A Community-Centred Design Approach for Accessible Rich Internet Applications (ARIA)

Dr Steve Green
Accessibility Research Centre
Teesside University
Middlesbrough, UK, TS1 3BA
s.j.green@tees.ac.uk

Dr Elaine Pearson
Accessibility Research Centre
Teesside University
Middlesbrough, UK, TS1 3BA
e.pearson@tees.ac.uk

Dr Voula Gkatzidou
Accessibility Research Centre
Teesside University
Middlesbrough, UK, TS1 3BA
s.gkatzidou@tees.ac.uk

Franck Oliver Perrin
Accessibility Research Centre
Teesside University
Middlesbrough, UK, TS1 3BA
f.perrin@tees.ac.uk

There are a number of emerging standards and guidelines which help the web developer or learning technologist produce inclusive static and dynamic internet-based applications which will meet the needs of users, regardless of their special needs or individual requirements. These standards and guidelines typically assume that the needs of the individual user are well defined, that the function of the web application is clear and that appropriate adaptations can be readily applied. However this also puts a heavy burden on the skills and knowledge of the developer and fails to utilise the expertise of tutors and other members of the community for what is potentially a very wide range of users and individual needs and requirements. Consequently this research suggests an approach which combines the benefits of using formally specified standards-based components in the form of W3C Widgets and Accessible Rich Internet Applications (ARIA) with a Community-Centred Design approach based on the UK JISC funded projects WIDE (Widgets for Inclusive Distributed Environments) and WIDGaT (a Widget Authoring Toolkit). This work forms part of a wider research topic on adaptable personal e-learning and e-media.

Accessibility, adaptability, disability, community of practice, user-centred design, personal learning.

1. INTRODUCTION

Authors of e-learning, e-media or web content usually refer to the W3C Web Content Accessibility Guidelines (WCAG, 2009) if they understand the need to produce content which is accessible to a wide audience. WCAG 2.0 provides extensive guidance on how to make static content accessible to people with a broad range of disabilities and as such can often be applied to static e-learning content too. However when we are producing dynamic content or an application which can be described as a Rich Internet Application (RIA), usually a widget, app or gadget, then the W3C WAI ARIA (2009) standard (Accessible RIA) might prove to be more appropriate. In some educational contexts where the content or application forms part of a Personal Learning Environment (PLE) (JISC 2009, van Harmelen 2006) then the ISO/IMS standards for Accessibility Metadata (ACCMD, IMS 2004) or user profiles (ACCLIP, IMS 2003) might be more specific to the problem. However the average web designer is unlikely to be fully familiar with all these standards and guidelines. They are simply too detailed and too many. If e-learning authors and web designers themselves find these standards difficult to follow then clearly it would be unreasonable to expect tutors and educationalists to understand them. However these are often the very people who have the expert knowledge of the needs of their learners. Consequently if we wish to involve the wider community in the design of accessible rich internet applications and media we need to develop a method which incorporates within the process the basic principles which these standards represent.

For this reason this research has taken two parallel approaches to its definition of e-learning content and applications. One approach is based on the formal specification of an e-learning framework using traditional modelling techniques such as Unified Modelling Language (UML) and Artificial Intelligence languages. This method has a close affinity to the standards on which it is based. However the second is a User-Centred Design approach based on a community of practice. This is what we refer to as Community-Centred Design. The authors believe that the combination of both formal and user-centred approaches has much to offer and can achieve a useful marriage of theory (standards, AI and formal specification) with practice (HCI, user needs, community design and development). The culmination of this was the Joint Information Systems Council (JISC) projects Widgets for Inclusive distributed Environments (WIDE) (ARC, 2010) and the follow up project widget authoring toolkit - WIDGaT. This paper looks at how inclusion, personalisation and adaptability can be designed into hypermedia to produce community designed widgets based on accessible rich internet applications. We will begin with a brief description of the guidelines and standards on which this work is based and then discuss the inclusive e-learning context. We will then go on to present our community-centred design approach applied to widget design and conclude with a discussion and evaluation of widget production and a preview of our WIDGaT authoring toolkit.

2. ACCESSIBLE E-LEARNING

The standards and guidelines which are relevant to the wider research on accessible e-learning are listed in table 1 below. They are primarily based around different types of static and dynamic web-based content and applications and the need to make hypermedia universally accessible (see table 1) (Lazarinis *et al*, 2011).

Table 1: *Accessibility standards and guidelines*

ACCMD	IMS *AccessForAll Meta-data* Information Model
ACCGuide	IMS *Guidelines* for Developing *Accessible* Learning Application
ACCLIP	IMS Learner Information Package *Accessibility for LIP*
APLR	*CEN-ISSS Learning Technologies Workshop Accessibility Properties for Learning Resources*
ARIA	*W3C/WAI Accessible Rich Internet Applications*
WCAG	*W3C/WAI Web Content Accessibility Guidelines*
W3C Widgets	*W3C Widget Standards (HTM5, CSS3 and JavaScript)*

There are also a number of other important e-learning standards not listed above which are relevant to the specific context of e-learning (IEEE 2002, IMS 2003, Lazarinis et al. 2011). The W3C Widget standards are included here because W3C Widgets are inherently adaptable and arguably, therefore contribute to the wider debate on inclusive web or e-learning design.

2.1 W3C Widgets

W3C Widgets is a recent standard (September 2011). While there is no current reference model there are two main examples: one is the Apache incubator project Wookie (2009) and the second the Opera developers' community (2010). They are typically known as Wookie and Opera widgets respectively but both basically follow the W3C standards and recommendations. Widgets are not specified with accessibility in mind, but because the standard is relatively simple, designed for personalisation and re-use and for a desktop or mobile context they are inherently adaptable. They are meant to provide a common design solution for a range of platforms. Most modern browsers (any that support HTML5) and many mobile devices can already support W3C widgets and more device profiles are being added to the list every day. Widgets, as defined by the W3C standard, are based on web browser technologies. They are zipped packages of files and directories usually with a *.wgt* or *.zip* extension. They would usually comprise at least four separate components:

- an XML-based configuration file *config.xml*
- an HTML5-based start page *index.html*
- a CSS3-based style sheet
- and a JavaScipt file

The XML configuration file defines the basic features, dimensions and startup operation of the widget. The HTML index file contains the main widget opening page. The CSS file has the style information for the document objects and the JavaScript file contains the code to handle widget events, API and web service access and any general interaction of the widget. In addition there could be many other HTML pages, JavaScript libraries further CSS and of course directories of media and resources. Some widgets will be entirely self-contained, installable on the device or desktop, others will need access to external media and services either locally served from the widget repository or through the internet. Widgets can typically make use of APIs and web services (usually JSON or REST) to take advantage of existing information or open access resources. An example would be using Google maps or geo-location services through Google APIs (2009).

Because widgets are based on existing web technologies the development process should be reasonably straightforward for web developers and programmers familiar with scripting or an object-oriented language like C++ or Java. W3C widgets come in a number of flavours. For example Wookie widgets (Wookie, 2009) run from a web-server allowing for ready communication between users and instances of the same widget. Opera widgets

(Opera, 2010) on the other hand can be directly installed on a desktop or mobile device allowing a greater flexibility in off-line or personalised use. However all widgets developed to the W3C standard are very easy to adapt to different platforms or contexts. It is this ease of adaptation which makes them of particular interest to our context of accessible personal learning. Next we consider the accessibility specific ARIA standard.

2.2 W3C/WAI ARIA

ARIA presents guidance on the way that dynamic web-based material should be designed and presented for assistive technologies. Unlike IMS *AccessForAll (2003, 2004, 2009)* which deals largely with e-learning content, WCAG (W3C 2009) and ARIA (W3C/WAI 2009) try to consider all types of web-based application. WCAG has a large number of recommendations for web-designers and web developers culminating in a range of web audit checks relevant to static content. However here we are primarily concerned with dynamic content and small applications including widgets. We also need to be able to handle adaptations and contextual elements (cf. Sloan *et al.* 2006). As a result we need to consider the W3C Web Accessibility Initiative (WAI) standard for Accessible Rich Internet Applications (ARIA). This standard has now been included into W3C HTML5.

What ARIA does (to add to WCAG) is to deal with dynamic web-based content such as widgets, apps, gadgets or tools that typically work in a way similar to a standard desktop application. These are the kind of tools that tend to blur the distinction between the web, the PC desktop and the mobile device. ARIA basically adds or re-uses a number of attributes. The first of these is the `role` attribute. Because in RIA HTML tags might actually perform a number of functions which are not clear from the tag element, the role performs the job of indicating the tag's function. For example the `<body>` tag might be given the `role="application"` attribute to indicate that this is an application-type widget or `role="document"` to indicate that it is largely static content to be read. A later `<div>` tag might be identified as the `"header"`, `"main"`, `"menu"` or `"footer"` content.

In addition to the role attribute ARIA also adds the `alt` and `tabindex` attributes to all tags. The first basically means any document object can have an alternative text attribute (useful for screen-readers). The `tabindex` extension is less obvious. It allows any tag to gain input focus or appear in the tab-order for navigation with the tab key (commonly used by screen-readers and assistive technologies). The `tabindex` can have any one of three sets of values: a positive index indicates that the tag should be visited in the indexed tab order; a zero index indicates that the object should appear in its natural tab order; a negative index means that the object should not appear in the tab-order but it may receive focus. In practice this is often used dynamically with one of a set of controls having a tabindex of 0 and the others with a tabindex of -1. The cursor keys or other controls are then used to shift the tab focus to the other items in the tab set. This is important to allow screen-readers a hint at how content should best be navigated.

Further to these three attributes ARIA also adds a whole class of status and value properties all prefixed 'aria-'. For example there are properties such as `aria-valuemin`, `aria-valuemax` and `aria-valuenow`. These are useful in describing to assistive technology the current state or value of a control. For example there may be a slider control using a graphic image. In ARIA this could be:

```
<img id="textFontSize"
  role="slider"
  alt="text size slider"
  src="graphics/slider.jpg"
  onclick="changeTextFontSize()"
  aria-valuemin="0.5"
  aria-valuemax="4.0"
  aria-valuenow="1.5" />
```

The HTML and ARIA markup define the initial values of the control as defined but it is up to the programmer to ensure that the value of the `aria-valuenow` property is kept up-to-date. In this instance a trick that the web designer may use is to have the CCS3 change the slider image based on the `aria-valuenow` property. Only by keeping the ARIA properties up-to-date and relevant can assistive technologies inspect the current state of a dynamic system.

2.3 Inclusive e-Learning

Within the context of accessible e-learning IMS AccessForAll is also a relevant. It is essentially a standard that proposes a mechanism by which e-learning content can be made universally available. It has a specific emphasis on matching media format to user needs and preferences. The IMS *AccessForAll* standards are divided up into two main components, ACCMD – metadata and ACCLIP – learner profiles. This section gives a brief overview of IMS *AccessForAll* (IMS 2004, 2009, ISO 2008). The metadata specification is based around the identification, adaptation and presentation of accessible web-based e-learning resources but much of it is equally applicable to any e-media. The standard divides resources into primary and equivalent alternative resources. The primary resource is the default whereas the alternative has equivalent 'semantic and behavioural functionality and addresses the same learning objective as the primary resource'. The *AccessForAll* overview says that the primary resource meta-data describes:

- *Access Modality*: vision, hearing, touch, text and combinations.
- *Adaptability*: display and control transformations (for devices/assistive technologies)
- *Equivalence*: any equivalent alternative i.e. an equivalent but alternative experience.

The following *AccessForAll* classes are especially important:

- The general accessibility class
- The equivalence class
- profiles of needs and preferences (PNPs)

The overall accessibility model defined by IMS ACCMD is presented below in Figure 1:

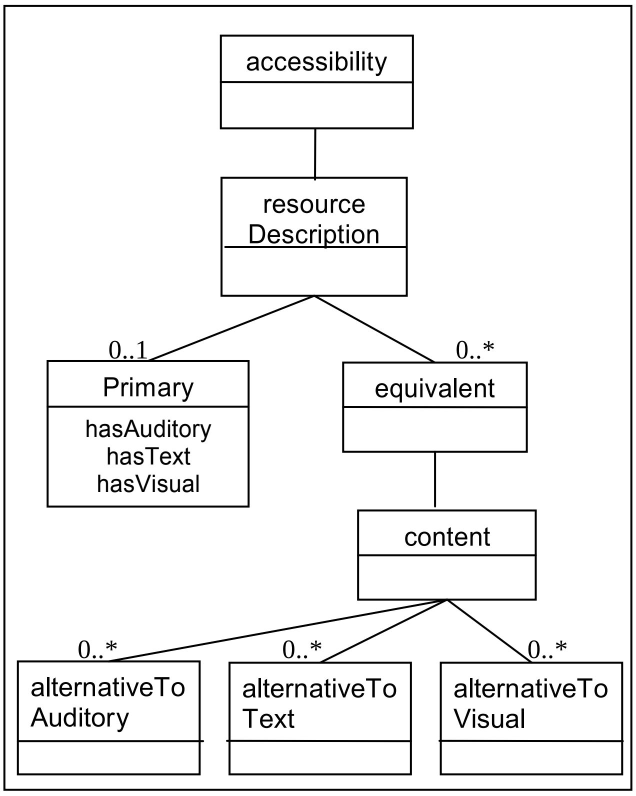


Figure 1: *overall Accessibility model*

While the IMS standards are important they rely on the e-learning author, web developer or tutor to provide a wealth of alternative media resources as alternatives to the primary resource. They also work best with learning objects or structured media which can be disaggregated into a number of media components. Consequently this standards work best when we can deal with clearly defined profiles of needs and preferences and re-usable learning patterns (Green *et al.* 2006).

2.4 Personalisation

Modern e-learning or learning management-systems expect environments to be supportive of learner preferences. In fact the JISC has commissioned a number of projects and adopted the term Personal Learning Environment (PLE) for such systems. JISC (2009) describes a PLE as one that replaces some or all of the tools of a standard Virtual Learning Environment (VLE) (or LMS – Learning Management System) with personal tools integrated with the student's own systems. These environments are therefore effectively mash-ups of components chosen by the user. The components of such a mash-up are likely to widget, apps or gadgets.In fact many software systems, e-learning based or not, try to personalise the environment to the preferences of the individual. An example of this is iGoogle (2009). In addition to the standard search box the user is presented with a range of tools and is also allowed to customize the interface in a number of important ways:

1. by choosing or creating the window theme
2. by adding, editing or moving gadgets
3. by sharing and communicating themes, gadgets and content feeds with others.

Similarly some web based applications can be automatically configured to handle small screen layouts for PDAs, mobile phones and other mobile devices. Most mobile phone manufacturers offer user-selectable themes and widgets for their customer base or in the case of the Apple iPhone thousands of applications through iTunes and the AppleStore (Apple, 2012). In a very real sense the

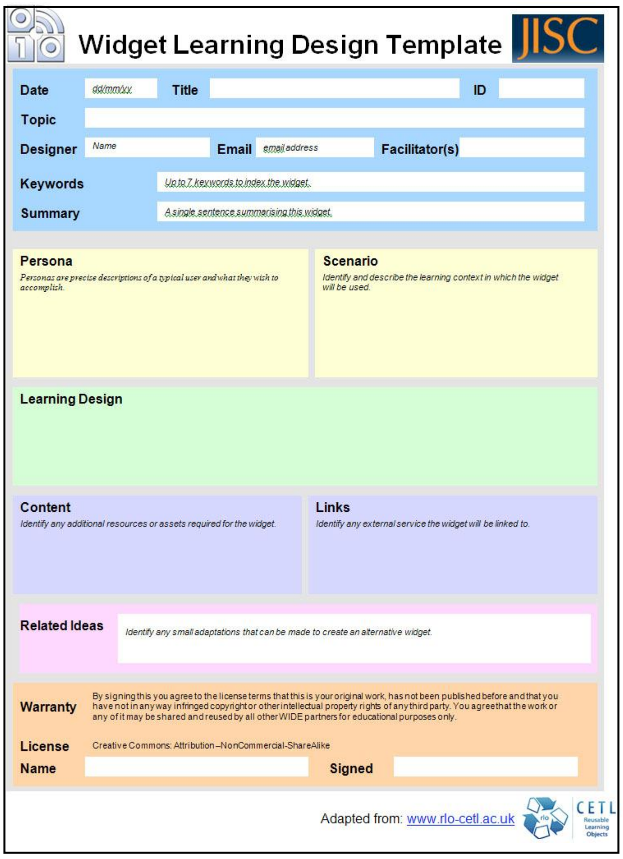
much of this thinking, but with the added dimension of an educational context. Van Harmelen (2006) believes that PLEs can be characterised by their underlying design in terms of pedagogy, personalisation and control. In principle personal learning environments typically support a collaborative, student-centred learning approach; they are normally open rather than closed systems which might even be constructed entirely from available web services; they can be personalised and the locus of control is with the user rather than the teacher or institution. Kompen (*et. al.* 2008) goes further and suggest a conceptual framework for developing a personal learning environments from Web 2.0 tools and services. Coming from a research background where we developed a specialist learning environment for students with severe cognitive and motor difficulties (Green, Pearson & Stockton, 2006) our first concern was that of dealing with need. However we had recognised that content adaptation has to be coupled with the personalisation of tools if we are to provide a truly adaptable environment and empower students to make choices.

3. COMMUNITY WIDGET DESIGN

The W3C Widget standard defines a plausible technology to provide these applications and tools. However we also wished to include our community of practice in the design of these tools. Consequently for this research we proposed and used a community-centred design approach. The process was based on widgets, existing and emerging standards, guidance and methods, and open source tools.

Our community of practice comprises those who are directly involved with the teaching or support of disabled students, together with researchers, educationalists and developers. The community participated in one of a series of tailored face-to-face workshops and on-line follow up activities. These activities were supported by dedicated web 2.0 collaboration tools including a wiki, RSS feeds and on-line resources. The result was a number of detailed designs. These were then translated into widgets created for specific learning needs. The widgets were classified and searchable by type, subject matter, complexity, adaptations and disability. They provided functions, tool, applications or facilities to help the student or tutor in the context of their education or in many cases, their everyday life. Each widget included a description of a *persona* or typical user for whom it was designed and an example *scenario*, which together made up a use-case. These use cases were an informal but very specific example profile of needs and preferences. The resulting widgets were made available for use and adaptation by the wider community under an open access, creative commons license.

3.1 Design Workshops



Widget Learning Design Template

Date	dd/mm/yy	Title		ID	
Topic					
Designer	Name	Email	email address	Facilitator(s)	
Keywords	Up to 7 keywords to index the widget.				
Summary	A single sentence summarising this widget.				

Persona
Personas are precise descriptions of a typical user and what they wish to accomplish.

Scenario
Identify and describe the learning context in which the widget will be used.

Learning Design

Content
Identify any additional resources or assets required for the widget.

Links
Identify any external service the widget will be linked to.

Related Ideas *Identify any small adaptations that can be made to create an alternative widget.*

Warranty By signing this you agree to the license terms that this is your original work, has not been published before and that you have not in any way infringed copyright or other intellectual property rights of any third party. You agree that the work or any of it may be shared and reused by all other WIDE partners for educational purposes only.

License Creative Commons: Attribution – NonCommercial-ShareAlike

Name **Signed**

Adapted from: www.rlo-cetl.ac.uk

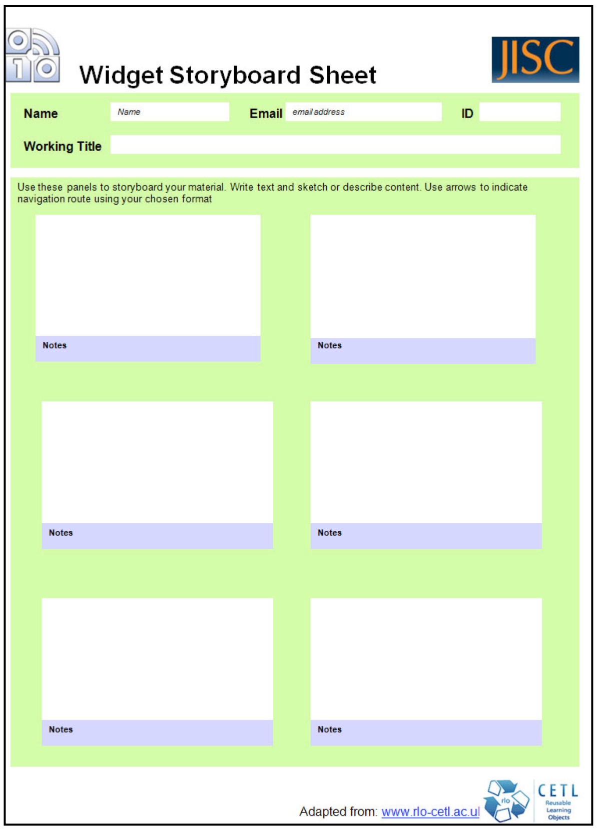
Figure 2: *Widget Learning Design Template*

The workshops were run three times in different locations and involved a total of 11 groups of four to six people. Each workshop had the format:

1. Introduction to Widgets and an overview of
   a. open source software for accessibility
   b. mobile prompts
2. Group activities
   a. widget design brainstorming
   b. working together on poster templates
3. Presentation and discussion of designs
4. Widget development and evaluation plans

During the design process each group considered five or six ideas and would then choose one idea to develop further with the aid of learning design and storyboard templates. For these A0-sized laminated posters were used (see figures 2 and 3). This produced a total of around 30 designs. During the workshops the learning design templates (see figure 4) were completed first giving us:

- Basic details (title, date, topic etc.)
- Persona – description of the typical user
- Scenario – the context or example use
- Learning design - the widget function
- Content – resources or assets needed
- Links – external web services or links
- Related ideas – adaptations & other uses



Widget Storyboard Sheet

Name	Name	Email	email address	ID	
Working Title					

Use these panels to storyboard your material. Write text and sketch or describe content. Use arrows to indicate navigation route using your chosen format

Notes

Notes

Notes

Notes

Notes

Notes

Adapted from: www.rlo-cetl.ac.ul

Figure 3: *Widget Storyboard sheet*

On the storyboard sheets (see figure 3) the participants were asked to illustrate the widget design interface and operation. Use of colourful A0 laminated templates encouraged creativity and group cohesion. This approach was adapted from the learning object design approach used by the RLO-CETL (Leader, 2009). However many of the elements such as personas and scenarios are common in user-centred or user-experience design and the broad design approach matches the JISC Users and Innovation Design Method (JISC, 2008)

3.2 Development Phase

Each of the widgets was classified in terms of a number of factors grouped into:

- technical elements,
- display features
- application compatibility
- development complexity.

An important consideration was their perceived development complexity namely simple, moderate and complex (see table 2). The complexity of the widget could be inferred largely from its design features (typically whether self-contained, use of APIs and web services etc.). Often simple widgets could typically be programmed in a day or two. Moderate (or medium complexity) widgets could take a few days or a week and complex widgets two weeks or more to develop. In addition to the basic widget a number of templates or widgets based on alternative content or interfaces were also identified as part of the process. Many of these templates were later incorporated into the WIDGaT toolkit.

Table 2: *WIDE widget by platform/complexity*

Platform	Simple	Medium	Complex	Total
Wookie	11 (36%)	11 (36%)	1 (3%)	23 (74%)
Opera	4 (13%)	1 (3%)	1 (3%)	6 (20%)
Windows	2 (7%)	0 -	0 -	28 (78%)
All	17 (55%)	12 (39%)	2 (7%)	31 (100%)

Our approach to widget classification was useful for predicting development time. However the development of even simple widgets might be held up waiting on further content or design decisions. Also the classification of development complexity could depend on the delivered widget context. Consequently widgets were also classified as:

- Wookie: W3C widget running from a Wookie web server.
- Opera: W3C widget but benefitting from installation on a desktop or mobile devices.
- Windows app: not W3C widgets but desktop applications designed to install and run on a specific system (MS Windows)

In practice the majority of designs (74%) could be implemented as Wookie widgets but a handful required facilities that weren't available to the widgets standard APIs so needed to be Opera only or Windows apps. The widgets were almost all classified as simple or moderate (94%) with very few (6%) considered complex.

Table 3. *WIDE: widget by category*

Categories	Total	Examples
Time and Task Management	9 (29%)	*One-click timer Virtual Shopping*
Independence & Social Networking	7 (22%)	*BSL Signing Community Active*
Assistive Technologies	5 (16%)	*Coloured Overlay Magnifier*
Learning Aids	5 (16%)	*Spell It Translate it*
Learning Objects and Content	5 (16%)	*Sentence Jumbler Quick Revision*

In terms of the categories of widgets (see table 3) the majority were either time or task management (29%) or independence tools (22%) which together accounted for slightly over half of the widget designs. The other three categories, namely assistive technologies, learning aids and learning objects (or content) accounted for the remainder with five designs each (16%). Examples of each category of widget are available on the WIDER resource (ARC 2011) for comment and use by the community.

3.3 WIDE Project Initial Evaluation

The initial WIDE evaluation was largely confined to reviewing the widgets themselves. This has turned out to be an on-going process with comments still coming through our WIDE community wiki and the WIDE resource site referred to as WIDER. However the summative WIDE project evaluation looked at the overall process, in terms of the original use of the wiki (its suitability as a platform for promoting collaborative design) the widget learning designs (templates and design process) and the implementation methods employed. For the final part of the evaluation study a widget learning design template was provided and an external evaluator invited to adapt it. The evaluator found the process simple, the resulting widget quickly delivered and of high quality but commented on the difficulty of following the wiki. This is probably not a surprise since the wiki was provided for the community of practice and made the assumption that participants had attended a workshop. However in the light of these comments the alternative WIDER repository is now available.

Initial findings suggest that the design approach involving dedicated workshops and a community of practice proved to be effective in that all forty participants' expectations (100%) were met or exceeded. Following on from the workshops participants were kept up-to-date with the development of their widget designs through the dedicated WIDE wiki, RSS feeds and through directed e-mails. In addition further groups were added to the community of practice through the JISC JORUMopen facility. This approach has been presented to the JISC community on a number of occasions and has been adopted by several research consortia (e.g. ROLE and EDUKAPP). A number of minor issues were identified. Firstly there is greater need for technical developer involvement to improve the speed at which individual widgets can be built. Some of the designs needed to be more thorough before they could be passed on to the developer. It was also apparent that some simple tools which would allow tutors to adapt widget templates or update content themselves would greatly increase the re-use of existing widgets and templates.

4. WIDGET AUTHORING and WIDGaT

Following on from the WIDE project a number of evaluation studies were carried out with groups of web developers to see whether the community design method used within the WIDE project could be replicated in a more traditional program or web design context. Here we report on our findings with a group of 36 undergraduate students, and some preliminary results from a further group of 35. In addition to this we take a brief look at the development of the WIDGaT authoring toolkit, designed to allow novice users to create widgets.

4.1 Widget Development Evaluation

Our first group of students were drawn from a number of final year computing courses in specific disciplines including web and multimedia design, creative digital media, web development and computer programming. A significant number (5 out of 36) had a recognised disability or special need (namely dyslexia, dyspraxia or motor disability). All students were studying an elective module on Accessibility and Adaptive Technologies. Each student was asked to identify a problem based on a persona and scenario. This could be an individual idea or an adaptation of one of the use-cases available on the WIDE project wiki. Almost all chose to develop a new use-case. The students then proceeded to go through the widget design process in a way that mirrored the community-based design activity. At the end of this they then continued to a detailed design and built a widget or a widget prototype, depending on the complexity of the widget design and their level of skill. For those with more creative design skills they had the option to concentrate on the visual and interface design elements although in fact all students chose to deliver a widget or prototype of some form.

Following each stage (over a month in total) tutor feedback was given based on the areas that they could improve. Guidance was given on technical issues, which they would be unlikely to be familiar with (such as the use of web services and APIs). At the end of the process they uploaded a final design document, the widget, prototype or detailed design and a reflective report. The student's attempt at designing and developing a widget design and a W3C compliant widget was then categorized by dividing the process into a number of stages and evaluating each as not-met, part-met or fully-met against a set of predefined criteria. This is the method that WCAG and other guidance checklists deal with accessibility audit and evaluation. The results of this evaluation are given in table 4.

Table 4. *Widget Design/Development by Element*

Design/Development Element	Fully Met	Part Met
Use-Case (persona, scenario)	28 (78%)	8 (22%)
Storyboards	20 (56%)	16 (44%)
Widget Classification	7 (19%)	28 (78%)
Graphical User Interface	32 (89%)	3 (8%)
Adaptations	24 (67%)	11 (31%)
Widget Development	4 (11%)	16 (44%)

In summary the students had no difficulty in following the design approach with the majority producing full use-cases (78%) and detailed storyboards (56%). From this they could typically also produce meaningful graphical user interfaces or visual prototypes (89%). The two areas where they were less successful was in classifying the widget (only 19% had a thorough classification) or producing a fully W3C compliant widget (11%). To some degree this might be put down to the fact that students could often successfully preview a widget that would not be valid or which had been inaccurately classified (in terms of their XML configuration or metadata); they did not understand the need for correct XML, identification of features and packaging for the widget to run in a specific context (such as on a Wookie widget server, as an Opera widget or on a specific device). However despite this most made a good attempt and appeared from their reflections to have found the process novel, challenging and rewarding.

The preliminary findings from a second group of 35 final year undergraduate students suggest similar results at least in the case of the widget design phases. However this second group were more successful in developing working widgets with 7 out of 35 (20%) producing fully compliant widgets and most an acceptable prototype. However the majority of students were still unable to follow fully the standards for W3C Widgets.

4.2 WIDGaT -Widget Authoring Toolkit Given the fact that even relatively skilled web designers and students may not be able to consistently develop fully compliant widgets, we would not expect novices to do so. What novices may be able to use is a widget authoring tool which can adapt to their level of expertise and allow them to express their own designs and author their own widgets. These widgets would typically be based on templates - hence the WIDGaT authoring toolkit.

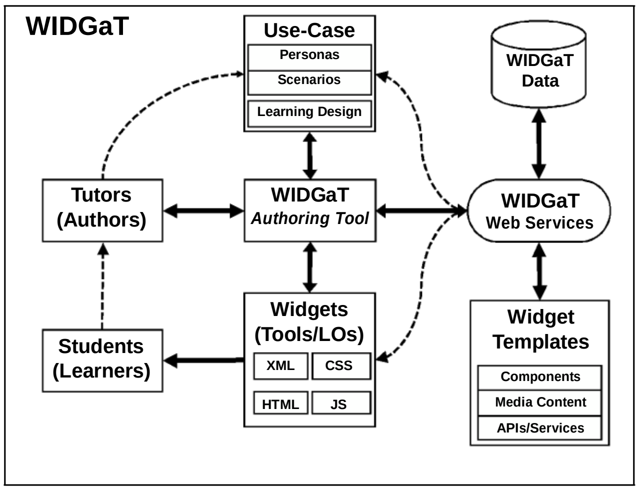


Figure 4: *Widget Authoring Services*

WIDGaT is designed to allow non-technical users (such as tutors or other members of our community of practice) to design and build simple widgets. The widgets can be easily adapted to different use-cases, contexts and devices, which is what makes them accessible. They conform to W3C ARIA and Widget standards and can be learning materials or support tools. WIDGaT is a fully working prototype designed with our community of practice and currently being evaluated by them as part of the JISC WIDGaT project. We plan to make it available as an open source project to anyone who wishes to use it over the coming months.

4.3 WIDGaT GUI
For tutors within our community of practice, the WIDGaT toolkit provides a graphical user interface by which they can realise their widget designs. Having invoked the WIDGaT toolkit (a simple web reference from a compliant browser) then the user is presented with a range of templates from which they must select. They are then given some simple dialog screens in which they can choose or edit an appropriate use case (persona and scenario) and provide their own author details (name, e-mail, organisation, link) if they wish. They can skip this stage but they will need to add these details later for the benefit of other members of the community.

Once they are happy to proceed they are then presented with the authoring screen (see figure 7). This is where they can make changes to their template by selecting and editing components, and choosing themes or alternate styles.

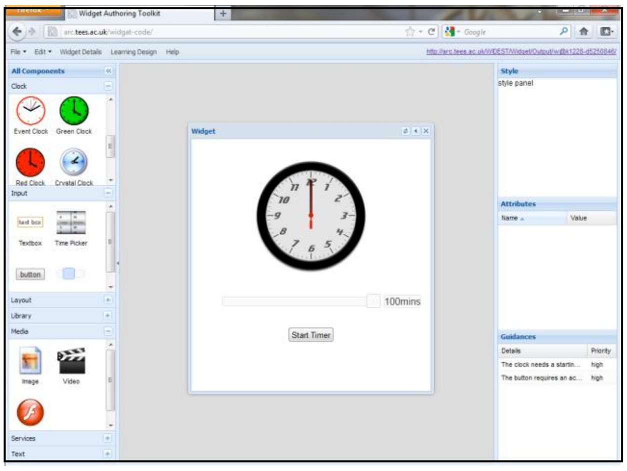


Figure 5: *Widget Authoring Toolkit (WIDGaT)*

Currently we are building up a number of use-cases and templates drawn from our community of practice including our student community and project evaluators. As we might have anticipated the same categories of widget, such as time and task management, keep re-occurring allowing us to concentrate on providing a limited number of very rich templates and associated components in the first instance. Advanced users will be able to start with a blank canvas but most users are expected to work from an existing template to apply their own adaptations. Consequently the richness of these templates is of primary importance.

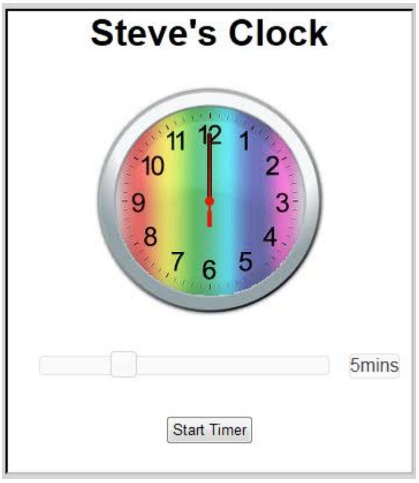


Figure 6: *Resulting Widget*

4.4 WIDGaT Technical Features
Behind the scenes WIDGaT works by providing a web application to the user's client PC which communicates by light-weight (JSON) web services with a web server. The widget can be edited as a work in progress but can eventually be saved, published to another widget server or exported as a package to work on a variety of platforms including a desktop or laptop PC, any Google Android mobile

device, a Smartphone or iOS for Apple i-phone and i-pad.

Technically the WIDGaT GUI itself is an EXT JS application which runs on a client PC or laptop or in principle on any system with a compliant browser. The GUI is designed to be an open source product. The web services currently run from an Microsoft ASP.NET web server. However the services are provided using standard JSON (JavaScript Object Notation). Consequently they could be rewritten to run on almost any type of web server. An overview of the services which the WIDGaT web server current provides is given in table 5.

Table 5. *WIDGaT JSON web services*

WIDGaT W3C Widget build services	
create	Create a new widget
duplicate	Create a duplicate widget
compose	Build a widget completely
modify	Change the specified widget
append	Add a new component/object
delete	Delete a component /object
replace	Replace a component /object
refresh	Rebuild the existing widget
update	updating to newest version
undo	Undo the last operation
package	Package as a .zip and .wgt
publish	Make the widget available
WIDGaT Information services	
info, templates, categories, components, themes	
WIDGaT display and media upload services	
display, upload, register	

WIDGaT will continue as an open source development project. The more technically minded will be able to contribute new widget templates, components, themes or media content or to contribute to the toolkit and services themselves.

5. CONCLUSIONS

Our original ideas were based on the notion of providing universally available e-learning content which can be adapted to the needs of everyone. However this was overly ambitious and failed to utilise the specific knowledge and expertise of our community of practice in the design and delivery of personalised learning experiences. Consequently the WIDE and WIDGaT projects have attempted to refocus our research by looking at specific use-cases in the form a precise personas and scenario developed into W3C and ARIA-compliant widgets. We have supplied the community with a set of tools to design their solutions and see them realised.

As part of our community-centred design approach we are developing a number of standard templates, learning design patterns and a widget component library. These templates, components and services, incorporate the ARIA and Widget standards and the metadata needed for *AccessForAll* and the TASS. While this will allow for transformation, augmentations and substitution of e-media content, the current approach with WIDGaT is to place design decisions and adaptations directly under the control of tutors and ultimately the learners themselves so that both tools and content can meet individual needs and preferences. This can be described as *access-for-me* rather than *access-for-all*. The next stage in our research is to evaluate the use of our Community-Centred Design methods and widget approach in practice within the wider accessibility and e-learning communities.

In conclusion this research has proved to have an application for accessibility in a wider context than originally anticipated. While the team set out with the idea of making rich learning materials and tools accessible as part of an adaptable personal learning environment, many of the widgets proposed were geared at allowing individuals to manage their time and resources, communicate their concerns or take ownership of the technology. This is clearly no bad thing but finally it forces us to reconsider the direction of our research.

6. ACKNOWLEDGEMENTS

The authors wish to acknowledge the aid of the JISC in funding both the WIDE and WIDGaT project. We also need to thank our partners TechDis and Portland College and the very many people from our Community of Practice who gave freely of their time, during the workshop and the WIDGaT design and prototype trials.

7. REFERENCES

Apple (2012), *Browse web apps.* Retrieved March 29 2011 from http://www.apple.com/webapps/

ARC (2011) *WIDER: Widgets for Inclusive Distributed Environments*. Retrieve March 29 2011 from http://arc.tees.ac.uk/wider/

Boni, M., Cenni, S., Mirri, S., Muratori, L. A., and Salomoni, P. (2006). Automatically producing IMS AccessForAll Metadata. In *Proceedings of the 2006 international Cross-Disciplinary Workshop on Web Accessibility (W4a): Building the Mobile Web: Rediscovering Accessibility?* (Edinburgh, U.K., May 22 - 22, 2006). W4A '06, vol. 134. ACM, New York.

Gay, G., Mirri, S., Roccetti, M., and Salomoni, P. (2009). Adapting learning environments with AccessForAll. In *Proceedings of the 2009 international Cross-Disciplinary Conference on Web Accessibililty (W4A)* (Madrid, Spain, April 20 - 21, 2009)

GKatzidou, S. & Peaeson, E. (2009). A Transformation, Augmentation, Substitution Service (TASS) to Meet the Needs and Preferences of the

Individual Learner. *Proceedings of the IEEE International Conference on Advanced Learning Technologies (ICALT). Riga, Latvia*

Google, *Gadgets *API Developer's Guide (2009),* Retrieved July 14, 2009 from http://code.google.com/intl/en/apis/gadgets/ docs/dev_guide.html

Green, S. Jones, R, Pearson, E. & Gkatzidou, S. (2006) "Accessibility and adaptability of learning objects: responding to metadata, learning patterns and profiles of needs and preferences", *ALT-J, Research in Learning Technology*, 14(1), 117-129, 2006

Green, S., Pearson, E. & Stockton, C. (2006) "Personal Learning Environments: Accessibility and Adaptability in the Design of an Inclusive Learning Management System", *AACE World Conference on Educational Multimedia (EDMEDIA)*, Orlando, Florida, USA, 2006

Harper, S., Yesilada, Y., and Goble, C. (2005). "Engineering accessible design": W4A -- international cross disciplinary workshop on web accessibility 2005 workshop report. *SIGACCESS Access. Computing.* , 83 (Sep. 2005), 64-72. DOI= http://doi.acm.org/10.1145/1102187.1102198

IEEE (2002) Standard for Learning Object Metadata IEEE Std 1484.12.1™-2002. Retrieved November 1st2005:http://www.ieeeltsc.org/wg12LOM IMS

(2004) *AccessForAll* . IMS Global learning/Dublin Core AccessForAll project, Meta-data Specification Version 1.0 Final Specification: Overview, Information Model, XML Binding, Best Practice Guide 2004. Retrieved July 14, 2009 from http://www.imsglobal.org/accessibility

IMS (2009) Global Learning Consortium. Guidelines for Developing Accessible Learning Applications, version 1.0. Retrieved July 14, 2009 from http://ncam.wgbh.org/salt/guidelines/

IMS (2003) Global Learning Consortium Learner Information PackageAccessibility for LIP Version 2.0 Final Specification: Information Model, XML Binding, Best Practice Guide, Conformance Specification, Use Cases, Examples, June 2003. Retrieved March 29, 2012 from http://www.imsglobal.org/accessibility

ISO IEC JTC1 SC36 WG7 Individualized Adaptability and Accessibility in E-learning, Education and Training. ISO/IEC 24751-1:2008

JISC (2009) *Personal Learning Environments*. Retrieved July 14, 2009 from http://www.jisc.ac.uk/index.cfm?name=cetis_ple

Kompen, T. R., Edirisingha, P. & Mobbs, R. (2008) "Building Web 2.0-based personal learning environments - a conceptual framework", EDEN conference, Pairs, October 2008, Retrieved March 29 2012 from http://hdl.handle.net/2381/4398

Lazarinis, F., Green S., & Pearson, E. (eds.), (2011), *Handbook of Research on E-learning Standards and Interoperability: Frameworks and Issues,* IGI Global, 2011

Leeder, D. (2009), INTERACTIVE e-learning development workshops. Proceedings of the IADIS International Conference Information Systems 2009, Barcelona, Spain.

Nevile, L., & Treviranus, J. (2006). Interoperability for Individual Learner Centred Accessibility for Web-based Educational. Systems. *Educational Technology & Society*, 9 (4), 215-227

Opera, (2010), *Dev.Opera 2008-2010 Widgets*, Retrieved January 19, 2011 from http://dev.opera.com/articles/widgets/

Pearson, E., Green, S. & Gkatzidou, S. (2009), "Responding to the challenge of providing learner-centred, accessible, personalized and flexible learning". *IEEE Conference on Advanced Learning Technologies (ICALT).* Riga, Latvia.

Sloan, D., Heath, A., Hamilton, F., Kelly, B., Petrie, H., & Phipps, L. (2006) "Contextual web accessibility - maximizing the benefit of accessibility guidelines", *ACM International Conference Proceeding, 134, W4A at WWW2006*, May 2006, Edinburgh, UK

Van Harmelen, M. (2006) "Personal Learning Environments", IEEE, Proceedings of the Sixth International Conference on Advanced Learning Technologies (ICALT'06), 815-816, 2006

W3C WAI, (2009) *Accessible Rich Internet Applications (WAI-ARIA) 1.0*, W3C Working Draft 24 February 2009, Retrieved July 14, 2009 from http://www.w3.org/TR/wai-aria/

W3C (2009), Authoring Tool Accessibility Guidelines (ATAG) 2.0, W3C Working Draft 21st May 2009, Retrieved March 29, 2012 from http://www.w3.org/TR/ATAG20/

W3C (2009) *Web Content Accessibility Guidelines (WCAG) 2.0*, W3C 11 December 2008, reformatted 3 March 2009, Retrieved March 29, 2012 from http://beta.w3.org/TR/2009/REC-WCAG20-20090303/

W3C (2011). *Widgets Packaging and XML Configuration Recommendtaion, September 2011,* Retrieved 29 March 2012 from http://www.w3.org/ TR/widgets/

W3C-WAI (2010), *WAI-ARIA overview: Retrieved January 21, 2011 from* http://www.w3.org/ WAI/intro/aria

Wookie, (2009). *Apache Wookie Incubator.*: Retrieved 19 January 2011 from http://incubator.apache.org/wookie/

Wenger, E. (1998). *Communities of Practice*. Cambridge: Cambridge University Press.