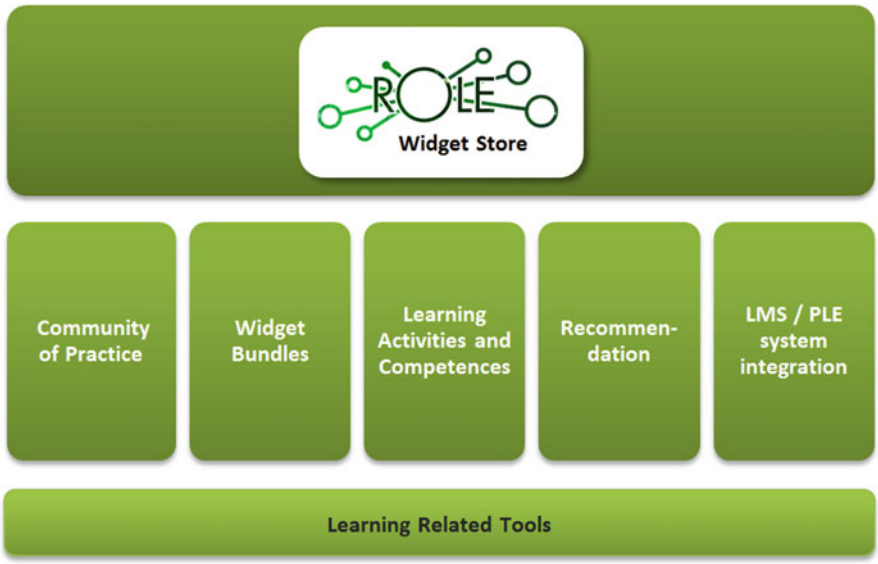

Fig. 13 Components of the ROLE Widget Store

widgets. Widget bundles are compilations of widgets, which are created to share good practices of widget use in learning environments. They are intended as a fast and simple way to provide learners with tools, services, content and a detailed description of how to use these to complete a specific learning task. Learners can select several tools from the store to create their learning environment. Additional references to learning resources can be added. For each tool and resource, learners are able to add learning activities in order to describe what should be done using the tool or learning resource. Once such a bundle is created by a learner, she can share it with the community. Such bundles can be reused by teachers and learners for their learning environments. In order to support learners in selecting applications for their PLEs three different categorisations are provided.

- Tool categories are derived from the Psycho-Pedagogical Integration Model (PPIM) (Fruhmann et al. 2010) (more information on PPIM is also available in Chap. 2), so users can select widgets supporting different learning phases.
- Tool functionalities represent features of widgets (e.g. text editing, video chat) and are based on an ontology developed in ROLE.
- Learning domains describe, if possible, the domain of the tools by providing semantic tags using DBpedia.[15]

The categorisation of bundles differs from the tool categorisation. A bundle can be designed to cover several phases of the PPIM model and thus refers to several tool categories. The approach of the Widget Store is that a bundle automatically inherits functionalities of tools it contains and can be tagged manually by learning domains from the DBpedia.

To provide an interface for external systems, the ROLE Widget Store offers an SPARQL endpoint which allows retrieval and insertion of the data of the Widget Store based on a standardised interface. Furthermore, different formats (Turtle, RDF/XML and JSON) are supported so that developers can choose their preferred data format. Another possibility for PLE platforms to integrate the store is to embed the store in the learning environment. The embedded version provides a simplified user interface and offers buttons that allow users to directly choose widgets to assemble their PLE. The store is connected to other ROLE components in the following ways:

- Graasp queries the store to provide a catalogue of widgets enabling easy integration of widgets in Graasp spaces.
- The ROLE SDK embeds the store and uses the embedding features to add widgets to the ROLE spaces.
- The ROLE Pedagogical Recommender (Nussbaumer et al. 2012) queries the store to provide recommendations based on the ontology of learning activities and the store categorisations.

[15] DBPedia, http://dbpedia.org/

- The ROLE Requirement Bazaar[16] (Renzel et al. 2013) uses the data to support the requirements elicitation and negotiation process being part of the ROLE Social Requirements Engineering approach. The ROLE Requirement Bazaar is a collaborative social platform where users can illicit their needs and wishes to developers who can extract requirements for future implementation.

Widgets and Tools

One major problem regarding the adoption of ROLE in new test beds and increasing the number of users was the limited number of widgets that were available.

One approach to overcome this problem is to enable a very simple transformation of existing Web resources into widgets (Ullrich et al. 2013). This transformation can be done by developers through the usage of widget templates as well as by non-technical people with the help of an authoring tool. Both solutions support the same ROLE technologies, namely the capturing of interactions via CAM and the possibility to rate the widgets. Interactions are captured on a very generic level: basically, whenever a student uses a Web application integrated into a widget for a period longer than five seconds, then the widget sends out CAM event of the type 'used'. Of course, a developer can refine the interactions, when required.

The 'widgetisation' of a Web application is simplified through the usage of a template and through JavaScript libraries that can be included in (existing) widgets. The template defines a widget that embeds the Web application via the iframe HTML element. This has the advantage that the original Web application does not need to be modified. In case, a widget of the Web application already exists, the capturing of interactions via ROLE can be enabled by the inclusion of the JavaScript library. This library uses IWC to send out the captured interactions, which can be made persistent on the CAM service via the CAM Monitor widget as described in section 'Contextualised Attention Metadata'.

The proposed approach has been implemented and a widget template is available in which the widget developers have to add the link to the Web page they want to integrate. To extend an existing widget, one has to include several lines of JavaScript code. The generation can be automated by using a set of shell scripts. The scripts take a list of URIs as input and generate widgets for the URIs. This reduces the authoring time to less than a minute. In summary, while this solution works very well for advanced software developers, it is still too complex for the average user. As an example, the code to extend an existing widget with ROLE technologies looks as follows:

[16] ROLE Requirements Bazaar, http://role-is.dbis.rwth-aachen.de:9090/BazaarFrontend/index.html

```
//Load two libraries for allowing the user to rate this gadget//and for
capturing interactions in CAM format $.getScript("http://widgets.
onlinesjtu.com/gadgets/libs/rating.js",

  function(){
    $.getScript(
  "http://widgets.onlinesjtu.com/gadgets/libs/interactioncapture.
js",
     function(){
      var rating = new ROLE_module.rating
                  ("#importedGadget");
       var interactioncapture =
         new ROLE_module.interactioncapture
                ("#importedGadget");
        })
  }
);
```

The first lines load the libraries. The functionality is activated by creating the appropriate objects. In the example, #importedGadget specifies the HTML element to which the interaction capturing and rating functionality should be attached (typically a div element, which is the parent of the iframe element).

In addition to the simplification of the usage of the libraries, the Shanghai Jiao Tong University (SJTU) created an authoring widget that allows teachers without Web development expertise to generate widgets from existing Web resources. The authoring widget asks users to input the URI of the Web application and add some metadata. Then, the authoring tool generates and uploads the widget to a server. Through an integration with the OpenSocial Moodle module (see section 'OpenSocial Moodle Plugin') users can create widgets without having to leave the learning environment.

Thanks to these tools, SJTU was able to create several hundreds of very domain-specific widgets for ROLE. Additional details on the SJTU test bed are presented in Chap. 4.

ROLE and Open-Source Developer Communities

All technical partners of the ROLE project have been collaborating successfully to create the ROLE framework. To foster this collaboration, various support mechanisms were set up, consisting of management structures, sub-projects, development software and developer meetings. This section elaborates on the developer collaboration within the ROLE project and with open-source projects to disseminate ROLE technologies.

The ROLE Developer Community

Technical cluster structure: To enable the assessment of requirements, exploration of technologies, creation of early prototypes and their evaluation, the development process was split up in consecutive sub-projects, each having its specific goals and deadlines. In total there were five of such projects: the Christmas project (ended on Christmas 2010), the Easter project (ended on Easter 2011), the Stonehenge project (ended on December 22, 2011), the Gunpowder project (ended on January 31, 2012) and the Shori project (ended on January 31, 2013). By defining use cases and goals for each project, the requirements and planning were defined. These projects also allowed easier planning of evaluations. The longer projects (i.e. the Gunpowder and Shori project) had a more elaborate planning phase and management methodology. For the Gunpowder and Shori project we aimed to apply the SCRUM (Schwaber 2004) and Kanban (Ladas 2009) methodology. But due to the large, geographically dispersed team from different organisations, we opted for an adapted version of SCRUM combined with Kanban, where one or two persons would manage the development process and report progress to the ROLE general assembly. The requirements and goals for the projects were often laid out in a face-to-face developer meeting or developer camps (see below) and follow-up virtual meetings. The project managers would then plan milestones (or sprints) often based on evaluation deadlines and showcases at conferences. Certain topics had smaller teams working on it in task forces, e.g. assessing a solution for authentication or CAM. Bi-weekly technical virtual meetings were organised to discuss progress, and to decide on technology and architecture choices. This setup allowed all developers to work on their own tasks and be involved in the decision making, but also get an overview of the current project status. Furthermore, it allowed the project managers to follow up the progress and react quickly where needed. This approach was received positive by developers, managers and general assembly. Next to this methodology, the development was also assisted by software.

Development software: To support the developers and the management, several software packages were set up. To provide access to and version control our source code we experimented with Git[17] and Subversion (SVN).[18] Initially, Git on GitHub[19] was used, but at that time GitHub did not fulfil the requirements of the project. To reduce the complexity, the source code was migrated to Subversion.[20] At the end of the project, the source code was migrated again to GitHub,[21] since GitHub has a more flexible scheme where any external developer can reuse the source code without any intervention from the repository owner. Whereas

[17] Git, http://git-scm.com/

[18] Subversion, http://subversion.tigris.org/

[19] GitHub, https://github.com/

[20] The Subversion repository is available on Sourceforge at http://sourceforge.net/projects/role-project/

[21] The ROLE GitHub repository is available at https://github.com/organizations/ROLE

Sourceforge still requires management by a ROLE partner. To manage the projects, milestones and bug and issue tracking, Atlassian JIRA[22] was used. Tasks, feature requests and issues were collected in JIRA and assigned to projects and milestones. Open tasks and issues were discussed in the technical meetings. Overall, our JIRA experience was quite positive, as it enabled a quick overview of the progress and future work for developers and managers. Clearly, to have a consistent and up-to-date overview, developers have to be committed to report their work in JIRA.

Developer camps: During the project period, three developer camps were organised. Originally, the developer camps were meant for internal developers to discuss the overall ROLE architecture and technical solutions, and plan the projects. During the first developer camp, a shared vision of the ROLE objectives was created. At the second and third developer camp, external experts were invited to provide feedback on the architecture, identify missing use cases and requirements and provide a broader scope on recent research results that could be applied in ROLE. At the third developer camp (November 2011), we invited a larger group of experts, presented the current status of the ROLE framework and had a small developer competition to develop widgets for the platform. This was only possible at this time, because the implementation was mature enough. This developer competition was good both for dissemination to research and open-source projects (e.g. Apache RAVE), and for getting feedback from external developers on the ROLE APIs and documentation. Later we organised four more widget competitions that were open to the public. In general, the developer camps were a good platform to collaborate with the whole ROLE technical team and external experts for a couple of days.

Contributing ROLE Software to Open-Source Projects

Several components and specifications of the ROLE framework were integrated in other open-source projects. This strategy enables further uptake and development of the research results of the ROLE project. This section highlights some of the contributions to the open-source community.

OpenSocial and Apache Shindig: In order to standardise the OpenSocial space extension (see section 'Spaces'), EPFL worked with the OpenSocial community for the specification and with the Apache Shindig community for the reference implementation of the specification. The communication with both communities happens through mailing lists. After our specification proposal was presented on the mailing list, it received very positive feedback and representatives of several companies showed interest in the extension for use in their products. After several discussions and refinements of the proposal, the work on the specification draft started. Typically, the procedure to get a proposal accepted is as follows: First, a patch to the

[22] Atlassian JIRA, http://www.atlassian.com/software/jira

OpenSocial specification has to be written. Second, the proposal has to be implemented in an open source, publicly available platform, e.g. Apache Shindig. Finally, when the proposal is finalised, the community votes on the final inclusion of the draft into the specification. Consequently, ROLE wrote a patch for the OpenSocial specification and extended Apache Shindig, which was shared with the OpenSocial community.

When we started our proposal, the OpenSocial community was finalising OpenSocial version 2.0. Thus, initially our proposal would be incorporated in the next version, 2.5. However, later the decision of the community was to have only limited changes in 2.5 and leave all larger revisions for the upcoming version 3.0. Hence, due to the large changes that our proposal would cause, it was decided to postpone its inclusion and it was only incubated in OpenSocial 2.5. Because of other changes in the specification of OpenSocial 3.0, our proposal had to be adapted. Eventually, the process that seemed open and efficient turned out to be quite time-consuming. Currently (December, 2013), the proposal is still on the road map for the OpenSocial 3.0. The patch for the specification is ready and the code for Apache Shindig is available. Once the work on the OpenSocial 3.0 is started, the patch should be evaluated and voted upon final inclusion into the newest version of the specification.

However, adding a proposal into the specification does not immediately guarantee that it can be used in all OpenSocial platforms. To be able to use the space proposal in widgets and to enable interoperability with other OpenSocial platforms, all platforms have to implement the latest version of the specification. There can be latency, since it takes time to upgrade to newer versions of OpenSocial.

Apache Rave: As mentioned in the introduction section, Apache RAVE is an open-source mash-up platform with similar functionality as the ROLE framework. Therefore, it was a very interesting project to contribute to. Technical ROLE partners have joined two Apache RAVE Hackathons in the Netherlands to present our work and discuss collaboration. The RAVE community received our concepts and implementation enthusiastically. Their main interest was in our IWC component, the space concept and our Linked Data-based APIs to retrieve and store data in the PLE. As the space specification proposal was already submitted to the OpenSocial community, ROLE decided to propose the two other components to the RAVE community. The process to achieve this is quite similar to the OpenSocial procedure. One has to announce the idea on the public mailing list of Apache Rave, where the idea and its specification can be openly discussed. The next phase is to provide an implementation of the component in RAVE and submit a patch. This patch will be reviewed by the community and after acceptance can be included in upcoming milestones. At the time of writing, both proposals have not yet been accepted. We hope to get approval of the RAVE community in the near future.

Strophe.js: The parts of the IWC component have been contributed to the open-source JavaScript XMPP library, named Strophe.js.[23] We mainly contributed our implementation of the XMPP protocol over WebSockets.[24] This makes the library more efficient as data can be efficiently pushed from server to client and long polling is no longer necessary.

Discussion and Conclusion

This chapter presented the architecture of the ROLE framework and the platforms where this framework has been integrated. The ROLE framework provides several components to enable responsive open PLEs, such as IWC, automated user activity tracking, collaborative spaces and authentication and authorisation services to protect data. These components provide the basis for real-time communication between widgets and users and automatic user activity tracking from tools and services. To evaluate the usability and usefulness of the ROLE philosophy and the ROLE framework, we have integrated the ROLE framework in various platforms, such as Moodle, CLIX (Govaerts et al. 2011; Rensing et al. 2013), Graasp and the ROLE SDK. These platforms have been used in various real-world evaluation settings (Govaerts et al. 2011), which have been documented in Chaps. 4, 5, 6 and 7.

In general, we can conclude that with the ROLE framework we were able to meet the project requirements and support the test beds. The birth of the Apache RAVE project with very similar goals indicates the interest and usefulness of the ROLE philosophy. Furthermore, the framework produced several components that were of interest to other open-source projects. Some of these open-source contributions have been completed, while others are still in progress. Additionally, results of the ROLE framework will be reused and extended in other research projects. For instance, the ROLE Widget Store will be reused in the Go-Lab project[25] as a repository of apps and online laboratories to enable teachers to assemble learning environments with online laboratories for inquiry-based learning. Additionally, Go-Lab will also use Graasp and the OpenSocial Spaces specification to enable inquiry-based learning spaces for STEM education at school. On the other hand, researchers of the Learning Layers project[26] are using and extending the ROLE SDK as their learning platform (Kovachev et al. 2013). As mentioned, the ROLE SDK is mainly meant for developers to extend their existing learning environments

[23] Strophe.js, http://strophe.im/strophejs/

[24] WebSocket, http://www.websocket.org/

[25] Go-Lab, http://www.go-lab-project.eu/

[26] Learning Layers, http://learning-layers.eu/

or extend the ROLE SDK itself to support their requirements. To support this, developers can easily contribute or fork the ROLE SDK GitHub repository.[27] We hope that in this way large parts of our efforts will be used beyond the end of the ROLE project.

Acknowledgments This research is funded by the European Commission's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 231396 (ROLE). Katrien Verbert is a Postdoctoral Fellow of the Research Foundation—Flanders (FWO).

References

Ashley H, Corbett J, Jones D, Garside B, Rambaldi G. Change at hand: Web 2.0 for development. Participatory Learning and Action. 2009;59:8–20(13).

Birman KP, Joseph TA. Exploiting virtual synchrony in distributed systems. In: 'SOSP'87: Proc. of the 11th ACM Symp. on Operating Systems Principles; 1987. p. 123–38.

Bogdanov E, Salzmann C, Gillet D. Contextual spaces with functional skins as OpenSocial extension. In: 4th International conference on advances in computer-human interactions; 2011. p. 158–63.

Bogdanov E, Limpens F, Li N, El Helou S, Salzmann C, Gillet D. A social media platform in higher education. In: IEEE Global Engineering Education Conference (EDUCON); 2012.

Bogdanov E, Ullrich C, Isaksson E, Palmr M, Gillet D. From LMS to PLE: a step forward through OpenSocial apps in Moodle. In: The 11th International Conference on Web-based Learning ICWL; 2012.

Butoianu V, Vidal P, Verbert K, Duval E, Broisin J. User context and personalized learning: a federation of contextualized attention metadata. J Univers Comput Sci. 2010;16(16): 2252–2271. https://lirias.kuleuven.be/handle/123456789/289802

Caceres M. Widget packaging and configuration, W3C working draft 22 March; 2011 (n.d.). http://www.w3.org/TR/widgets/

Chudnovskyy O, Nestler T, Gaedke M, Daniel F,Fernández-Villamor JI, Chepegin V, Fornas JA, Wilson S, Kögler C, Chang H. End-user-oriented telco mashups: the omelette approach. In: Proceedings of the 21st international conference companion on World Wide Web, WWW'12 Companion. New York: ACM; 2012. p. 235–38. http://doi.acm.org/10.1145/2187980.2188017

Dalsgaard C. Social software: E-learning beyond learning management systems. Eur J Open Distance E-learn (n.d.). 2006(2).

David P-C, Ledoux T. Wildcat: a generic framework for context-aware applications. In: Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing, MPAC'05, ACM, New York; 2005. p. 1–7. http://doi.acm.org/10.1145/1101480.1101483

DCMI Usage Board. DCMI metadata terms, DCMI recommendation. Dublin Core Metadata Initiative; 2006. http://dublincore.org/documents/2006/12/18/dcmi-terms/

Dragunov AN, Dietterich TG, Johnsrude K, McLaughlin MR, Li L, Her-locker JL. Tasktracer: a desktop environment to support multi-tasking knowledge workers. In: IUI; 2005. p. 75–82.

[27] The ROLE GitHub repository is available at https://github.com/organizations/ROLE

Ebner M, Taraghi B. Personal learning environment for higher education a first prototype. In: World conference on educational multimedia, hypermedia and telecommunications 2010. Chesapeake: AACE; 2010. p. 1158–66.

Eugster PT, Felber PA, Guerraroui R, Kermarrec A-M. The many faces of publish/subscribe. ACM Comput Surv. 2003;35(2):114–31.

Friedrich M, Wolpers M, Shen R, Ullrich C, Klamma R, Renzel D, et al. Early results of experiments with responsive open learning environments. J Univers Comput Sci. 2011; 17(3):451–71.

Fruhmann K, Nussbaumer A, Albert D. A psycho-pedagogical framework for self-regulated learning in a responsive open learning environment In: Proc. of the International Conference eLearning Baltics Science (eLBa Science 2010), Rostock, Germany; 2010.

Gillet D, Bogdanov E. Personal learning environments and embedded contextual spaces as aggregator of cloud resources. In: Proceedings of the 1st International workshop on cloud education environments (WCLOUD 2012); 2012.

Glahn C. Using the adl experience api for mobile learning, sensing, informing, encouraging, orchestrating. In: 2013 Seventh international conference on Next generation mobile apps, services and technologies (NGMAST); 2013. p. 268–73.

Govaerts S, Verbert K, Dahrendorf D, Ullrich C, Schmidt M, Werkle M, Chatterjee A, Nussbaumer A, Renzel D, Scheffel M, Friedrich M, Santos Odriozola JL, Duval E, Law EL-C. Towards responsive open learning environments: the ROLE Interoperability framework. In: Delgado Kloos C, Gillet D, Crespo Garcia RM, Wild F, Wolpers M (eds) Towards ubiquitous learning—Proceedings of 6th European conference of technology enhanced learning, EC-TEL 2011. Berlin: Springer; 2011. p. 125–38. https://lirias.kuleuven.be/handle/123456789/319048

Guo Y, Rui J, Zhou H. Pervasive and personal learning environment using service-oriented architecture: a framework design. In: International conference on networking and distributed computing; 2010. p. 153–5.

Health T, Bizer C. Linked data: evolving the web into a global data space (book) (n.d.). http://linkeddatabook.com/editions/1.0/

Henricksen K, Indulska J. Developing context-aware pervasive computing applications: models and approach. Pervas Mob Comput 2006;2(1), 37–64. http://dx.doi.org/10.1016/j.pmcj.2005.07.003

Hickson I. HTML5 web messaging. Working draft, W3C; 2011.

Hickson I. The web sockets API, Technical report, W3C. W3C Working Draft; 2009.

Isaksson E, Palmer M. OpenApplication specification (n.d.). https://docs.google.com/document/d/1VReR5Aod7HyIAhBXRB_u9mY3vKdUHR9uXk7_w6KBGX0/preview

Isaksson E, Palmer M. Usability and inter-widget communication in PLEs. In: Proceedings of the 3rd Workshop on Mashup Personal Learning Environments; 2010.

Kovachev D, Renzel D, Nicolaescu P, Klamma R. Direwolf—distributing and migrating user interfaces for widget-based web applications. In: Daniel F, Dolog P, Li Q (eds) Web engineering, vol. 7977, Lecture notes in computer science. Berlin: Springer; 2013. p. 99–113. http://dx.doi.org/10.1007/978-3-642-39200-9_10

Ladas C. Scrumban—essays on Kanban Systems for Lean Software development. Seattle: Modus Cooperandi Press; 2009.

Maness JM. Library 2.0 theory: Web 2.0 and its implications for libraries. Webology. 2006;3(2).

Millard P, Saint-Andre P, Meijer R. XEP-0060: publish-subscribe. Technical report. XMPP Standards Foundation; 2010. Draft Standard. http://xmpp.org/extensions/xep-0060.html

Mitchell-Wong J, Kowalczyk R, Roshelova A, Joy B, Tsai H. Opensocial: from social networks to social ecosystem. In: Digital EcoSystems and Technologies conference, 2007. DEST'07. Inaugural IEEE-IES. p. 361–6.

Moffit J, Cestari E. An XMPP sub-protocol for WebSocket. Technical report, Internet engineering taskforce; 2010. Draft Standard. http://tools.ietf.org/html/draft-moffitt-xmpp-over-websocket-00

Nussbaumer A, Berthold M, Dahrendorf D, Schmitz H-C, Kravcik M, Albert D. A mashup recommender for creating personal learning environments. In: ICWL; 2012. p. 79–88.

Oliver N, Smith G, Thakkar C, Surendran AC. Swish: semantic analysis of window titles and switching history. In: Proceedings of the 11th international conference on intelligent user interfaces, IUI'06, ACM, New York; 2006. p. 194–201. http://doi.acm.org/10.1145/1111449.1111492

Paterson I, Smith D, Saint-Andre P, Moffitt J. XEP-0124: bidirectional-streams over synchronous HTTP (BOSH), Technical report, XMPP Standards Foundation; 2010. http://xmpp.org/extensions/xep-0124.html

Pierce ME, Singh R, Guo Z, Marru S, Rattadilok P, Goyal A. Open community development for science gateways with apache rave. In: Proceedings of the 2011 ACM workshop on gateway computing environments, GCE'11, ACM, New York; 2011. p. 29–36. http://doi.acm.org/10.1145/2110486.2110491

Rath AS, Devaurs D, Lindstaedt SN. Uico: an ontology-based user interaction context model for automatic task detection on the computer desktop. In: Proceedings of the 1st Workshop on context, information and ontologies', CIAO'09, ACM, New York; 2009. p. 8:1–8:10. http://doi.acm.org/10.1145/1552262.1552270

Rensing C, Schwantzer S, Faltin N. Integration selbstgesteuerten ressourcen-basierten lernens in eine durch instruktion geprägte lernumgebung. In: Andreas Breiter CR, Meier M (ed) Proceedings der Pre-Conference Workshops der 11. e-Learning Fachtagung Informatik-DeLFI 2013. Berlin: Logos. p. 69–70. ftp://ftp.kom.tu-darmstadt.de/papers/RSF13-1.pdf

Renzel D, Behrendt M, Klamma R, Jarke M. Requirements bazaar: Social requirements engineering for community-driven innovation. In: Proceedings of the 21st International requirements engineering conference; 2013. p. 326–7.

Saint-Andre P. RFC 3920—Extensible Messaging and Presence Protocol (XMPP): Core, Technical report; 2004a. Jabber Software Foundation. http://www.ietf.org/rfc/rfc3920.txt

Saint-Andre P. RFC 3921—Extensible Messaging and Presence Protocol (XMPP): instant messaging and presence, Technical report; 2004b. Jabber Software Foundation. http://www.ietf.org/rfc/rfc3921.txt

Saint-Andre P. XEP-0045: Multi-User Chat, Technical report. XMPP Standards Foundation; 2008. Draft Standard. http://xmpp.org/extensions/xep-0045.html

Schmitz H-C, Kirschenmann U, Niemann K, Wolpers M. Contextualized attention metadata. In: Roda C, editor. Human attention in digital environments. Cambridge: Cambridge University Press; 2011. p. 186–209.

Schwaber K. Agile project management with scrum, best practices; 2004. Redmond: Microsoft Press. http://books.google.be/books?id=dJlqJfm8FM4C

Sire S, Paquier M, Vagner A, Bogaerts J. A messaging API for inter-widgets communication. In: Proceedings of the 18th International Conference on World Wide Web, WWW'09. New York: ACM; 2009. p. 1115–6.

Ullrich C, Borau K, Luo H, Tan X, Shen L, Shen R. Why Web 2.0 is good for learning and for research: principles and prototypes. In: Proceedings of the 17th International world wide web conference. New York: ACM; 2008. p. 705–14.

Ullrich C, Shen R, Borau K. Learning from learning objects and their repositories to create sustainable educational app environments. In: IEEE International Conference on Advanced Learning Technologies. Los Alamitos: IEEE Computer Society; 2013.

Vozniuk A, Govaerts S, Gillet D. Towards portable learning analytics dashboards. In: International Conference on Advanced Learning Technologies; 2013. Los Alamitos: IEEE

Wilson S, Sharples P, Popat K, Griffiths D. Moodle wave: reinventing the VLE using widget technologies. In: Proceedings of 2nd workshop mash-up personal learning environments (MUPPLE'09). Workshop in conj. with 4th European Conference on Technology Enhanced Learning (EC-TEL 2009): Synergy of disciplines; 2009. Berlin: Springer, p. 47–58.

Wolpers M, Martin G, Najjar J, Duval E. Attention metadata in knowledge and learning management. In: Proceedings of I-Know 2006: 6th International Conference on Knowledge Management; 2006. p. 1–8.
Wolpers M, Najjar J, Verbert K, Duval E. Tracking actual usage: the attention metadata approach. Educ Technol Soc. 2007;3(10):106–21.
Yuan J. Liferay Portal 5. 2 Systems development, from technologies to solutions; 2009. Birmingham: Packt Publishing. http://books.google.be/books?id=OaR3JEZJz5EC