Chapter 8
The iTEC Widget Store

David Griffiths and Kris Popat

Abstract The iTEC project undertook the task of distributing resources and services for learning activities across a wide range of technological platforms in many different countries. Interoperability was achieved through the W3C widget specification and the Apache Wookie widget server. A connector framework was developed to enable widgets to be embedded in host platforms. In order to facilitate the discovery and deployment of widgets the iTEC Widget Store was developed and evaluated. This is an open source app store whose functionality is separated from the widgets which it serves. It was found that the adoption of W3C widgets beyond the project was very weak, and consequently there were few widgets available for inclusion in the Widget Store. Consequently a range of authoring functionality was made available in the Widget Store, enabling users to create their own widgets from online resources or local files. The Widget Store was also extended to enable it to handle LTI tools, including the management of authorisation keys.

Keywords Flexible services • Education • Interoperability • App store • Open source • Widget • Apache Wookie • Open social • LTI

The Role of Widgets in iTEC

The iTEC project was established to pilot innovative Technology Enhanced Learning (TEL) activities on a large scale across Europe. This presented the challenge of delivering technological support for TEL scenarios to schools using a range of different technologies in many different countries. At the proposal stage the decision was taken that in order to achieve this, the project would make use of the W3C widget specification (W3C 2011), as described in Chap. 4. A W3C based infrastructure was to be provided to enable a collection of resources and services to be collected and curated on central servers, and to deliver them to a wide range of

D. Griffiths (✉)
Institute of Educational Cybernatics, University of Bolton, Bolton, UK
e-mail: D.E.Griffiths@bolton.ac.uk

K. Popat
University of Bolton, Bolton, UK
e-mail: projects@krispopat.co.uk

F. Van Assche et al. (eds.), *Re-engineering the Uptake of ICT in Schools*,
DOI 10.1007/978-3-319-19366-3_8

platforms. It was also set out that this would be achieved using open source software and standards based systems, so that others could adopt and build on the systems developed by the project. The Wookie Widget server was identified as the technical means to achieve this functionality, so as to

> …provide a technological infrastructure which supports the mash-up and interoperation between different tools and services in order to ensure a seamless experience for teachers, learners and other stakeholders while providing the user with access to a variety of tools and services (European Commission 2010).

Wookie was originally developed by the Institute for Educational Cybernetics, located at iTEC partner Bolton. By the time iTEC commenced Wookie had been accepted into the Apache Incubator, which seeks to generate community support for software projects before they are definitively accepted by the Apache Foundation. Wookie graduated as a top level Apache project during the lifetime of iTEC. This use of an emerging open source infrastructure enabled the project to support innovative functionality by working with an evolving code base in which project staff had great expertise, while also ensuring that project outcomes were as widely available as possible. The planned work focused on the enhancement and extension of Apache Wookie, the creation of connectors which would enable Wookie widgets to be embedded in host environments, and development of tools for the authoring of widgets.

The iTEC work with widgets was therefore a means towards the projects wider research goals, rather than an end in itself. Nevertheless, although the underlying technology of W3C widgets was in place, it was not mature. Consequently there were a number of technical research questions to be addressed concerning the most effective architecture and methods to be used in managing and delivering widgets.

- What extensions are required to the Apache Wookie W3C widget API in order to support the planned iTEC functionality?
- What affordances opportunities and difficulties are raised by implementing a full separation between user interface and business logic?
- What is the appropriate outline data model for store services?
- What are the critical usability factors in designing an open online store?
- What user interface can support users in making sense of the process of managing widgets? The process of mixing functionality from a number of sources on a single Web page is conceptually complex for users who have only a vague idea of what a server is, or how a Web page is composed.

This led to the iterative design of the users' interaction with the system, not only in terms of the interface elements, but also in the underlying functionality. Indeed, as the project progressed evaluation showed that the technical solutions which were developed for delivery of tools and services worked well, but the system was not widely adopted by teachers. This led to the development team to review the assumptions which lay behind the technical plan, and to propose the development of an App Store which would make the affordances of the infrastructure clearer and more available to teachers. The architecture, features, and design of the Widget Store, as detailed below, embody our response to the research questions which we have identified. The store front of the Widget Store is shown in Fig. 8.1.

Fig. 8.1 The iTEC Widget Store

A Long-Standing Problem of Interoperability

The need for the widget infrastructure developed by the iTEC project was not only determined by the practical requirements of the project, it was also informed by an established line of work which critiqued the prevailing technical infrastructure for learning. Building on Koper's (Koper and Tattersall 2005) critique of the lack of a connection between pedagogical thinking and the structure of online applications and courses, the Learning Design movement within educational technology sought to create abstract representations of designs for learning activities which could be instantiated for particular contexts. This gave rise to the development of a wide range of tools which were intended to enable teachers to author reusable lesson plans. These include LAMS (Dalziel 2003), the Graphical Learning Modeler (Neumann and Oberhuemer 2009), the Pedagogic Planner (Laurillard et al. 2011). Within this line of work the Reload and Recourse editors (Griffiths et al. 2009) were created by a team drawn from the IEC, and the Centre for Educational Technology Interoperability and Specifications (Cetis) service run by the IEC. The Wookie Widget Server was originally designed within the TENCompetence project (TENCompetence Foundation 2010; Sharples et al. 2008) to provide flexible services for IMS Learning Design (LD) that could be selected and contextualized with the Recourse editor. These abstract descriptions of lesson plans could be provisioned, and then delivered to specific learners and teachers in particular institutions. The work reported in this chapter was in some respects an extension of this effort to provide teachers with effective tools for planning learning activities, as described in Griffiths et al. (2009).

The Widget Server was also strongly related to the concept of the Personal Learning Environment, which emerged from contributions by members of the IEC and Cetis. The concept has its origins in a paper by Olivier and Liber (2001), in which they point out that

> We all acknowledge the importance of being learner-centred and of supporting the lifelong learner. However the Web-server-and-stateless-Web-browser paradigm inherently supports an institution-centred approach and fails to meet some important needs of the learner.

A line of work was established which explored the constraints imposed on the learner and the teacher by the dominant paradigm of the Virtual Learning Environment (VLE), in which the institutional infrastructure is responsible for storing and delivering all learning services and content. The effort to find technical alternatives to the VLE resulted in research led by Wilson, which identified widgets as a promising approach. The ambition and the rationale for the technical approach behind this work was summarized in Wilson et al. (2011) as

> … an approach to challenging the dominant design through creatively subverting the VLE using highly interactive applications (widgets) that can be delivered within the VLE but also embedded by the users into other platforms, including individually-owned tools and websites. By extending the capabilities of the VLE in this manner, we can create a new conversation about the VLE that moves us away from the dominant design, but stays within the comfort zone of lecturers, managers and students who have become used to the existing model. Also, rather than attempt to 'create' a personal learning environment (PLE) that is provided to learners, we instead open up the VLE to be remixed by users to construct their own PLE using technologies of their choosing.

The relationships between these two aspects of interoperability, and the way in which they contributed to the Wookie Widget Server and the iTEC Widget Store, are described in greater detail in Griffiths et al. (2012a, b). 'The Wookie Server, a case study of piecemeal integration of tools and services'. For both aspects the central contribution of the Wookie server was to enable services and resources to be managed and delivered separately from the VLE which teachers and learners were required to use. There was a good fit between iTEC and these technologies for two reasons. Firstly, the pragmatic requirements generated by the need to deliver centrally managed services to pilots in a wide range of target platforms in different contexts were similar to those generated by the Learning Design and PLE approaches. Secondly, the focus of iTEC on innovation made it attractive to make use of a platform which enabled teachers to have access to services and resources from beyond their institutional platform, and in this it echoed the discourse around the PLE.

In practical terms, initiatives such as iTEC, which seek to develop and share innovative teaching activities and practices, are constrained by the technical affordances of existing platforms. For example, there are limitations on the use of the same tools across different learning environments, and in the integration of activities between different tools in different environments. Often the only viable approach is to leave the confines of the institutional system, and to adopt the services of a third party Web applications provider, an option which brings with it a different set of constraints relating to lack of control over functionality and data. The IMS Learning

Tools Interoperability specification (IMS Global Learning Inc. 2010–2012), which we discuss in the section on "Moving Beyond Widgets: IMS LTI Compatibility" below, has made some progress in addressing this issue, but there remains a great deal to be done. This problem is a long standing one, and essentially it remains as described by Liber and Britain in their report on Virtual Learning Environments in Universities (1999), who analyse how tools are locked-in, not only to the particular VLE platform, but also with little provision for tools to be deployed across modules or lessons in ways which would facilitate innovative pedagogical organization.

The Technical Response of the iTEC Project

The iTEC project responded to the challenges identified in the previous section by establishing a Connector Framework for use in iTEC pilots. The connector framework addresses the specific issues of interoperability between tools and platforms and the removal of technical barriers by enabling widgets to be embedded in host platforms (known as shells in iTEC). A 'connector framework' is a broad term for a set of Application Programming Interfaces (APIs) and Software Development Kits (SDKs) which allow for the instantiation of and communication between a common toolkit across a range of different platforms. Such toolkits, including Google OpenSocial Apps and 'gadgets', which were transferred to W3C in 2015, is one such toolkit (W3C 2015). The use of connector frameworks featured strongly in efforts to realize learning environments which marry centrally-provided tools with Personal Learning Environments, for example the EU-funded ROLE project (see Kroop et al. (2015)), which made use of OpenSocial. In these projects, efforts have been made to facilitate the inter-operation of widgets across the diversity of platforms where they might be used, removing barriers of authentication, data sharing and platform dependence. The connector framework in iTEC moved forward this established work by providing a service designed for managing educational tools for schools, implementing it in Apache Wookie, and piloting it on a large scale. The requirements of the connector framework for iTEC were that it should be:

- Adaptable, so that it is capable of functioning with a range of infrastructures in different schools and countries, and supporting the pedagogic adaptation of scenarios for differing school contexts.
- An enhancement of the ability of teachers and educational leaders to manage the teaching for which they are responsible.
- Capable of being centrally managed, so that the coherence of the pedagogic designs and technical offering is maintained.

As a server-side support and delivery mechanism for W3C widgets, with additional support for Open Social gadgets and widgets with specific Wookie features, the architecture of Apache Wookie was conceived within this paradigm. Wookie functions to both store and deliver W3C widgets to a range of platforms through the provision of the connector framework API. The essence of this approach to a

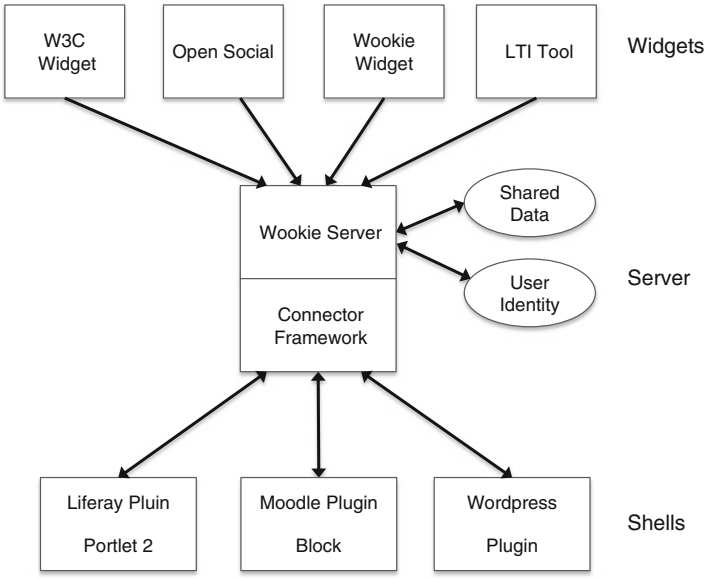

Fig. 8.2 Architecture of the Wookie Connector Framework

'connector framework' is shown in Fig. 8.2, where API calls are provided both to instantiate tools and to manage users.

The iTEC connector framework enabled developers to create plugins in new environments to allow for the linkage (including user authentication) and embedding of tools. RESTian APIs provide function calls to allow the plugin to get lists of widgets/tools, set user information, instantiate widgets or get a URL to retrieve a widget. SDKs were developed as part of this toolkit to provide easy access to these APIs in a variety of programming languages. Wookie manages the unpacking and delivery of widgets to web applications and download to devices that already support widget packages, and acts as a mechanism for managing widget users and facilitating data storage and widget interoperability. These mechanisms enable a rich set of additional tools and content (indeed anything that can be housed in a browser) to be integrated with existing shells such as virtual learning environments, social software, mobile devices and whiteboards. The technical challenge lay in taking a technology designed for delivering small, self-contained applications and allowing them to be collected, connected (mashed-up) and delivered to the specific shell requirements of iTEC.

Unlike other widget platforms (for example Google gadgets), Wookie is platform neutral, requiring for authentication purposes only a 'screen name' of a user, which is passed to it from a shell. The plugins are configured with a host URL for Wookie itself and an API key. This key is created within Wookie to identify the calling environment. It is used by Wookie for data sharing which is particularly useful for widgets that need data to be persisted and communicated between users, or for persisting

data for a single user. For instance, a chat widget or a vote widget needs to send chat or vote data to the server. Using Wookie's "sharedDataForKey" function this data is accessible to other users of the same widget, given that the widget id and the API key are the same. As a result, collaborative multi-user activities can be established in Wookie with no need to create users for that particular activity. In effect, this means that the user management for a Wookie widget-based activity need only be done by the shell that instantiates the widget, thus removing one of the principle barriers to the integration of external tools.

In the standard Wookie setup there are three modes of use for a widget. These are established by a set of terms defining the role of the user. The roles, and thus modes, are: student, teacher and administrator, and in most cases teacher and administrator are the same. For some widgets this dual role allows the widget to be configured rather than used, for example, RSS feeds may need to be set up, or chat rooms to be created, etc. The way in which the roles are used is determined by the way in which a particular widget is programmed, and Wookie provides the framework for this to happen. These roles should be passed on from the host environment where possible. In the case of the Moodle plugin the roles defined in Moodle are rationalized (there are seven standard roles in Moodle hence the need for rationalization) and passed via the connector framework to Wookie which then passes them on to the widget.

Each individual plugin makes use of the Wookie connector framework, but it is a separate entity and is more akin to the environment in which it is embedded than it is to Wookie. For instance, in the case of Liferay the plugin was written in Java as a Portlet using the JSR 286 specification,[1] and it should work with any environment that supports that specification. For iTEC it was targeted and tested on Liferay.[2] Similarly a Moodle plugin was written as a Moodle block in PHP.[3] The source code for the connector framework itself is part of Apache Wookie.[4] Despite this range of technical underpinnings, the user experience in each plugin is similar for each.

The Need for a Widget Store

The connector framework was developed in the first phase of the project, together with its associated plugins for clients.[5] These provided the infrastructure that was necessary for administrators and teachers to be able to use centrally managed widgets in activities across the range of schools involved in iTEC pilots. The discovery and selection of widgets for use, however, proved problematic. The available widgets

[1] https://jcp.org/aboutJava/communityprocess/final/jsr286/

[2] The Liferay plugin is available at: (#http://iecbolton.jira.com/svn/ITEC/liferay_plugin/trunk/#)

[3] The Moodle plugin is available at: https://github.com/krispopat/Wookie-Moodle-Connector

[4] Apache Wookie is available here https://svn.apache.org/repos/asf/wookie/trunk

[5] The REST API for the store is documented at http://www.widget-store.org/index.html?subpage=documentation. Access to the REST API for the demonstrator version is at http://www.widget-store.org/edukapp/api/rest

were shown to users on a Web page generated by the Wookie server, and they had to scroll down to find the widget that they wanted to use. This arrangement had the virtue of simplicity, but once large numbers of widgets were made stored on Wookie it quickly became unmanageable. It was found that the connector framework software was creating its own barriers to the effective deployment in iTEC which it was seeking to promote. In seeking a way out of this impasse, the project decided to develop and deploy an app store.

Linux based operating systems have long used package managers and app stores as a means of hiding the complexity involved in finding the appropriate packages and installing software. App stores provide users with a single place to go where new functionality, tools, and activities can be added to their computers with a guarantee that they will work without further configuration. In recent years the app store approach has been adopted by mobile phone providers, but most of these app stores are currently proprietary systems tied into particular operating system architectures. With the interoperability opportunities presented by Wookie widgets, an educational app store presented itself as a way of extending the metaphor of 'apps' into the education space and providing teachers with a solution to the over-burdensome processes of discovering and installing new tools. The fact that teachers had high levels of familiarity with app stores on mobile platforms was a strong argument in favour of adopting this approach in iTEC.

While the purpose of our development work was to create an app store to meet the needs of education, the decision was taken to use industry standard technologies wherever possible, rather than to develop our own education-specific systems. By building on open specifications and open source software we were able not only to achieve more effective development, but also to make it easier to extend and adapt the functionality of the app store. The store was called the 'iTEC Widget Store' to reflect its role within the project, but it constitutes a set of open source software which can be used to build an app store for any purpose. The flexibility of the software was demonstrated by the provision of support for IMS Learning Tools Interoperability (LTI) in the final release of the iTEC Widget Store.

Building the Store

The Widget Store is built from several pre-existing software systems as well as some newly created ones. The pre-existing software systems are:

- Apache Wookie,[6] which houses, parses, manages and delivers W3C widgets.
- Solr,[7] which is used for search indexing and query language. The engine behind the discovery service.
- Shindig,[8] which is used to house, parse, and manage OpenSocial gadgets.

[6] http://wookie.apache.org/

[7] http://lucene.apache.org/solr/

[8] http://shindig.apache.org/

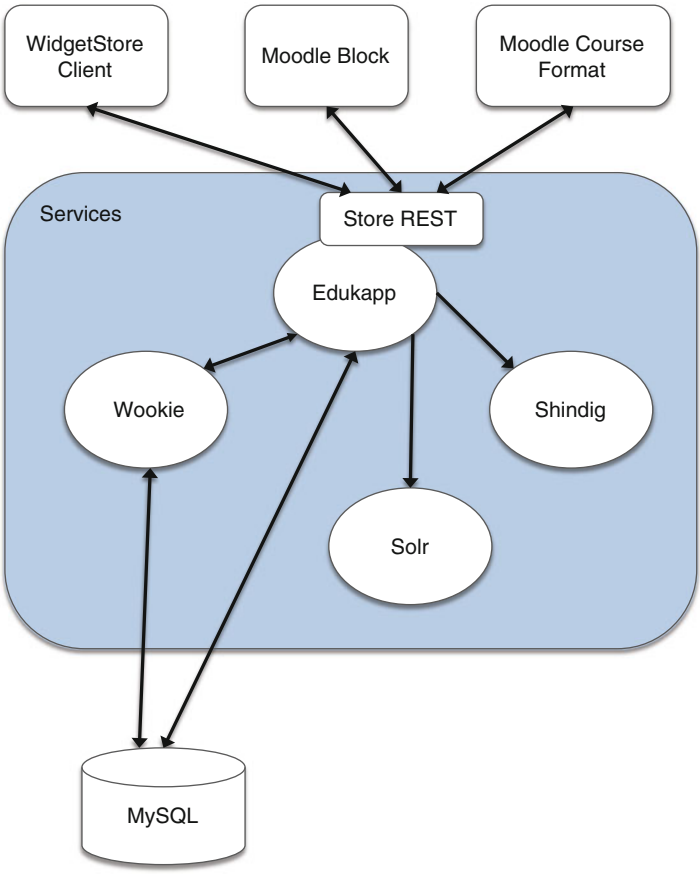

Fig. 8.3 The Widget Store architecture

The store service itself is based upon Edukapp,[9] a prototype widget app store developed with funding from both Jisc in the UK and the European Commission. This software was substantially modified and extended to include a dedicated pure REST API and also to include some model requirements particularly to describe functionalities.

A user interface for the store is implemented as a separate software package. In the case of the store implemented for iTEC, this is a pure HTML/JavaScript client, written and packaged as a W3C Widget. Figure 8.3 gives an overview of the various services that make up the store. The iTEC Widget Store as seen by the user is a client which accesses a service to manage the data for tags, functionalities, reviews and ratings. This service is based upon an open-source web application called Edukapp, initially funded by the Joint Information Systems Committee (Jisc) in the

[9] http://widgets.open.ac.uk:8080/

UK, but which was further developed as a collaboration between iTEC (through the University of Bolton) and the European Commission funded ROLE project (through the Knowledge Media Institute of the Open University UK). This offers all projects the advantages of pooling resources towards a common goal, and of enhancing the prospects for sustainability of project outcomes.

The server exposes a set of calls that can be made remotely by a software client in order to perform the following actions.

- Search for widgets—using the discovery service set up earlier in the project.
- Get individual widget information (extended profile including all reviews, tags, functionalities and ratings averages)
- Get user information
- User sign-in
- User registration
- Widget upload
- Widget Creation:
    - Flash file, Java file
    - Web folder
    - URL
    - Embed Code
    - LTI Tool
- Tagging widgets
- Adding reviews to a widget
- Assigning functionalities to a widget
- Adding or updating a user rating for a widget
- Categorizing a widget
- User/Widget association for favourites

During work on the Store the capabilities of Edukapp were greatly extended, and a number of iTEC-specific extensions were added with the aim of meeting the requirements of the project. In order for the iTEC Widget Store to be fully independent of Wookie, the Edukapp kernel was separated so that it communicated with Wookie solely through the REST API. It was also necessary to develop a means of representing and setting functionalities of widgets, as well as introducing date management capabilities. Some extensions to the data model were also required to address iTEC specific meta-data requirements, in particular the ontologies developed to describe functionalities.

The diagram shows that the Store REST API built upon Edukapp is central to communication between the store and the clients. In this case two clients are shown, One is the Widget Store Client which, as mentioned above, has been packaged as a W3C Widget. The other is a Moodle block, which allows widgets to be included in a Moodle course. The lines indicate the flow of control and information between the services. Edukapp is central to the service architecture as it makes use of the functionalities in the other services to support two different formats of widgets and searching.

The Discovery Service

In order for the Widget Store is to be usable in practice, it was essential to enable teachers to discover new tools based on search criteria. For this a 'widget discovery service' was implemented. This is a backend search engine for widgets that are stored in Wookie, and it allows widgets to be found through searches on the meta-data stored in Wookie. The search engine runs as a separate service that sits alongside Wookie, and this separation allows the discovery service to be flexible and extensible. For instance, there may be a number of running instances of Wookie with interesting widgets installed in different locations. The discovery service could be configured to search all or a number of these instances.

Figure 8.4, below, shows how information flows between the discovery service interface and the Store. The user sees the search interface as a text box which is embedded in the Store. When a search term is entered the user is presented with a list of results in the store interface from which a widget could be chosen. Choosing the widget sends a request back to the store with the Widget ID. The store responds by getting an instance or creating an instance of the widget from Wookie of from its own internal data store and sends the instance information back to the plug-in or store interface so it can be displayed.

The discovery service makes use of Apache Solr/Lucene, with Lucene being a search language, while Solr is a search engine. Solr is a separate web application which, in this case, is configured to run with Wookie and the store as data sources for its indexes.. The search engine is written as a cluster of search cores, which

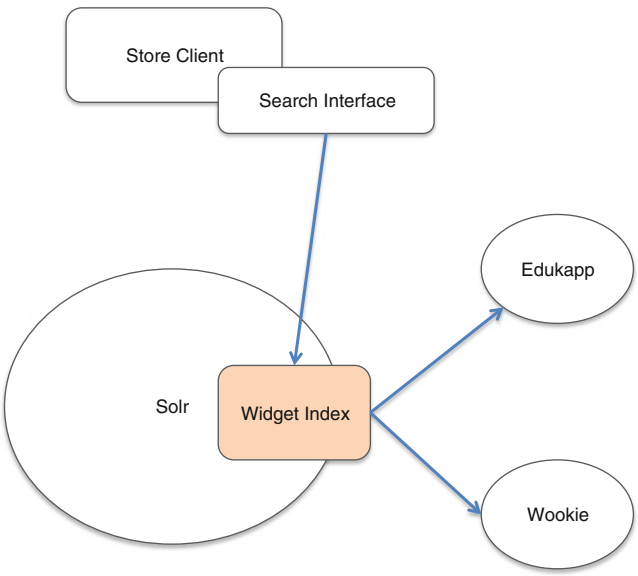

Fig. 8.4 Wookie discovery service architecture

communicate with the indexing services via REST. The list of widgets is returned in ATOM format and categories in JSON.

The discovery service sits behind the searching in indexing capabilities of the Store, and it has been extended to index data from the store as well as from Wookie. Originally the discovery service simply indexed the data contained in the config file of each widget. With the development of the store it was extended to include data from tags, categories and functionalities.

The iTEC Store extended Apache Wookie in a number of ways in order to support the functionalities required by the project. A store API was established as a separate service located along side Wookie itself. This extends Wookie's capabilities with meta-data for each widget beyond the meta-data associated directly with the widget in its config file. As a result the store is able to provide the following extensions:

- Ratings for Widgets: This enables each user to rate a widget. Each user has one rating record per widget which can be updated, and the ratings of all users can be aggregated (averaged).
- Reviews for Widgets: Reviews are composed of a block of text which is associated with a user record and a widget record. The time of creation is recorded.
- Tagging: Tags can be created by users, and those tags which have already been created can be re-used by other users.
- Functionalities: These enable users to provide a weighting value for widgets which conforms to the taxonomy for functionalities developed by iTEC (Anido et al. 2012).
- Categories: These allow widgets to be categorized according to administrator-defined words. The categories are used by the discovery service search but also a faceted filtering system, which uses a group, based exclusive-or logic to narrow down the number of widgets displayed.
- Favourites: Allowing users to build a list of favourites widgets and also to view other user's favourites.

The Store Deployed as a Widget

The first version of the store was an HTML/JavaScript site which called the REST API, which in effect meant that the Widget Store was a place on the web. However the widgets would not actually be used in the store or installed directly from the store. Rather most of the widgets were used in a shell, and within iTEC this was usually Moodle or DotLRN. The result was that widgets were to be created, reviewed and rated in one place but deployed in a different one, whereas it would be more elegant and clearer to users to access the store in the same web location as the widget container or shell. Because of this the store was re-developed the store as a W3C widget—albeit quite a large one. All the functionality and more that was in the original site was transferred to the widget. This could then be embedded in the shell

in the same way as any other widget. This also had the benefit that an additional login was not required, as widgets are provided with information about the users logged into the shell. The store interface was written as html with JavaScript, and all functionality is accessed via the REST API using Ajax. Security on the REST API is handled via http authc basic. This is the recommended way of securing a REST API, as typically such interfaces do not make use of session management and the authentication is passed with each function call. This allows clients to the API to be written in any language which supports network calls, and so they are not tied to web browser technologies.

The REST API

The REST API exposes the core functionality of the store to clients and it contains a number of different modules, which supply discrete functionality. It has been designed to encapsulate the types of functionalities associated with store including reviews and ratings. Our definition of the store goes beyond this by encompassing the publishing side of the store and extended categorizations through functionalities. The modules provide these capabilities.

Creator: This handles uploading of widget packages to the store and has calls allowing widgets to be created either from Flash, Java applets, embed codes or web folder packages. Web folder packages are ZIP files with self-contained web sites in. The web site can have any kind of functionality. This zipped folder is converted to a W3C widget by the system.

Discovery: This API exposes the store's search and filtering mechanism. The calling system can also get extended profile information for particular widgets.

Tags: These functions allow widgets to be tagged and those tags to be managed. Tags can also be used to get a list of widget profiles associated with a tag.

Reviews and Ratings: These modules handle reviews and ratings for widgets, both of which are many to one. In the iTEC store each user can only have one rating per widget, and they can be changed. They are also averaged.

Functionalities: This is an iTEC specific requirement. It is a type of weighted tag associated with the widget profile that is based purely on an agreed taxonomy. In this way these functionalities are directly usable by the recommender and composer.

Users: User management is included to allow the store to be used independently of the iTEC environment. This aspect was handled by UMAC for iTEC project activities.

Statistics: Calls made to the store are recorded in the database automatically. Other calls can be made to update the statistics from external services. This is particularly useful to allow the client to track external actions outside of the REST services such as users downloading the widget, embedding it or merely viewing it. These statistics are included within the widget profile structure. The full REST API and data types are attached to the end of this document as appendices.

Widget Store Content Tools

The content available for use with the Widget Store has been constrained by changes in the wider ICT industry, and in the eLearning market. The choice of W3C widgets as an enabling technology for the project was based on the expectation that the positive trends in adoption of the specification at the time writing the proposal would be continued during the life of the project. In this we were sorely disappointed. The W3C widget specification has not achieved its goal of unifying the Web app market, and number of useful publicly available widgets is very small. Consequently the development team placed a great deal of emphasis on the provision of tools for widget creation. These were added as the project progressed, often in response to requests from users.

The first widget creation tools which were provided enabled the user to upload either a W3C Widget file or an Open-Social gadget. The user was expected to know how to create these packages already before installation. This upload feature was expanded by providing a form allowing users to create a widget using an existing Flash or Java applet. These can be uploaded and form fields ask the user for the extra metadata required for making a widget package. Sending this form triggers a widget package creation section in the store which, using templates embedded the applet in an html page, created the widget configuration file and packaged the whole thing together as a widget package, which was then posted to Wookie, indexed and made searchable. It became clear however that more was needed, and the team proposed that it would be useful to handle embed codes as a way of sharing existing widget tools, movies, content etc. Initially a special widget was developed that allowed the user to input an embed code, this then generated a widget package and installed it on the server. This functionality proved popular with users, and so it was then moved into the store itself as a widget creation mechanism. A further extension of functionality was The Web Address tool, which creates a widget from a web URL. This effectively creates a mash-up portal to another web site, and is shown in Fig. 8.5, below. Finally, the Mini Web Site creation tool was provided in response to some teachers who commented that while they taught their students to build simple web sites, it was often very difficult for them to actually publish them. The new tool enables teachers or learners to upload a zipped set of web pages, which are converted into a widget package and made available on the server.

Managing Widgets

The widget creator tools provided in the store proved effective in enabling large numbers of widgets to be created, and in the process it changed the focus of the iTEC Widget Store. The Store had originally been conceived as a means of managing resources and services provided by third parties, but its use in iTEC increasingly became as a tool which could be used by teachers and coordinators to identify and

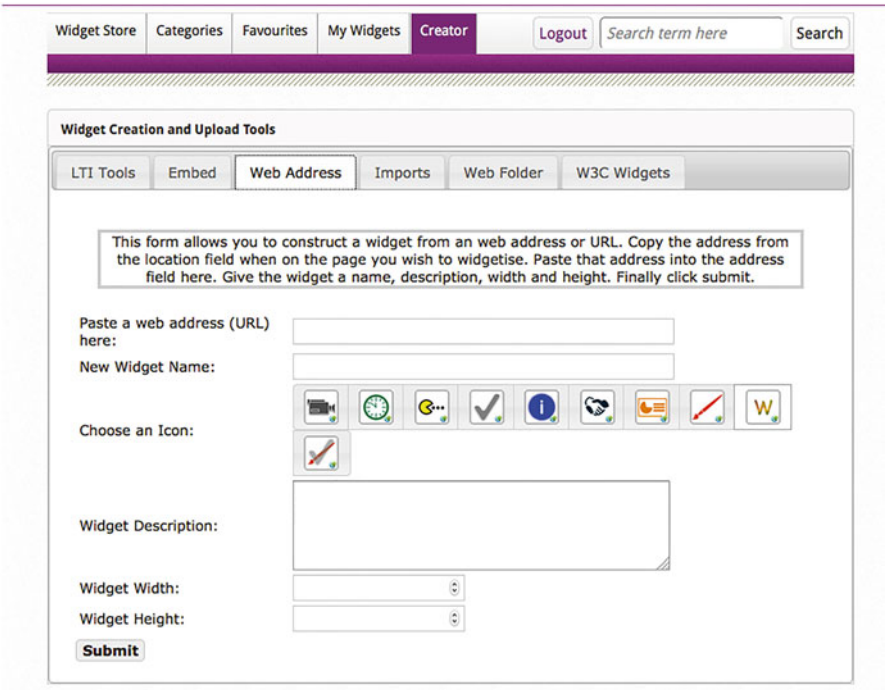

Fig. 8.5 The Web address tool

encapsulate valuable web functionality of any kind as a widget. The search and description features of the Store enabled the resulting widgets to be shared and curated.

The ease with which new widgets could be created quickly created a problem for users, who were unable to distinguish or find their new widgets among the many widgets in the store. To help users find what they needed, different types of user created widgets in the store were distinguished by a set of icons included in the widget creation tools. The icons correspond to the following categories: Collaboration Tools; Creativity Tools; Games and Fun Widgets; Research and Information Tools; Films, Videos (e.g. YouTube Embeds); Reflection and Self Organization Tools; Presentations; Quizzes and Questions; Quiz Creation Tools. Three sets of the same icons were created, with different mini icons in the corner indicating whether the widgets are flash files, web embeds or web folders. When the mouse rolls over the widget additional information is displayed, including whether the resource is an LTI tool. When uploading standard W3C widgets the creator does not get the option to add icons as they contain their own configurations and icons.

In early versions of the Store a simple list of widgets was presented to users. Selecting one of them allowed the user to delete that widget, but little more. The My Widgets area now shows the user's widgets in a table list with tools to publish, categorize, edit and delete their widgets.

There were also requests from teachers that there should be a moderation process in place to ensure that learners did not upload inappropriate content, and the same consideration may be a concern for the administrator of an open demonstrator. To meet this need publish levels were created for widgets. In this workflow new widgets are not automatically live, but are pending publication. Only the creator can see them and edit them. They can also set them as published when happy with them. That generates a request to an administrator to review the widget and accept or reject it. The widget can be set as published or unpublished. In the default configuration of the iTEC Widget Store the publishing workflow option is turned off, leaving direct publication is in the hands of the user, following the iTEC philosophy of open, crowd-sourced, community based publishing.

Discovering Widgets

In addition to managing and finding the widgets that they have created, users also often want to find known widgets from other sources, or to discover useful widgets. A range of tools has been developed to support users in doing this, and they may be categorized as follows.

Firstly, personal tools enable users to gather and describe their widgets as they wish. Firstly, they can gather them into a 'My favourites' collection on the front page of the store. This is helpful for their own reference, but it also becomes a searchable resource available under the 'Favourites' tab, enabling users to browse through the favourites collections of other users. Secondly users can add searchable tags to their widgets, making use of the tag cloud built into the Store.

Secondly, the Store defines groups of categories which can be applied by users to their widgets when they create them. Several users from different user groups had requested better searching and categories, and in response a faceted search interface for categories was designed in order to make searching simpler and more meaningful to users. Twenty-four categories in three groups were added the store, with which the creator of the widget can categorize widgets with multiple categories in their 'my widgets section'. The categories and groups can only be edited by the system administrator. Within each group of categories the discovery is accumulative, and between the groups it is subtractive. This faceted approach allows users to tailor the discovery of the widgets to best suit their needs. It acts as way of filtering down to the subjects, skills and age ranges in which you are interested.

Thirdly, the administrator can also designate certain widgets as being 'featured'. These widgets are then available to users through the 'featured widgets' in the main Widget Store tab.

Fourthly, automated discovery was supported by an API which exposed user descriptions of widgets using the iTEC taxonomies that describe the functionalities of tools. These are not specific to widgets, but rather describe the functions of tools in a general way in order to maintain maximum flexibility. This enables tools to be

described in terms of 'what needs to be done' in an activity without specifically identifying individual tools. This work built on the approach adopted by the EU-funded iCAMP project (see http://www.icamp.eu) and the iCAMP tools have formed the principal inspiration behind the approach to tool description adopted in iTEC. Thus the technical description of tools (their operating environments, language, interoperability capabilities/requirements, etc.) was been separated from a description of what they do, and the iCamp approach of 'Soft Ontology' (iCamp 2006) was followed to identify the 'things to be done'. Interfaces for defining functionalities were built into the Store, with sliders with which users indicated the degree to which a widget provided a functionality. The information generated was made available as one of the services harvested by the iTEC SDE recommender service. Because the same taxonomy is used in other parts of the infrastructure, widgets could be mapped to the functional requirements of learning scenarios and learning activities. In this way the project created an over-arching architecture which related scenario description through to the instantiation of tools in technical settings. For further details of this aspect of iTEC work, see Chap. 6.

Moving Beyond Widgets: IMS LTI Compatibility

The Widget Store architecture has been designed so that the functionality offered by the store is entirely separate from the tools and resources which it makes available. This greatly increases in the range of contexts within which the Store can be usefully applied, with consequent benefits to its future viability. This flexibility was put to the test late in the project, when two separate factors indicated to the development team that it would be valuable to adapt the Widget Store so that it could work with the IMS Learning Tools Interoperability (IMS Global Learning Inc. 2010–2012). This specification shares some aspects of the widget approach, in particular a unique identifier which is passed via web type services to instantiate some web content within a frame or via browser redirect. The specification has been adopted by a number of online learning environments, such as Blackboard and Canvas, by tools producers, in particular by eBook providers. One factor which has driven this adoption is that LTI includes a secure, extensible model, which allows online objects to be sold between provider and consumer.

The first indication that the inclusion of LTI in the Widget Store would be valuable came from the inclusion of LTI services in the .LRN platform as part of iTEC pilots in Austria. This produced very promising results, and a higher level of engagement by teachers and institutions than widget-based services had been able to achieve. The second factor was that IMS Global Learning Inc. established the IMS Community App Store Architecture (CASA) initiative, which was announced at San Diego in May 2013 (IMS Global Learning n.d.). LTI has been successful as a means of enabling publishers to market their content to educational institutions while ensuring that the publishers maintain control over access to the materials.

However, the relationships involved are always between the consumer and an individual publisher; there is no marketplace where a range of possibly interesting LTI resources are made available to a teacher and presented according to the teacher's profile. This is the mission of CASA, and the LTI capabilities of the final release of the Widget Store fulfil this role. In the light of these two factors, the final release of the Store under the umbrella of iTEC added the capability to both consume and produce LTI tools.

This was not the first extension of the Store's ability to work with formats beyond its native W3C widgets and Open Social Gadgets. During the development cycles formats such as Flash, Java applets, embed scripts, ZIP folders with web sites and web addresses were added. However, these formats were invariably converted by the store into W3C widgets and stored within Wookie. LTI required a different approach as the actual content of the tool remains with the tool producer and is referenced by the host environment using a key and secret combination to secure the content. The store could already be consumed via LTI, this had been added when Edukapp was first developed and this allowed the store to be embedded in an environment that support LTI Basic or LTI version 1. Wookie could also produce LTI so any Wookie widget could be consumed via LTI. The big barrier though was that the store initially could not itself consume other LTI tools and include them in its listings, search engine or associate any of the ontological data or para-data with them.

As far as the user is concerned all tools within the store look the same. With LTI tools there are some additional complexities to accommodate when using the tools from the store, related to the additional security required to view content via LTI. These complexities arise only when installing the widgets/tools from the store into a host environment. The Store could be used by an institution in wide variety of ways. For example, it might be used as collection point for an institution to host and provide their own set of tools; it might be used by a tool producer as a catalogue of the tools they publish or it might be used by a tool reseller. There are three possible cases for the installation of the Store. Three scenarios were defined which anticipate how the store might be installed to cover this range of uses:

- Case 1: The store is installed on the same host as the shell. In this case the administrator of the store can only include LTI tools into the store using a key and secret supplied by the tools supplier. The user of the store can include and use the tools without having to worry about the key and secret.
- Case 2: The store is installed on the same host as the tool provider. In this case the administrator the store sets a special provider key and secret in the store configuration. The user of the store can only use or install a widget in their shell with a key and secret supplied by the tool (and store) provider.
- Case 3: The store is installed on a separate host to both the tool provider and shell. In this case the administrator of the store needs a key and secret from the tool producer to include the tool in the store. The user of the shell needs a key and secret from the store host (reseller key) to install the tool in their shell.

Conclusions

Like the rest of the infrastructure developed by iTEC, the Widget Store was designed to provide technical support for the pilots, which would develop an improved understanding of how to introduce information into schools in an effective way. Consequently much of the insight generated by the Widget Store may be subsumed in the results of that pilot program. We have also published elsewhere and the barriers which we encountered to adoption by teachers of the Widget Store, setting this in the context research lines which led to the Widget Store, and wider issues in the adoption of TEL. Readers who are interested in this wider discussion are directed to the papers on this topic by Johnson (2014) and Griffiths and Goddard (accepted for publication). There are, however, a number of lessons learned which are more specific to the technology, and to its affordances for education, which are worth drawing out here.

First, the technical strategy adopted by the project was justified, as the system provided all the functionality foreseen by the project plan, and indeed went substantially beyond this. The connector framework and the Widget Store not only fulfilled their functional requirements and performed well, they also led directly to major changes in Apache Wookie.

Second, the architecture, features and design of the Widget Store constitute a finding concerning the most effective architecture and methods to be used in managing and delivering widgets. This addresses the technical questions raised in the introduction to this chapter, and is based on extensive technical evaluation and pilots. This work also had practical implications for open source code projects beyond the project, in particular the deprecation of the user interface to Apache Wookie, and its replacement with an API which could be accessed by an app store, and the major restructuring of Edukapp.

Third, the choice of the W3C widget specification as the underlying interoperability specification for iTEC, as a means of gathering third party content, has not proved to be successful. The specification was chosen in the belief that it would become widely adopted on desktop and mobile platforms, providing many resources and services to consume. Indeed, when the iTEC project was planned W3C widgets were the format for Opera mobile apps and seemed well positioned to become a successful exchange format for web apps on multiple platforms. However, the business model adopted by mobile providers has given them no reason to welcome an interoperability specification, which could threaten the competitive advantage which they hope to gain from their own exclusive catalogue of apps. Consequently, the specifications for Web apps adopted by each provider vary slightly to ensure that interoperability cannot become a reality, even though at the technical level the tasks that they perform are quite similar. As a result there has not been a flow to iTEC of services and resources from the expanding mobile and tablet platforms. The shift away from the PC also had an impact on the iTEC interoperability strategy. The expansion of the use of ICT in schools was dominated for a decade by Virtual Learning Environments running on PCs, and projects which supported this platform

could be confident in achieving strong penetration in the education market. In recent years, however, the technical environment of eLearning has changed, and the Virtual Learning Environment (VLE) is no longer seen as a leading context for innovative technical development or teaching practice. Indeed, in many cases the need for a VLE has been questioned. This was not unforeseen by iTEC, and the choice of W3C widgets as an interoperability specification, and the development of the Widget Store were both in part intended to unite mobile and VLE platforms. Moreover, VLEs are mostly open systems, the increasingly dominant mobile and tablet platforms are closed, due to the strategy of each provider to capture and maintain a sector of the market. The consequence for iTEC was that while VLEs can be easily adapted to work with the Widget Store and can be administrated and configured locally, or at regional level, the incorporation of the Widget Store into mobile platforms is much more problematic, as administration and configuration of the system is largely restricted to commercial providers.

Fourth, in seeking to overcome the consequences of the failure of the W3C widget specification to achieve widespread adoption, the Widget Store developed innovative functionality, which has potential value within education. The content creation tools we have developed enable the W3C widget specification to be used in a different way. As an alternative to being a means of offering interoperability between different widget publishers, the specification has been used to enable individuals to encapsulate resources and services which they find useful from anywhere on the Web, to re-publish these as widgets, and to embed them within a wide range of Web applications. Where these resources do not exist, the user is supported in publishing their own, using the 'mini web site' creation tool. This is combined with the ability of the Widget Store to describe and discover widgets in a number of ways, as described above. Evaluation carried out by iTEC showed that individual teachers were comfortable with using these tools in training sessions, and many of them could see that they could be valuable, but they did not move on to making use of them in their own practice. It seems that the functionality that was offered did not make a very convincing case to the individual teacher. Indeed, on the one hand the final round of evaluation reported that teachers who were not technologically oriented had difficulty in understanding the purpose and functionality of the technology, and/or were defeated by inadequate network connections. For example, the idea of embedding content, rather than linking to it, was new for some teachers and proved to be a challenge. On the other hand teachers who were experienced users of technology had established habits, and often preferred to stick to the tools they already knew and trusted. Thus although the Widget Store has features which could be of value to individual teachers, its use may appear to those teachers as responding more the needs of the researchers who developed it then it does to their own needs. Nor does it enable them to carry out their core tasks in ways which are difficult or impossible to achieve by other means. The iTEC National Coordinator in Italy offered a suggestion for the use of the Store which is in line with our own rationale for an alternative use of W3C widget tools, saying that "the ways in which it is used need to be expanded beyond using the widget store simply to search for useful

content, for example, by focusing on the ability for teachers to share content via the widget store." However, the sharing practice is not usually a priority for individual teachers, It is, however, a major concern for pedagogic coordinators at the level of department, school or ministry. The Widget Store it possible to share and describe sets of resources which consume live services from the Web, and to embed these in training resources and in classroom practice. From this perspective it is not surprising that the most successful deployment of the Store during the iTEC pilots was achieved when its use was driven by the Ministry of Education in Portugal,[10] which provided local support and technical leadership for a community of teachers around the widget store, who created widgets from available resources and embedded them in blog posts which shared the way in which they were used. Had the pilots of the Widget Store focused on this use case more strongly at an earlier stage the Widget Store might have achieved higher levels of use. However, clarity about who benefits from a technology is often elusive, and particularly within a project with a strong focus on activity within the classroom.

References

Anido Rifon L et al (2013) D10.3 support for implementing iTEC engaging scenarios V3. ITEC Project. http://itec.eun.org/web/guest/deliverables. Accessed 2 July 2015

Dalziel J (2003) Implementing learning design: the learning activity management system (LAMS). In Crisp G, Thiele D, Scholten I, Barker S, Baron J (eds) Integrate, impact: proceedings of 20th annual conference of AscilitE, Adelaide, pp 293–296

European Commission (2010) Grant agreement for the iTEC project, Annex 1 description of work

Griffiths D, Beauvoir P, Liber O, Baxendale MB (2009) From reload to recourse: learning from IMS learning design implementations. Dist Educ 30(2):201–222

Griffiths D, Johnson MW, Popat K, Sharples P, Wilson S (2012a) The Wookie widget server: a case study of piecemeal integration of tools and services. J Univers Comput Sci 18(11):1432–1453

Griffiths D, Johnson MW, Popat K, Sharples P, Wilson S, Goddard T (2012b) The educational affordances of widgets and application stores. J Univers Comput Sci 18(16):2252–2273

Griffiths D, Goddard T (accepted for publication) Understanding teachers resistance to adopting educational technology. In Kybernetes, Special issue: Proceedings of ASC 2014, Washington, DC

iCamp (2006) Deliverable D2.2 iCamp building blocks. http://www.icamp.eu/wp-content/uploads/2007/05/d22___icamp___building-blocks.pdf. Accessed 24 Jan 2015

IMS Global Learning Inc. (2010–2012) Learning tools interoperability. http://www.imsglobal.org/toolsinteroperability2.cfm. Accessed 24 Jan 2015

[10] We would like to thank Fernando Rui Campos of the Direção de Serviços de Projetos Educativos, Direção-General da Educação, Portugal, for his sustained enthusiasm for our work, and imaginative exploration of its capabilities.

IMS Global Learning Consortium (2015) CASA Community App Sharing Architecture. http://www.imsglobal.org/casa/. Accessed 2 July 2015

Johnson M (2014) Learning design, social ontology and unintended functionalism in the ITEC project. In Design4learning conference, Open University UK. http://dailyimprovisation.blogspot.com.es/2014/09/learning-design-social-ontology-and.html. Accessed 24 Jan 2015

Koper R, Tattersall C (2005) An introduction to learning design. In: Koper R, Tattersall C (eds) Learning design: modelling and implementing network-based education & training. Springer, Berlin-Heidelberg, pp 3–20

Kroop S, Mikroyannidis A, Wolpers M (eds) (2015) Responsive open learning environments: outcomes of research from the ROLE project. Springer International Publishing, Berlin

Laurillard D, Charlton P, Craft B, Dimakopoulos D, Ljubojevic D, Magoulas G, Masterman E, Pujadas R, Whitley EA, Whittlestone K (2011) A constructionist learning environment for teachers to model learning designs. J Comput Assist Learn 29(1):15–30

Neumann S, Oberhuemer P (2009) User evaluation of a graphical modeling tool for IMS learning design. In: Spaniol M, Li Q, Klamma R, Lau RWH (eds) Advances in Web based learning—ICWL 2009, vol 5686, Lecture notes in computer science. Springer, Berlin, pp 287–296

Olivier B, Liber O (2001) Lifelong learning: the need for portable personal learning environments and supporting interoperability standards. Cetis Whitepaper. http://wiki.cetis.ac.uk/images/6/67/Olivierandliber2001.doc. Accessed 24 Jan 2015

Sharples P, Griffiths D, Wilson S (2008) Using widgets to provide portable services for IMS learning design. In Koper R, Stefanov K (eds) Proceedings of the TENCompetence international workshop on stimulating personal development and knowledge sharing. TENCompetence, pp 57–60. http://dspace.ou.nl/handle/1820/196. Accessed 24 Jan 2015

TENCompetence Foundation (2010) TENCompetence: building the technical and organisational infrastructure for life long competence development. http://hdl.handle.net/1820/2432. Accessed 28 Feb 2015

W3C (2011) Widget packaging and XML configuration, W3C recommendation 27 Sept 2011. http://www.w3.org/TR/widgets/. Accessed 24 Jan 2015

W3C (2015) OpenSocial foundation moves standards work to W3C social web activity. http://www.w3.org/blog/2014/12/opensocial-foundation-moves-standards-work-to-w3c-social-web-activity/. Accessed 28 Feb 2015

Wilson S, Sharples P, Griffiths D, Popat K (2011) Augmenting the VLE using widget technologies. Int J Technol Enhanc Learn 3(1):4–20