

Marcos S. Cáceres

Standardising Widgets

Improving various aspects of client-side web applications.

This is a confirmation document submitted in partial fulfilment of the requirements for Presentation of PhD Thesis by Published Papers.

This document is now stable and represents confirmation of PhD candidature as of the 29th August, 2007. No further amendments will be made to this document.

Abstract

This document argues that the continued proliferation of a class of software application known as *widgets* onto desktop computers and mobile phones has resulted in incompatibilities across most widget-related software. This document demonstrates how these incompatibilities currently affect distribution and deployment, accessibility, security, metadata, internationalisation and the device-independence of widgets. This document proposes standardisation as a possible solution to overcoming these incompatibilities.

The PhD study involves investigating technical aspects that constitute a widget, aspects related to the development and deployment of widgets on multiple devices, and aspects related to the applications on which widgets are hosted, known as *widget engines*. Outcomes from this PhD have contributed to an international effort to standardise widgets through an international consortium known as the *World Wide Web Consortium (W3C)*.

In addition, this confirmation document describes what methods will be employed to conduct the research, what publications will be produced, and how that knowledge will be disseminated within the two year timeframe remaining to complete this PhD. This confirmation document attempts to meet the requirements of Confirmation of Candidature as described in Section 12 of QUT's Manual of Policies and Procedures (MOPP) (QUT, 2006). The final form of this PhD will be a Presentation of PhD Thesis by Published Papers as specified in Section 14 of the MOPP.

Keywords: Widgets, widget engines, standardisation, W3C, standards research.

Table of Contents

| | |
|--|----|
| Abstract..... | 1 |
| Executive Summary..... | 4 |
| Confirmation Document Overview | 5 |
| Section 1: Proliferation of Widgets | 6 |
| The Year of the Widget? | 7 |
| Issues with Widgets: A Contextual Review. | 7 |
| Development..... | 8 |
| Distribution and Deployment (Packaging) | 9 |
| Internationalisation and Localisation..... | 13 |
| Accessibility..... | 15 |
| Security. | 15 |
| Metadata..... | 16 |
| Device Independence..... | 16 |
| Summary of the issues. | 17 |
| Overcoming the Issues. | 18 |
| The Thesis and What I’m Going to Do. | 19 |
| Working With the W3C to Standardise Widgets. | 20 |
| About the WAF-WG. | 20 |
| What is Standardisation? | 20 |
| The Standardisation Process..... | 21 |
| A Brief summary of Specification Levels of Maturity..... | 22 |
| The <i>Widgets 1.0 Requirements</i> document..... | 22 |
| The <i>Widgets 1.0 Specification</i> | 24 |
| The <i>Widgets 1.0 Test suite</i> | 24 |
| The <i>Widgets 1.0 Primer</i> | 25 |
| Being an Editor at the W3C..... | 25 |
| The Process in the ‘Real World’. | 25 |
| What the W3C Provides..... | 26 |
| Engaging the Relevant Communities. | 27 |
| What else I have been doing..... | 28 |
| Independent Publications. | 29 |
| Experimenting With Widgets | 29 |
| Summary of Section 1. | 31 |

| | |
|---|----|
| Section 2: Methods and Timeframe. | 33 |
| Selection Criteria for the Methods. | 35 |
| Specification Writing and Verification. | 35 |
| Meeting the ISO/IEC Methodological Criteria. | 37 |
| Specification-Based Software Testing. | 37 |
| Meeting the ISO/IEC Methodological Criteria. | 37 |
| Participatory Design. | 37 |
| Meeting the ISO/IEC Methodological Criteria. | 39 |
| Mixing the Contextual Review and Software Testing. | 39 |
| Software Development. | 40 |
| Timeframe. | 41 |
| Bibliography | 43 |
| Acknowledgements..... | 49 |

Executive Summary

The continued proliferation of a class of software application known as *widgets* onto desktop computers and mobile phones has resulted in significant incompatibilities across widget-related software. This PhD sets out to expose various issues that arise when one attempts to deploy a widget on multiple devices or software platforms, and/or in an internationalised context.

To overcome these issues, the objective is to directly contribute to an international effort to standardise widgets currently being undertaken by the *Web Application Formats Working Group* (WAF-WG) (Jackson, 2005), in which I am involved as the representative for the *Queensland University of Technology* (QUT)¹, within the *World Wide Web Consortium*² (W3C). I am contributing to this standardisation effort by collaboratively producing a number of related publications and allowing the research to feed back into the standardisation process.

Within the timeframe³ of two years from date of completion of this confirmation document, the WAF-WG hopes to have a number of widget-related specifications reach *Candidate Recommendation*⁴ status at the W3C; these specifications are described in detail later in this document. The *W3C Process Document*⁵ (Dubost, Rosenthal, Hazaël-Massieux, & Lofton, 2005) defines a Candidate Recommendation as “a document that W3C believes has been widely reviewed and satisfies the Working Group's technical requirements. W3C publishes a Candidate Recommendation to gather implementation experience.”

This PhD will be defended on the basis of the following three publications. To claim sole authorship, I will produce these publications independently of the standardisation process⁶. The following list represent working titles of the three publications; they are intended to give the reader an idea of the topics that will be discussed:

1. *Widgets and incompatibilities across widget engines: use cases and requirements*

¹ QUT has been a W3C Member since the 19 May 2005. See <http://www.w3.org/Consortium/Member/Testimonial/List>

² See <http://www.w3.org/Consortium/> for more information.

³ See also the Timeframe section of this document on page 26.

⁴ The ultimate goal for any W3C document that is on the standardisation track is to have the document reach W3C Recommendation status, which is, according to the W3C's Process Document (Jacobs, World Wide Web Consortium Process Document, 2005), “a specification or set of guidelines that, after extensive consensus-building, has received the endorsement of W3C Members and the Director. W3C recommends the wide deployment of its Recommendations... W3C Recommendations are similar to the standards published by other organisations.”

⁵ The W3C's Process Document outlines the policies and procedures of the W3C.

2. *Overcoming incompatibilities in widgets through standardization*
3. *Design and issues of the Widgets 1.0 Test Suite*

At the end of this three year PhD process, it is my intention to submit the above three publications, along with a summarising chapter, to QUT to meet the requirement of *PhD Theses by Published Papers* (QUT, 2006). It is my plan to submit these papers to either the international *X-Tech Conference* or the international *World Wide Web Conference* as they are the most highly regarded conferences in the field. Papers submitted to these conferences undergo a peer-review process and hence are valid for PhD submission under Section 14 of QUT's Manual of Policies and Procedures (QUT, 2006). Regardless of these publications, this confirmation document is about the standardization effort and the research that has gone into that process to date.

Confirmation Document Overview

Aside from this Executive Summary, this document is divided up into three sections and an Appendix. Section 1 provides a brief introduction into what widgets are and some of the market forces driving widgets' rise in popularity. Section 1 also identifies a set of issues with widgets and widget engines and makes an argument for the need to standardise widgets. Section 1 also discusses what outputs the PhD will have and how those outputs have started to be disseminated to various relevant communities. Section 2 discusses the methods and processes I will use to create various outputs, and present a timeframe that estimates how long it will take to complete all the outputs associated with the PhD.

Section 1: Proliferation of Widgets

Since 2003⁷, a relatively new kind of application has seen significant proliferation onto both the desktops of computers and, more recently, web-enabled portable devices like mobile phones. This kind of application is generally referred to by developers as a *widget engine*: a piece of software that is able to run other smaller applications known as *widgets* or *gadgets*. Jaokar and Fish (2006, p. 99) define a widget as “a downloadable, interactive software object that provides a single service such as a map, news feed etc”.



Figure 1. Yahoo!'s Weather Widget periodically downloads weather data from weather.com and displays it to the user based on their geographical location.

On the user's desktop, widgets have gradually taken the place of traditional single-purpose applications. Typical examples of widgets range from simple clocks, CPU gauges, post-it notes, games, battery-life indicators, to more sophisticated web-enabled widgets like weather forecasters (see Figure 1), news readers, email checkers, photo albums and currency converters (see Figure 2).



Figure 2. Yahoo's Flickr photo album Widget and Microsoft's Currency Converter Gadget, examples of widgets that make use of web services.

There are literally thousands of unique widgets available for download on the web⁸, which users generally collect to create personal widget inventories. These widget inventories provide users with access to online services that they use everyday. This means that, in a lot of instances, users don't need to open up a web browser to get the information that they want; an aspect of widgets that makes them particularly attractive on mobile devices⁹ (Jaokar & Fish, 2006, pp. 106-7), where the monetary cost of downloading web pages is currently an issue for many users (Goodwins, 2005).

⁷ The concept of 'widgets' is, of course, encountered in the computing literature much earlier than 2003 with the first known reference occurring in computer literature in (Swick & Ackerman, 1988) and being applied to user interface construction kits; However, it is generally accepted, and will be shown, that *widget engines* as discussed in this document emerged in 2003.

⁸ Each widget vendor provides virtual gallery of widgets created by their developer communities. See, for example, Yahoo! Widget Gallery at <http://widgets.yahoo.com/gallery/>.

⁹ Widgets don't need to download complete web pages; they can download optimised data as needed.

For developers, some widgets differ from traditional binary applications in that they are created using the same open technologies used to create web pages¹⁰. Widget engines mimic, in many ways, the behaviour of web browsers: an increasing number are actually built directly on top of web browsers so they are able to render web pages, while others incorporate web browser components¹¹. To developers, and vendors, this means that most widgets are significantly easier to build than applications developed with lower-level programming languages¹² (Barstow, 2007). Jaokar and fish (2006, p. 99) note that, “Widgets harmonise applications development across the Web and enable a wider application distribution model.”

The Year of the Widget?

With the inclusion of the *Sidebar* (Microsoft Inc., 2007) in Windows Vista and Apple’s MacOS X *DashBoard* (Apple Inc., 2007)¹³, widget engines are now a mainstream component of the two major consumer Operating Systems (OSs). Even alternative desktop OSs (eg. the many flavours of *Linux*) provide users with an increasing range of widget engines¹⁴. The continual growth in popularity of widgets has prompted traditional news media sources like *Newsweek* magazine to ask if 2007 would be “The year of the widget?” (Braiker, 2006). The marketing potential of widgets, particularly on mobile devices, has also been publicised in other mainstream media with *Business Week* citing an analyst as estimating that “in the U.S. marketing spending on mobile widgets will reach [US]\$500 million by 2010, up from about \$2 million [in March 2007]” (Kharif, 2007). Assumedly, recognising the potential for revenue, large players in the Information and Communication Technology (ICT) industry, like Nokia and Apple, announced early in 2007 that they will independently release widget engines for some of their mobile devices (Nokia Inc, 2007); (Apple Inc, 2007).

Issues with Widgets: A Contextual Review

Amidst the popular rise of widgets and widget engines lay a number of issues for users, developers, current vendors and new vendors wanting to enter the widget market. In this section I present these issues as they relate to:

¹⁰ Namely using HTML (Raggett, Le Hors, & Jacobs, 1999) or XML (Bray, Paoli, Sperberg-McQueen, Maler, & François, 2006), CSS (Bos, Çelik, Hickson, & Wium Lie, 2006) and/or ECMAScript (ECMA, 1999), as well as the popular image formats PNG, JPG, and GIF, and the Zip (PKWare Inc, 2006) compression format. Widgets also make extensive use of the Ajax (Garrett, 2005) development approach. Some widget engines, particularly those on Linux, leverage the Python for both runtime and scripting. Some engines also have support for playing MP3 files and other audio formats. Other engines have been exclusively developed to only support Adobe Flash.

¹¹ For example, *Yahoo!’s Widget Engine* makes use of the *SpiderMonkey* JavaScript interpreter, built by the Mozilla Foundation for the Gecko-based browsers like *Firefox*.

¹² like Java and C#.

¹³ Dashboard was actually released in 2004, with Mac OS X 10.4.

¹⁴ Such as *gDesklets* (gDesklets, 2007) and *aDesklets* (adesklets, 2007)

- *Development.*
- *Distribution and Deployment.*
- *Accessibility.*
- *Security.*
- *Metadata.*
- *Internationalisation and localisation.*
- *Device-independence.*

A word of warning to the reader: the following sections go into quite a bit of technical detail about the inconsistencies and incompatibilities of widgets and widget engines. Although I have made an attempt to make the following discussion as non-technical as possible, it still contains quite a few necessary technical details. This section is also an ongoing work in progress. If one is not technically inclined, or would just prefer to skip ahead, I provide a simple summary of the issues I have identified on page 17 under the heading *Summary of the issues*.

Development

Development refers to how widgets are made or otherwise *coded*¹⁵. As mentioned earlier, most widgets are built using the same technologies used to build web pages: namely, generally by combining HTML or XML with CSS, images and sounds for creating the user interface; and ECMAScript for enabling interactivity, event handling, and network requests¹⁶. In general, widgets that access web services make extensive use of the now well-established *Ajax* development approach (Garrett, 2005); (Jaokar & Fish, 2006, pp. 96-9).

Issues related to development

The main issue with widget development is the inconsistencies in programmatic control provided to the developer by the widget engine. Although there is consistency across what kind of functionality is afforded across widget engines (eg. being able to access resources on the web, and being able to read, write, and execute files), the way that functionality is provided to developers is inconsistent across most widget engines¹⁷. This makes it extremely difficult for developers to create a widget that can be run on more than one widget engine. This is an issue because it results in vendor lock-in for

¹⁵ Coded here means using both mark-up languages and scripting languages, like HTML+CSS and JavaScript.

¹⁶ Through XMLHttpRequest or similar network request brokering object. Note that some widget engines also leverage Adobe Flash.

¹⁷ That is, the *Application Programming Interfaces* (APIs) are different across all widget engines! Even though they mostly do the same things.

developers and for those users wanting to use the widget on a different widget engine or platform other than the one the widget was designed for¹⁸.

Distribution and Deployment (Packaging)

Distribution and deployment refers to getting a widget from the web to run on a user's device as easily as possible. Users generally get their widgets directly from widget vendors (eg. apple.com) who often have dedicated online galleries where users can search for widgets and where developers can submit or update widgets they have created (see Figure 3¹⁹ for example). Users generally browse these galleries through a web browser or sometimes even through a dedicated widget. Once a user finds a widget they want, they download it from the web using the *Hypertext Transfer Protocol* (HTTP) (Fielding, et al., 1999).

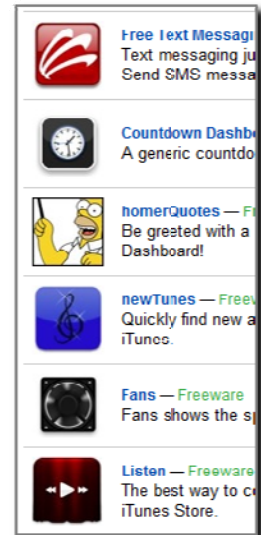


Figure 3. Apple's Widgets Gallery Website , where users can go and get widgets for Dashboard.

HTTP is the protocol of the Web, enabling users to access hyperlinked resources on the internet using the Uniform Resource Identifier (URI)²⁰ addressing scheme (Berners-Lee, Fielding, & Masinter, 2005) (eg. '<http://example.org/homerQuotes.widget>'). Concisely, HTTP defines the rules and methods that allow a client's device to negotiate access to resources with a web server; URI is how a user's web browser addresses the file it wants to download. In the case of widgets, this is a single file like 'homerQuotes.widget'.

The *resources* of the widget (the images, sounds, ECMAScript, HTML files) are bundled into some *packaging* format. Packaging refers to encapsulating all the necessary resources and metadata required by the widget into a single file for the purpose of distribution and deployment. The *de facto* standard²¹ for packaging widgets is the Zip file format (PKWare Inc, 2006), but with vendors requesting that their developers use a vendor specified file extension (ie. not '.zip', but '.widget', or '.gadget', etc).

¹⁸ An example of this inconvenience would similar to only being able to play a music file in one application, and not being able to transfer it to another device, like an iPod.

¹⁹ Screen shot from <http://www.apple.com/downloads/dashboard/>

²⁰ URIs are a subset of Internationalised Resource Identifiers (IRIs) (Duerst & Suignard, 2005). IRIs allow international characters to be used with Uniform Resource Locators (URLs).

²¹ A *de facto* standard is a technology that is used by industry without any formal specification.

Once a widget has been packaged for distribution, it is put onto a web server and must be served with an appropriate *Media Type*. The Media Type refers to the kind of data contained in the resource being served over HTTP and it is identified by a *HTTP header*. A HTTP header is a simple text string that describes some details about a request or response to/from a web server.

Some choices in Media Types include `audio`, `video`, `text`, and `application`²². When a resource is served over HTTP, it will usually be accompanied by a media type that takes the form `'content-type: content-type/sub-type'`, the sub-type is usually the file format. For example, widgets served for Opera's widget engine are served as `'application/x-opera-widgets'`, where `'application'` is the content-type, and `'x-opera-widgets'` is the sub-type.

If the appropriate widget engine is installed on the user's computer, and the widget engine has correctly registered a media type and/or file extension with the file system to be associated with the widget engine, the web browser should automatically associate the downloaded resource with the widget engine. This can be seen in Figure 4, where the *Firefox*²³ web browser has correctly identified the resource `'CI-Alerts.widget'` as a `'Yahoo! Widget'` that should be opened with the `'Yahoo! Widget Engine'`²⁴. The desired outcome is that the user need only click on the OK button and the browser should automatically attempt to instantiate the widget on the appropriate widget engine.

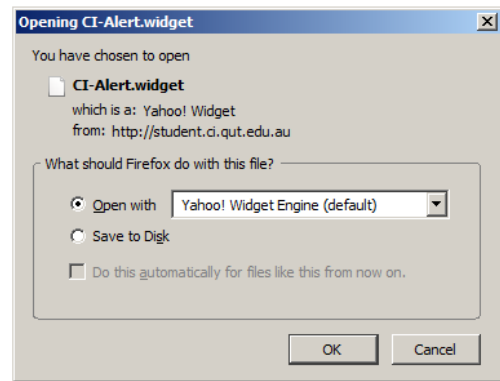


Figure 4. Firefox's save dialog associating a media type and file extension (.widget) to the correct widget engine (Yahoo! Widget Engine).

Issues with packaging widgets for distribution and deployment

There are four issues with regards to packaging as described above:

1. There is no consistent file extension for widgets; each vendor uses their own unique file extension.
2. Nowhere is it defined exactly which parts of the Zip specification should be supported by widgets. Zip is a large and complex specification, supporting multiple compression algorithms and features. If a widget is packaged using the wrong compression algorithm, it might not be compatible with another widget engine or device. Although the issue of

²² see <http://www.isi.edu/in-notes/rfc2045.txt> for a complete list of options, or <http://www.iana.org/assignments/media-types/> for a complete list of officially registered media types.

²³ <http://www.mozilla.com/en-US/firefox/>

²⁴ Please note that in this instance the association with the widget engine is done through the file extension (which are the characters after the `'.'` in the file name) rather than by the media type.

- compression is not currently a major issue, it has the potential to become one as widgets become more prevalent on mobile devices and computers start to work natively in 64-bit.
3. There is no standardised Media Type, which results in widgets served over HTTP being associated with only one kind of widget engine. One media type per widget engine results in vendor lock-in.
 4. Unlike most other reputable binary executable software distributed on the web, most widget engines lack support for a standard way to check if a widget has been *digitally signed*.

A digital signature is “a number dependent on some secret known only to the signer (the signer’s private key), and, additionally, on the contents of the [package] being signed” (Johnson, Menezes, & Vanstone, 2001). The only way to validate a digital signature is to use a matching public key in a process known as *asymmetric* or *public key cryptography* (Mollins, 2003, pp. 53-78). The *Yahoo! Widget Engine* is currently the only widget engine that supports validating widget packages by means of public key cryptography²⁵. However, the *Yahoo! Widget Engine*’s means of handling the signing of widgets and digital signature validation is non-standard (Voas, 2006).

Digital Certificates and distribution and deployment

As just mentioned, it’s common practice in the software industry to digitally sign application packages for distribution. However, digitally signing widgets is not a common or much supported feature in the widget space. Because there is currently no standard way to digitally sign and verify widgets, widgets are susceptible to being intercepted and maliciously modified when transferred over HTTP²⁶. For this reason, some widget engines will not automatically instantiate widgets on request unless the widget has been digitally signed by a *trusted partner*. A trusted partner, is a *digital certificate* (a very large encrypted number, and some additional identifying information (Gentry, 2003, p. 641)) included with the default widget engine installation. Simply, if the digital signature on the widget matches the trusted partner, then the widget will be instantiated by the widget engine. If the signature does not match, the user will usually be warned and the widget will normally be discarded.

When a widget is not digitally signed, or otherwise not recognised as coming from a trusted partner, then it is customary for the widget engine to warn the user of the potential dangers of running the widget. Two examples of this can be seen in Figure 5. On the left, *Yahoo! Widget Engine* warns the

²⁵ Note however, most widget engines support encrypted network communications using public key cryptography. However, this is different to digitally signing a widget.

²⁶ This is known as a *man in the middle attack* (Mel & Baker, 2001).

user that they should only instantiate the widget if they “know and trust the source”. And on the right, *Opera’s Widget Engine* warns the user of the potential dangers of running a widget by saying that “Opera may become unstable and insecure as a result”.



Figure 5. Security warnings, Yahoo! Widget Engine on the left, Opera Widgets on the right.

Returning to public key cryptography; in cryptography (Mel & Baker, 2001), the advantages of using digital signatures and certificates are categorised by:

- *Authenticity.*
- *Data Integrity.*
- *Non-repudiation.*

Authenticity: The digital certificate is pivotal in providing a means to verify that a widget package was really made by the author who claims to have created it. Authenticating a widget is possible because the widget engine is able to verify the signer of a widget against its list of pre-packaged digital certificates. However, in the case of the *Yahoo!’s Widget Engine*, the only trusted partner is *Yahoo! Inc*, so no other widget developers are considered trusted partners. Widget engines verify the authenticity of a package by decrypting an encrypted hash of the content of the package, known as a *digest*, using a public key; theoretically, only a trusted partner’s public key should be able to decrypt the digest.

Data integrity: Checking the integrity of the package means the widget engine is checking if any of the data inside the package has been changed after the package was signed. A widget engine can then check if the data of a package is corrupt by checking a mathematic sum (known as the *hash*) of all the bytes in the package against the digital signature. If the hash value do not match the digest, then it is likely that the file is either corrupt or someone has tampered with the contents of the package after it was digitally signed.

Non-repudiation: Digital certificates make it extremely difficult for authors to deny, or *repute*, having signed a package; hence the term *non-repudiation*. Non-repudiation works because a digital

signature makes it difficult for a signer to deny having signed the contents of a package as only the signer should have access to the private key used to create the digital signature. From a user's perspective, if their computer is maliciously damaged by a digitally signed widget, they are able to trace the signature back to the person or organisation that signed the package and, if necessary, take legal action.

Issue with digital certificates and distribution and deployment

Digital certificates and signatures are not themselves an issue. The issue is that, given all the advantages of using asymmetric cryptography, digital signatures and digital certificates should be supported *by standard* on all widget engines. Currently they are not. In addition, users should also be provided a secure means to add digital certificates to the widget engines to increase the number of trusted partners. The ability to add trusted partners is functionality available on most common web browsers today.

Internationalisation and Localisation

Internationalisation and localisation refers to the ability for widgets to be used in languages and cultures outside the English-speaking world. Although most widgets have support for displaying internationalised content (ie. any language other than English), the way that content is localised is inconsistent across all widget engines.

Technically speaking, Internationalisation and localisation refers to both mechanisms and guidelines that vendors put in place to facilitate developing widgets for cross-cultural contexts. Savourel (2001) defines *internationalisation* as "the process of developing a product in such a way that it works with data in a different language and can be adapted to various target markets without engineering changes." Savourel also defines *localisation* as "the subsequent process of translating and adapting a product to a given market's cultural conventions." Widget vendors often provide guidelines that developers should follow in order to effectively internationalise and localise their widgets. Widget vendors also build automated mechanisms into their widget engines to, for instance, automatically select the appropriate localised content for a widget based on a system's regional and language options, or *locale information*.

From an Australian perspective, examples of localised content include spelling the word "centre" instead of "center", or using kilograms instead of pounds, or formatting short dates in the 'day/month/year' format instead the 'month/day/year' format as is the custom in the United States. One can imagine that using the American date format in Australia would cause significant problems, and vice versa.

In widget development, localisation of content is commonly achieved in three ways; none of these methods are supported across all widget engines:

1. Place localised content into specifically named folders and let the widget engine automatically select the localised content based on locale information retrieved from the operating system; as is the case with Vista's *Sidebar* gadgets, Apple's *Dashboard* widgets, and *Google Gadgets*.
2. Placing localised text content inside a text file, which is in turn placed inside folders that have localised names (eg. `"/en-au/"` for "English Australian"). For example, in an Australian context, *Yahoo! Widget Engine* would automatically load `'en-au/localised.strings'` file from the appropriate folder and makes it available to the developer via a scripting interface.
3. Create a new widget from scratch for each culture one wishes to target, as is the case with *Opera Widgets*. On the one hand, having to localise content cultural context will be more laborious than having the widget engine automatically select the right content for the widget. And, by Savourel's definition, having to do manual labour to achieve localisation is clearly not internationalisation. On the other hand, one minor drawback of having lots of localised content inside the widget is that it increases the size of the package which can potentially cause concern to users, particularly in the mobile market segments where data download is charged per kilobyte (Goodwins, 2005).

Issues relating to internationalisation

Most widget vendors have documented methods that make it possible for developers to have their widgets localised with minimal or no change to the widget itself. In this document, we refer to these methods as *localisation strategies*. However, the way vendors ask users to name directories for localisation are inconsistent. The most common²⁷ localisation strategy used by widget engines is to use the case-insensitive 'language-region' pattern that combines *Language Codes* (ISO, 1998), a dash, and *country codes* (ISO, 1997) into a *language tag*. For example, the language tag for the English language in the United States is "en-us" and the language tag for Afrikaans, South Africa, is "af-za"²⁸. How languages and regions may be combined is standardised by the RFC1766 specification (Alvestrand, 1995). If the developer chooses to only target one specific language, they would declare just the language code (eg. "en"). Specifying just one language code serves as a fallback mechanism for when attempts to localise fail. The RFC1776 specification states that,

²⁷ that is, the strategy supported by Vista Sidebar, Yahoo!'s Widget Engine, and Dashboard.

²⁸ Note that patterns are not limited to four letter codes. Subregions can also be specified for particular dialects (eg. "x-mn-Trad" for Traditional Mongolian). Language codes are generally associated to character encoding schemes that allow those characters to be rendered properly on computers.

The language tag is composed of 1 or more parts: A primary language tag and a (possibly empty) series of subtags.

The syntax of this tag in RFC-822 EBNF is:

```
Language-Tag = Primary-tag *( "-" Subtag )
Primary-tag = 1*8ALPHA
Subtag = 1*8ALPHA
```

Whitespace is not allowed within the tag.

In localising content, both *Microsoft's Sidebar* and *Apple's Dashboard* follow the four-letter code convention separated by a hyphen. *Yahoo! Widget Engine*, however, breaks convention by using an underscore character ("_")²⁹. *Google Gadgets*, breaks convention even further by requiring the use of both Microsoft's Language Codes³⁰ for the default English container and language tags as defined in RFC1766 but modified to use an underscore ("_") as the delimiter between the primary-tag and the sub-tags.

The *non-standard* use of language codes is the major issue for achieving consistent internationalisation and localisation across widget engines.

Accessibility

Accessibility refers to how everyday people, and those with special needs, can access the content and use the interactive elements of widgets. As stated earlier, most widgets make use of technologies used to build web pages so those widgets can theoretically be made to be fairly accessible by applying, for example, the relevant sections of the *Web Content Accessibility Guidelines* (Chisholm, Vanderheiden, & Jacobs, 1999).

Issues relating to Accessibility

For widget engines that don't make use of the standards used to create web documents, like the *Yahoo! Widget Engine*, nowhere is it specified exactly how those widgets can be made accessible.

Security

Security refers to how users can be kept safe, and how they keep themselves safe, from potentially malicious widgets. The scope of security in widgets deals with mainly with *access-control*. From a user's standpoint, access-control generally refers to mechanisms that declare what a widget can and can't access on a user's device or over a network: basically what the widget is allowed and not allowed to do.

²⁹ or 005F in *Unicode*.

³⁰ <http://www.microsoft.com/globaldev/reference/win2k/setup/lcid.msp>

Currently, some widget engines allow widgets to do just about anything on a users computer: they can read, write, modify, and/or delete files, automatically upload files, automatically download and run files, run local executables, "mash-up" data from multiple different sources; all without the user having any idea of what is happening. The ability to do any of the things just listed is strictly forbidden on web browsers (Huseby, 2004). The consequence of widget engines having such a relaxed security model has been generally positive with is very useful and powerful widget being created; However, but at the same time, it has created an environment where users are left extremely vulnerable to malicious attacks.

Issues relating to security

There is currently *no standard* security model for widgets and the issues of widget security are not very well understood. This is evidenced by the lack of documentation about security in current widget engine documentation.

Metadata

Metadata refers to how authors store information about their widget inside the widget, and how that data is made accessible to people or machines.

Issues relating to metadata

Although there is consistency about the format in which the metadata is stored³¹, there is currently inconsistency across all widget engines about exactly what information an author should be allowed to record and how that information should be structured.

There is also *no standardised* way to identify the version of a widget; hence it becomes difficult for developers to redeploy a widget in the case of an error or in the case of the emergence of a security issue.

Device Independence

Device independence refers to how widgets can be run on devices other than desktop computers (eg. mobile phones, specialised gadgets, PDAs).

Issues Relating to Device Independence

At the time of writing, most widget engines for mobile devices are in their infancy and issues relating to the device independence of widgets are only just starting to emerge. They include issues like how to:

- Automatically cope with differences in screen resolutions, aspect ratios, Dots Per Inch (DPI) when migrating a widget from one device to another.

³¹ Generally using XML.

- Programmatically accessing device specific capabilities, for example, the camera, SMS, and contacts lists on mobile phones.
- Reducing widgets consumption of battery life, particularly when the widget makes use of JavaScript.
- Dealing with inconsistent or 'spotty' network coverage.
- Persistent offline storage.

Many of these device independence issues listed above are beyond the scope of this PhD. However, a general set of guidelines specific for widgets may assist in getting widgets to become device independent.

Summary of the issues

The list below represents the core issues that I have identified with widgets and widget engines resulting from my Contextual Review of widgets and widget engines:

- Development:
 - Inconsistent programming interfaces and supported features.
 - Inconsistent mark-up languages.
- Distribution and Deployment:
 - Inconsistent file extensions.
 - Inconsistent Media Types.
 - Poorly defined packaging format.
 - Lack of support of automatically finding and updating widgets.
 - No standard digital signature format.
- Accessibility:
 - No baseline for accessibility.
- Security:
 - Inconsistent security model.
- Metadata:
 - Lack of consistent fields or structure.
- Internationalisation and localisation:
 - Inconsistent or non-existent support for localising content.
- Device-independence :
 - Lack of guidelines for how to make widgets device independence.

Together these issues represent the *fragmentation* of the widget market. This PhD sets out to investigate and attempt to solve most of the issues listed above in a way that is as interoperable as possible across market-leading widget engines³². This list is by no means complete, there are many other issues starting to emerge as widgets become more prevalent, and, as previously mentioned, as widgets move into the mobile space.

Overcoming the Issues

The list of issues described in the previous section represents a complex problem and writing a PhD about them will have limited impact unless vendors are also willing to be involved in fixing the problems. What is required is an open forum where users, developers, researchers, and vendors can work together to resolve these issues (Krechmer, 2006). In other words, a forum where by widgets can be *standardised*.

In standardisation literature (Jakobs, 2006), resolving issues to make various software applications work together is commonly referred to as a *matching problem*: “a problem of determining one or more features of different interrelated entities in a way that they harmonise with one another, or of determining one or more features of an entity because of its relation(s) with one or more other entities” (Vries, 2006, p. 5).

One of the major side-effects of the lack of standardisation is that widgets from one widget engine cannot be run on another widget engine; that is, implementations are unable to *interoperate*. For developers, this means having to recode their widgets to have them work on other widget engines. For users, this has resulted in either vendor lock-in or having to run multiple widget engines simultaneously to use the widgets they like. These are common side-effects when there is no standardisation amongst competing technological products³³ (Besen & Farrell, 1994). Besen and Farrell explain that “agreeing on a standard may eliminate competition between technologies, but it does not eliminate competition altogether. Instead, it channels it into different and (to economists) more conventional dimensions, such as price, service, and product” (Besen & Farrell, 1994, pp. 119-120).

³² Vista Sidebar, Yahoo!’s Widget Engine, Opera Widgets, and Dashboard.

³³ This is not to imply that standardisation always results in positive economic outcomes. Various factors influence the need for standardisation and a company’s choice to adopt a standard or participate in a standardisation process.

The Thesis and What I'm Going to Do

Firstly, this *thesis*³⁴ will show that the lack of standardisation has led to fragmentation of the widget space leading to extensive incompatibility across widget engines. That is, widgets developed for one widget engine cannot generally be run on another widget engine without significant modification to either the widget or the widget engine³⁵. More significantly, as I have demonstrated above, the current market-driven growth of the widget space has resulted in implementations that sometimes ignore or do not fully resolve aspects relating to development, distribution and deployment, internationalisation and localisation, metadata, accessibility, security, and device independence. Addressing these issues through standardisation has potential economic benefits: “most analysts believe that price competition is more intense when vendor’s products are compatible, both because product variety is reduced and because users are less likely to be locked-in to a single firm’s product” (Besen & Farrell, 1994)³⁶.

Secondly, this thesis will aim at contributing to the current international effort to standardise widgets through the World Wide Web Consortium (W3C) by:

1. Providing research evidence of fragmentation and issues relating to development, distribution and deployment, internationalisation and localisation, metadata, accessibility, security, and device independence of widgets and widget engines.
2. Suggest possible ways to reduce fragmentation and to resolve as many of the issues summarised on page 17 of this document as possible.
3. Collaborating to publish a number of specifications related to standardising widgets, which, if adopted by participating vendors, may reduce the issues of fragmentation.

In summary, for the widget market, standardisation has some of the following potential benefits:

- More accessible and secure widgets.
- Users having to install fewer widget engines on their computers or mobile devices, thus reducing vendor lock-in and user inconvenience.
- Simplifying the process of entering the market for new vendors by providing them with a complete specification from which to build a product.

³⁴ The word *thesis* used here in its traditional sense to mean “argument and not simply the end product of the research processes manifested in a classical bound tome” (Gray & Malins, 2004, p. 165).

³⁵ Tool like Netvibes’ *Universal Widget API* (NetVibes, 2007) have limited success in overcoming this problem. Although it claims to be “universal”, at the time of writing the API only supports a small number of widget engines.

³⁶ However, Besen & Farrell also state that economic theory is not conclusive.

- A larger widget market with widgets that are built to conform to standard.

Working With the W3C to Standardise Widgets

As stated previously, overcoming the issues that currently fragment the widget market without the collaboration of widget engine vendors would be tremendously difficult, if not totally futile. The only feasible solution for overcoming these issues is to work together with vendors, developers, users within the context of a standardisation body like the W3C.

The W3C is recognised as a standards organisation of World Wide Web technologies (U.S. Dep't of Justice & Fed., 2007); (Sherif, 2006, p. 197): some of the most widely implemented and used electronic publishing standards in the world, such as HTML, CSS, and XML, were all standardised within the W3C³⁷.

Standardisation efforts within the W3C are undertaken by *working groups*. A Working Group is a collection of people from various different organisations that work together to solve one or more matching problems. The working group that has chosen to attempt to standardise widgets is known as the *Web Application Formats Working Group* (WAF-WG).

About the WAF-WG

The WAF-WG³⁸ started in November 2005,

“...to develop specifications that enable improved client-side application development on the Web...The target platforms for this Working Group includes desktop and mobile browsers as well as many specialty, browser-like environments that use Web client technologies. The goal is to promote universal access both for users and devices, including those with special needs.” (Jackson, 2005)

Some organisations that have become members of the WAF-WG, and have direct involvement in the widget market, include: Microsoft, Apple, Nokia, Google, Opera, Vodafone, AOL, the Mozilla Foundation, and France Telecom³⁹.

What is Standardisation?

As a point of entry, one can consider standardisation as a process whereby competing entities and other interested parties collaborate on the creation and ratification of a *standard* that defines how products are supposed to interact in the form of a *specification* (Vries, 2006, pp. 9-13); (Bunsson & Jacobsson, 2000, pp. 1-15); (Hickson, 2006). Although many definitions of a standard exist (Vries,

³⁷ See (Berners-Lee & Fischetti, 2000, pp. 93,181,129) for history of each standard at the W3C.

³⁸ WAF-WG's home page: <http://www.w3.org/2006/appformats/>

³⁹ Although there is no commitment from any company to implement the specification, Microsoft, Apple, Nokia, Google, France Telecom, and Opera are all mayor players in the widget market.

2006, pp. 3-5), in this document I will be using the formal definition put forward by the International Standards Organisation (ISO) and the International Electronic Commission (IEC):

“[A] document, established by consensus and approved by a recognised body, that provides, for common and repeated use, rules, guidelines or characteristics for activities or their results, aimed at the achievement of the optimum degree of order in a given context. Note – Standards should be based on the consolidated results of science, technology and experience, and aimed at the promotion of optimum community benefit.” (ISO/IEC, 2004) in (Vries, 2006, p. 4).

The W3C defines a specification as a document “that prescribes requirements to be fulfilled by a product, process, or service”⁴⁰ (QA Framework: Specification Guidelines, 2005) . Poston explains that a specification “states or pictures how software is expected to behave and describes operational characteristics (performance, reliability, and so on) of the software.”

The *standardisation process* of widgets involves extensive collaboration and consultation with vendors and the development community with the ultimate aim of creating a number of publications, each which undergo extensive public and peer review (Henderson, 2005).

The Standardisation Process

“It is well known that the production of technical specifications intertwines technological considerations with business, social, interpersonal, political, and ideological aspects.” (Sherif, 2006, p. 129). At the W3C, the process of standardisation is largely mandated by the *W3C’s Quality Assurance Framework*⁴¹ and *Process Document* (Jacobs, 2005) and includes the production of:

- The *Widgets 1.0 Requirements* (Caceres, 2007).
- The *Widgets 1.0 Specification* (Caceres & Kesteren, 2006).
- The *Widgets 1.0 test suite* (*forthcoming*).
- The *Widgets 1.0 primer* (*forthcoming*).

Below I describe what the purpose of each of the above publications, provide historical background, and detail the *level of maturity* of these publications.

⁴⁰ Based on ISO’s formal definition of a specification.

⁴¹ See <http://www.w3.org/QA/Library/> for all the official and unofficial documents that make up the framework.

A Brief summary of Specification Levels of Maturity

The W3C's *Process Document* (Jacobs, 2005), outlines four *levels of maturity* that a document must go through before being ratified as a standard (at the W3C, standards are referred to as *Recommendations*). These publications go through an iterative peer-review process:

1. **Working Draft (WD):** “a document that W3C has published for review by the community, including W3C Members, the public, and other technical organisations” (Jacobs, 2005).
2. **Candidate Recommendation (CR):** “a document that W3C believes has been widely reviewed and satisfies the Working Group's technical requirements. W3C publishes a Candidate Recommendation to gather implementation experience” (Jacobs, 2005).
3. **Proposed Recommendation (PR):** “a mature technical report that, after wide review for technical soundness and implementability, W3C has sent to the W3C Advisory Committee for final endorsement” (Jacobs, 2005).
4. **Recommendation (REC):** “a specification or set of guidelines that, after extensive consensus-building, has received the endorsement of W3C Members and the Director. W3C recommends the wide deployment of its Recommendations. **Note:** W3C Recommendations are similar to the standards published by other organisations” (Jacobs, 2005).

On average, it takes three to four years for a specification to become a standard (Sherif, 2006, p. 136) and sometimes longer to be successfully implemented. For this reason, and as stated at the beginning of this document, I only expect to be involved with the listed publications to the level of CR.

The Widgets 1.0 Requirements document

The *Widgets 1.0 Requirements* document enumerates the requirements and *design goals* that the *Widgets 1.0 Specification* needs to address, or otherwise conform to, before the community who is working on the standardisation process will allow it to proceed to become a *Candidate Recommendation*. The design goals are generally qualitative high-level goals. The goals include, for example, making sure that:

- products⁴² produced from the *Widgets 1.0 Specification* will be usable by people with special needs.
- The *Widgets 1.0 Specification* is not bias towards any one vendor.
- The *Widgets 1.0 Specification* remains as interoperable as possible with existing software implementations and other standards.

⁴² For example, a *widget* can be considered a *product* of the specification.

The *Widgets 1.0 Requirements* document came into existence after discussions in the working group about the potential for widgets to be standardised⁴³. Motivated by the discussions, I started investigating how widgets work and identified some of the differences across widget engines. After a period of researching, I bundled all of my findings into a paper and submitted it to the WAF-WG (Caceres, Packaging Web Applications: Requirements, 2006). After a month of internal member review, the document was renamed *Web Applications Packaging Format Requirements*⁴⁴ and published to the public as a WD on August 21st, 2006 (Caceres, Web Applications Packaging Format Requirements, 2006). After further public and internal review, the publication was again renamed⁴⁵ to *Client-Side Web Applications (Widgets) Requirements*, and republished as the second public WD on the 9th of November 2006. The document then underwent another three months public and internal review before being renamed, yet again⁴⁶, to *Widgets 1.0 Requirements* and republished at the as the third public WD on the 9 February 2007. Further publications have postponed pending the successful completion of this confirmation document. The WAF-WG has scheduled the Fourth Public Working Draft of the *Widgets 1.0 Requirements* to be republished by July, 2007⁴⁷.

The *Widgets 1.0 Requirements* represent an attempt to capture what vendors and the development community identify as the aspects needed to be standardised by the *Widgets 1.0 Specification*, while balancing those requirements with the goals of the standardisation body⁴⁸. For the W3C, the goals are to produce specifications whose products are “available to all people, whatever their hardware, software, network infrastructure, native language, culture, geographical location, or physical or mental ability” (Jacobs, About W3C: Goals, 2007). So, the specification must attempt to address all the requirements outlined in the *Widgets 1.0 Requirements*, particularly those requirements that are marked with the special keywords “must” and “should”⁴⁹, as well as attempt to meet the goals laid out by the W3C. The *Widgets 1.0 Requirements* is generally targeted at vendors, developers, the public, and those working group members that will edit the specification.

⁴³ The discussions were mainly driven by Opera Software; whose CTO, according to (Sivonen, 2006), had mentioned at the international XTech 2006 Conferences that widget packaging was “ripe” for standardisation.

⁴⁴ The working group initially felt that using the word ‘Widgets’ might make people think that the document was not vendor neutral... particularly in the eyes of Microsoft’s *gadgets* and AOL’s *modules*.

⁴⁵ The working group felt that the original title was too ambiguous, so we decided to put in the word “widgets” in there.

⁴⁶ Made sense to the WAF-WG to rename it to *Widgets 1.0 Requirements* as it is only the *Widgets 1.0 Specification* that would need to address the requirements. Anne van Kesteren also argued that it made good marketing sense, which turned out to be true as the public response this publication received much more attention than the other two previous publications.

⁴⁷ I have included the forthcoming Fourth Public Working Draft of the *Widgets 1.0 Requirements* as an Appendix to this document⁴⁷, see page 34.

⁴⁸ See (Jacobs, About W3C: Goals, 2007) for the goals of the W3C.

⁴⁹ See (Bradner, 1997) for how these keywords are used in specifications.

The Widgets 1.0 Specification

The *Widgets 1.0 Specification* attempts to address the requirements set out in the *Widgets 1.0 Requirements*. As stated earlier, Poston (1996, p. 9) defines a specification as a document that “states or pictures how software is expected to behave and describes operational characteristics (performance, reliability, and so on) of the software. A specification serves as a reference or base to test against.” The *Widgets 1.0 Specification* will define how widgets are to be scripted, digitally signed, secured, packaged and deployed in a way that is accessible and device independent.

The *Widgets 1.0 Specification* was originally written by developers of *Opera Software* and by the WAF-WG on the October, 2006⁵⁰. After submission, it went through a minor internal (WAF-WG) review process before it was published as a WD on November 9th, 2006. The specification is partly based on Opera’s own *Widget Information File Syntax* specification (Opera Software, 2006), which describes how authors structure metadata for widgets that run on Opera’s widget engine. The Working Group agreed that specification would be co-edited by Anne Van Kesteren, of Opera Software, and I. I describe the role of being an editor at the W3C on page 25.

Even though specifications are written in English prose⁵¹, they are generally targeted at implementers and so they are sometimes illegible to the general public. This is because, in the words of Stock and Carrington (1996, p. 777), a specification “precisely defines fundamental aspects of the software, while more detailed and structure information is omitted”; thus, a specification will generally require the use of technical language to precisely describe various computational process and operational characteristics⁵². It is exactly the computational process and operational characteristics that vendors must be able to test through the *Widgets 1.0 Test Suite*.

The Widgets 1.0 Test suite

A *test suite* is a finite set of test cases that vendor’s can use to check their widget engine’s level of *conformance* to the specification (Stocks & Carrington, 1996) . At the W3C, a test suite is generally expected to take the form of “a set of documents and tools providing tool developers with an objective methodology to verify the level of conformance of an implementation for a given

⁵⁰ The submitted draft is available from <http://dev.w3.org/cvsweb/~checkout~/2006/waf/widgets/Overview.src.html?rev=1.1>

⁵¹ Not all specifications are written in English prose. Sometimes formal computational languages, like Z and UML, are used to create specifications. It is argued that formal languages are better for capturing specifications as they are less prone to ambiguities inherent in natural languages (Stocks & Carrington, 1996). However, specifications written in formal languages are not accessible to anyone not formally trained in those languages. The W3C does not mandate the use of any formal language; normative specifications are written in US-English and sometimes translated to other languages like Spanish.

⁵² This is not to say that specifications should not be written in a way that is inaccessible to the lay reader. However, the lay reader is not the intended audience and but the prose must be accessible to implementers (usually very experienced software engineers used to reading technical specifications).

standard” (Dubost, 2002). Rosenthal et al (1997) explain that the purpose of conformance testing is to “tests the functionality and correctness of a given implementation against a publicly available standard, to produce tests for as many of the requirements of the standard as feasible and then use the test to find errors in the implementation.” In the case of widgets, only once an implementer passes all necessary tests in the *Widgets 1.0 Test Suite* can they claim conformance to the *Widgets 1.0 Specification*. Creating the test suite will involve creating simple widgets that vendors can access over the web to rapidly test their conformance according to the specification.

Work on the *Widgets 1.0 Test Suite* is currently scheduled to begin after the next publication of the *Widgets 1.0 Specification* in mid July, 2007.

The Widgets 1.0 Primer

The WAF-WG also intends to produce some instructional tutorials that demonstrate usage scenarios that should be considered best-practise. This material is usually referred to as a *primer*. A primer is targeted at developers, educators, or generally anyone interested in learning how to use relevant parts of the *Widgets 1.0 Specification* in the ‘real world’. The WAF-WG has not yet set a date for when work will begin on this deliverable, but we hope to have something by the end of the year⁵³.

Being an Editor at the W3C

I have mentioned that I am the *editor* of the *Widgets 1.0 Requirements* and *co-editor*⁵⁴ of the *Widgets 1.0 Specification*, but have so far neglected to give a proper definition of the role and responsibilities of an editor at the W3C. Sperberg-McQueen (1999) explains that, at the W3C,

“...the editor should expect to have all the responsibilities of the author but none of the rights. That is, the editors should definitely expect to draft the specification without relying on wording from others in the [Working Group (WG)], but not to have the kind of intellectual ownership that comes from being the ones who determine what should and should not be in the spec. The intellectual responsibility for the spec should remain with the WG, and the editors need to be willing to write into the spec whatever the WG decides, even if they personally would prefer something else.”

The Process in the ‘Real World’

Although I have presented the standardisation process as if it is linear, in reality it is not. The process of standardisation is much more organic and messy, with all documents and the test suite

⁵³ I’m still undecided as to whether I will contribute to the Primer deliverable at this time.

⁵⁴ Together With Anne van Kesteren, who represents Opera Software in the Working Group.

being developed simultaneously and continuously feeding back on each other. There are many advantages to the simultaneous development of all aspects of the standardisation process, particularly in the simultaneous development of the specification and the test suite (Rosenthal, Skall, Bradly, Kass, & Mntanez-Rivera, 1997) (Dubost, Test Development FAQ, 2005).

What the W3C Provides

As I have made clear in this document, the success of this PhD work relies heavily on the World Wide Web Consortium. The W3C provides the infrastructure that I need to conduct this research. In general, the W3C provides:

- **Extensive peer, vendor, developer and public review and collaboration on a global scale:** Through the WAF-WG, member organisations and the public are able to openly collaborate on the development of standards. In addition, the W3C provides me with access to some of the world's recognised experts in the fields of Computer Science, Software Engineering, and Web Science.
- **An applied and established standardisation process through which ideas can be extensively developed and tested:** the W3C provides me with the infrastructure for promoting, certifying, and deploying freely available specifications and test suites.
- **Open channels of communications and means of collaboration with peers, vendors, developers and the public:** the W3C hosts public mailing lists and Internet Relay Chat (IRC) channels allow me to collaborate with vendors, developers, and users. I meet face-to-face with members of the Working Group four times a year, in various cities around the world, to review and discuss publications and ideas⁵⁵. The W3C also hosts an international conference once a year and at least one plenary meeting with all its members per annum⁵⁶. This gives people involved various opportunities to voice their opinions, and helps bolster the chances that the specifications that I am currently collaboratively working on will be implemented by vendors and ultimately endorsed by the W3C as a Recommendation.
- **Promotion and access to publications:** The W3C mandates the regular publication of Working Drafts for review by the public (Jacobs, 2005), which they promote though the W3C homepage⁵⁷. As I have mentioned previously, both the *Widgets 1.0 Requirements* and the *Widgets 1.0 Specifications* have been featured on the W3C front page; resulting in wide public and media exposure and feedback from the public. Unlike some journal publications

⁵⁵ So far, I have attended three meetings: Madrid in November 2006; Boston in February 2007; and I hosted one in Brisbane in April, 2007.

⁵⁶ See Timeframe section for meeting held in Boston.

⁵⁷ See <http://w3.org>

and some other standards bodies, documents published by the W3C are always free of charge⁵⁸ and available to all⁵⁹.

To convert the ideas in this PhD into practice it is imperative that research outputs are tested on a global scale and for a standards organisation like the W3C offers an established path (Jacobs, World Wide Web Consortium Process Document, 2005). However, the standardisation process is feeble unless it engages the relevant communities (Egyedi & Joode, 2006).

Engaging the Relevant Communities

The findings from this PhD have already started to be disseminated to various relevant communities in the form of W3C publications, public mailing-lists discussions and announcements, via the popular news websites for developers, such as *Ajaxian.com*⁶⁰, international meetings, conferences, and presentations. Publishing through the W3C website also has the advantage of large media exposure. For example, after the publication of the *Widgets 1.0 Requirements* (Caceres, Widgets 1.0 Requirements, 2007), *PC Pro Magazine* published an article online summarising key requirements outlined in the document (Aughton, 2007).

Furthermore in regards to engaging relevant communities, in November 2006, I presented the widget standardisation documents to the widget engine development teams at both Nokia Research and France Telecom Research in Boston, Massachusetts⁶¹. In April of 2006, I held a W3C Working Group meeting at QUT, where working group members that represent large international organisations, like Nokia and Opera Software, attended. In addition, I am currently collaborating with a team of researchers from SAP⁶² Research Australia to derive requirements that will allow widgets to work more effectively in enterprise settings. I have also been invited to give a workshop to developers at the *Web Directions South* conference⁶³, to be held in Sydney in September 2007. I also keep a blog⁶⁴ where I periodically publish updates about the standardisation effort and generally keep people up to date with what I am doing in regards to widgets. Plus I monitor and

⁵⁸ Unlike ISO, where gaining access to a single standard can cost an individual around AU\$180; or Journal articles, where from my experience the cost to view a single article can range from around AU\$25 upwards.

⁵⁹ That have an internet connection and a means to access the w3c website.

⁶⁰ See the following URL for one such published article. It also contains a discussion between *Ajaxian* readers and myself (<http://ajaxian.com/archives/W3C-widgets-10-requirements>).

⁶¹ Presentation slides available at <http://datadriven.com.au/2007/presentations/widgets/widgets.ppt>

⁶² According to the *Wikipedia*, SAP is Europe's largest software company and the fourth largest world-wide. In 2006 it had revenues in excess of 9.4Billion Euros. See http://en.wikipedia.org/wiki/SAP_AG or <http://sap.com> for SAP's official homepage.

⁶³ See <http://www.webdirections.org/program/workshops/#W3Csig>

⁶⁴ See <http://datadriven.com.au>

make announcements on relevant developer mailing lists and forums, such as the Yahoo! Widgets developers' forum⁶⁵, and Apple Dashboard developers' mailing list⁶⁶.

As this is a collaborative effort, other members of the WAF-WG also advocate the promotion of the WAF-WG's the *Widgets 1.0 Requirements* and *Widgets 1.0 Specification* at major international forums. For example, Arthur Barstow, who is the Working Group Chair, presented the state of these specifications at both the 2006 and 2007 *International World Wide Web Conference*, held in and Edinburgh, Scotland, and Alberta, Canada respectively (Barstow, 2006) (Barstow, 2007); And Charles McCathieNevile, the Chief Standards Officer of Opera Software, spoke to developers about the standardisation effort at a Birds of a Feather session held at the International *XTech 2007* developers conference⁶⁷. McCathieNevile also presented at the *Widget Week 2007*⁶⁸, held in London... to name a few significant events. It is also important for this standardisation process that developers write blog entries⁶⁹ about what is said at these events and further advocate (or otherwise review, criticise, or comment on) the standardisation effort. I collect and reflect on as many blog entries as I can on my blog⁷⁰. Other members of the WAG-WG members attempt to engage the developer community by either publishing blog entries⁷¹ or chatting to developers directly using Internet Relay Chat⁷².

Engaging the various communities and keeping track of developers' responses (thoughts, concerns, criticism, etc) on the standardisation effort are vital to the *methods* that I am employing to successfully complete this work. The *methods* are the topic of Section 2.

What else I have been doing

As already mentioned, in this PhD, I have set out to provide the research that will drive the development of the W3C specifications:

- *Widgets 1.0 Requirements*.
- *Widgets 1.0 Specification*.
- *Widgets 1.0 Test Suite*.

⁶⁵ <http://www2.konfabulator.com/forums/>

⁶⁶ Email dashboard-dev-request@lists.apple.com

⁶⁷ See <http://2007.xtech.org/public/schedule/detail/224>

⁶⁸ <http://live.chinwag.com/widgetweek>

⁶⁹ See for example <http://oatmealstout.wordpress.com/2007/05/09/www2007-art-barstow-on-widgets-and-web-applications/> and <http://innovationeye.wordpress.com/2007/05/20/widget-week-part-1-mobile-monday-mobile-widgets/> and http://www.pavingways.com/xtech-2007-widgets-and-mobile_97.html.

⁷⁰ See <http://datadriven.com.au/2006/11/17/widgets-10-responses-and-thoughts>

⁷¹ Eg <http://virtuelvis.com/archives/2006/11/widgets-10-working-draft> and

<http://annevankesteren.nl/2006/11/widgets>

⁷² See <http://www.greywyvern.com/code/opera/chats/default07.htm> for transcripts.

In addition, I will also be working independently on the following publications. The following publications will be the basis on which this PhD will be defended.

- *Widgets and incompatibilities across widget engines: use cases and requirements*
- *Overcoming incompatibilities in widgets through standardization*
- *Design and issues of the Widgets 1.0 Test Suite*

These papers will be bundled with a short introductory chapter and submitted for assessment. In other words, the above three papers constitute the PhD.

In addition, to get an in-depth understanding of the technologies involved in creating widgets, I will also be creating a number of widgets for desktop and mobile environments.

Independent Publications.

The paper, *Widgets and incompatibilities across widget engines: use cases and requirements*, basically build upon the issues discussed in Section 1 of this document. The paper is intended to formally define widgets, expose the incompatibilities and recommend standardization as a possible solution. I intend to submit this paper to the 17th International World Wide Web Conference, to be held in Beijing, China, in May 2008.

The paper, *Overcoming incompatibilities in Widgets through standardization*, will justify the technical solutions that went into the Widgets 1.0 Specification. It will discuss what alternative solutions where possible and how the specification came to be. I intend to submit this paper to the 18th International World Wide Web Conference, to be held in Madrid, Spain, in May 2009.

The paper, *Design and Issues of the Widgets 1.0 Test Suite*, will discuss, as the name suggests, the design decisions and issues encountered in the creating of the test suite for the Widgets 1.0 specification. I will find a suitable journal or conference for publication at a future date.

Once completed, these three publications will be bundled together with a short summarizing chapter, or *exegesis*, that will describe the PhD process as a whole. The purpose will be to give context to the three publications so they can be understood as a whole, obviously. The final submitted work will likely contain the Widgets 1.0 Requirements and Widgets 1.0 Specification as an appendix.

Experimenting With Widgets

Reading and writing *about* widgets is limited in that it lacks the crucial experiential component of building and deploying widgets in the 'real world'. Kensing and Blomberg (1998, p. 175) would argue

that that it would be more difficult to understand the needs and requirements of developers, vendors, and users, without having first-hand experience in widget development and deployment in multiple contexts and devices. For that reason I have developed some widgets for the Computing Services Section of QUT's Creative Industries Faculty to help me understand the use cases for widgets within an enterprise setting. In addition, this method of research has also allowed me to understand differences between widget engines and web browsers⁷³.



In the time remaining to complete this PhD, I also intend to design and build some widgets for mobile devices to get a better understanding of issues relating to widgets' devices independence and mobility. I have scheduled time for further widget experimentation in the Timeframe on page 41. Below, I describe the widgets I have created so far.

To date, I have built two widgets for the Creative Industries Faculty:

- *Creative Industries Network Alerts Widget*
- *QUT Web Access Widget*

Creative Industries Network Alerts Widget

The *Creative Industries Network Alerts Widget*⁷⁴ (Figure 6) allows Creative Industry administrators to inform staff and students of any disruption to commonly used network services (eg. server failures, performance issues, etc.). Administrators send out alert through a simple web interface⁷⁵. The widget then receives the alert by periodically polling a web server for any new alerts. If an alert is found, then it is displayed to the user. This widget has succeeded in technical trials and has now been deployed on laboratory machines used by students.

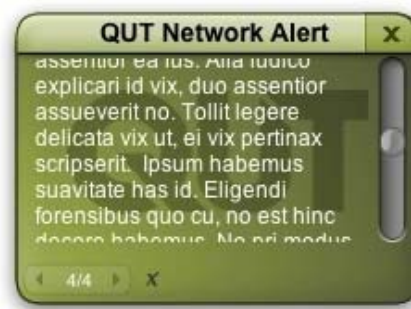


Figure 6. The *Creative Industries Network Alerts Widget*.

QUT Web Access Widget

The *QUT Web Access Widget* (Figure 7) is intended to emulate the behaviour of a browser pop-up window that QUT uses to allow staff and students to access the web (Figure 8). Without going into the technical details of how the popup window actually works, the intention of the widget is to emulate the behaviour of the pop up window without needing the user to open their web browser.

Figure 7. The *QUT Web Access Widget*

There are a number of advantages of using the widget over the pop-up window:

⁷³ I have been professionally creating web applications for a number of years, so I have a fairly good understanding of how web browsers behave.

⁷⁴ built to run on the *Yahoo! Widget Engine* (Yahoo! Inc, 2007)

⁷⁵ <http://alerts.ci.qut.edu.au>

- No need for the user to continuously authenticate when they browse the web; the widget keeps them logged in even after a reboot.
- Smaller memory footprint than running a web browser⁷⁶.
- Both secure and persistent storage of user's credentials.
- Automatic reconnection on timeout or network disruption⁷⁷.
- One less application in the task bar.

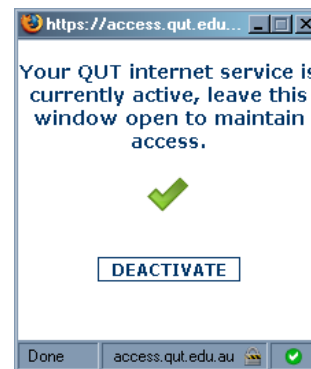


Figure 8. QUT's Web Access "Keep Alive" pop-up window.

Although I have built a proof of concept, the *QUT Web Access Widget* is still under development and is yet to be deployed. I hope to deploy the widget sometime in the next three months.

After both the widgets have been deployed, the next step will be to port these widgets to other widget engines to allow me to research the differences between widget platforms. The platforms I intend to port these widgets to include Windows Vista's *Sidebar* and Apple's *Dashboard*. This will coincide with QUT's roll-over to Windows Vista in 2007. If the project is successful and shown to be secure, then there is a chance that the *QUT Web Access Widget* could be made available to all QUT Staff and students⁷⁸.

Summary of Section 1

In this section I have explained that widgets are single purpose applications intended for everyday use, which are built using many of the same technologies used to build web pages. I also showed that ITC companies have become interested in widgets and that widgets offer potential for generating revenue. I then argued that a widgets market without standards has become problematic, specifically in regards to development, distribution and deployment, accessibility, security, metadata, internationalisation and device-independence. I then proposed that the most effective way to address these issues is for vendors, developers, and the public to collaborate through the W3C. I provided an overview of the standardisation process, noting that a requirements document, a specification, a test suite, and a primer were going to be published by the WAF-WG, with a detailed account of my involvement in each of these publications. I then argued that it is

⁷⁶ Particularly when compared to running a browser like Firefox 2.0.x or Internet Explorer 7. At worst, memory usage is comparable, but rarely, larger.

⁷⁷ The pop-up cannot reconnect itself, especially if there is a network disruption.

⁷⁸ QUT's Information Technology Services have been informed of this project and further discussions about deployment will be organised once a fully functional proof of concept is available.

imperative that relevant communities are engaged in the standardisation process. I also outline the strategies that I, and other members of the working group, would use to keep the relevant communities informed about what we are doing. The strategies include the exploitation of the media, the web, blogs, the W3C homepage, Internet Relay Chat, public mailing list, meetings, conferences, and so on. Finally, I discussed how creating widgets and conducting a Contextual Review has helped me put together this research, including various parts of this document.

In the next section, I describe what *methods* I have been applying to do what I have described in Section 1, and provide a *timeframe* to show what else is left to be done on this PhD over the remaining two years.

Section 2: Methods and Timeframe

This section describes the methods I have been using to collaboratively participate in the development of the following specifications:

- The *W3C Widgets 1.0 Requirements*,
- the *W3C Widgets 1.0 Specification*,
- the *W3C Widgets 1.0 Test Suite*.

Because the collaborative editorial nature of standards development throws into confusion the “authorship” aspect of a specification, three independent publications will be created for the purposes of assessment of this PhD together with a short summarizing chapter. As stated previously, the three publications that will make up this PhD include:

- *Widgets and incompatibilities across widget engines: use cases and requirements*
- *Overcoming incompatibilities in widgets through standardization*
- *Design and issues of the Widgets 1.0 Test Suite*

Please refer to the *Independent Publications*. section on page 29 for more details about each of these publications.

To edit the *Widgets 1.0 Specification* and the *Widgets 1.0 Requirements* I have been applying methods of *specification verification* and *specification writing*. Specification writing, as the name implies, are strategies and techniques that, when properly applied, result in “better specifications, by making a specification easier to interpret without ambiguity and clearer as to what is required in order to conform” (Dubost, Rosenthal, Hazaël-Massieux, & Lofton, 2005). Specification verification is concerned with verifying that what is written in a specification is formally correct and testable (Toroi & Eerola, 2006). In addition, to collaboratively gather, discuss, and disseminate information about the *Widgets 1.0 Specification* and the *Widgets 1.0 Requirements* I have been applying *participatory design*. Muller (2002) defines participatory design as “a set of theories, practices, and studies related to *end-user* as full participants in the activities leading to software and hardware computer products and computer-based activities”.

For the *Widgets 1.0 Test Suite*, I am currently researching applicable methodologies in the literature of *specification-based software testing*. Stocks and Carrington (1996, p. 778) explain that the purpose of specification-based software testing “is to demonstrate that an implementation conforms to the specification”. However, as Stocks and Carrington (1996, p. 778) note that,

“demonstrating that an implementation conforms to an incorrect specification is of dubious value”. So, specification verification, which is part of the methods of software-based specification testing, provides means to verify the quality of a specification (Patton, 2006, pp. 53-62): Stocks and Carrington (1996, p. 778) state that “ensuring that the specification is correct... is perhaps the most crucial task in any software engineering project”. To the *Widgets 1.0 Test Suite* I will also be applying participatory design methodologies to make sure that is usable by those people that actually need to do the conformance testing.

To create the Widgets for QUT, I have been using methods of *software development* as a means to research what widgets are, and how they are developed, and the differences across widget engines. Software development provides the process and methods used to develop computer software (McConnell, 1996). In the process of deploying those widgets, I have also been applying participatory design to make sure they meet the needs of QUT Staff.

To conduct the Contextual Review, I have been combining methods for conducting a Contextual Review (Gray & Malins, 2004, pp. 35-65) with software testing. Methods for conducting a Contextual Review are limited in their ability to discuss software, so software testing will provide me with the practical methods and processes to test and report on various aspects of software I need to investigate for my research (Whittaker, 2003, pp. 4-5) (Patton, 2006, pp. 19-21).

The application of these methods to the outputs of this PhD can be seen in Figure 9.

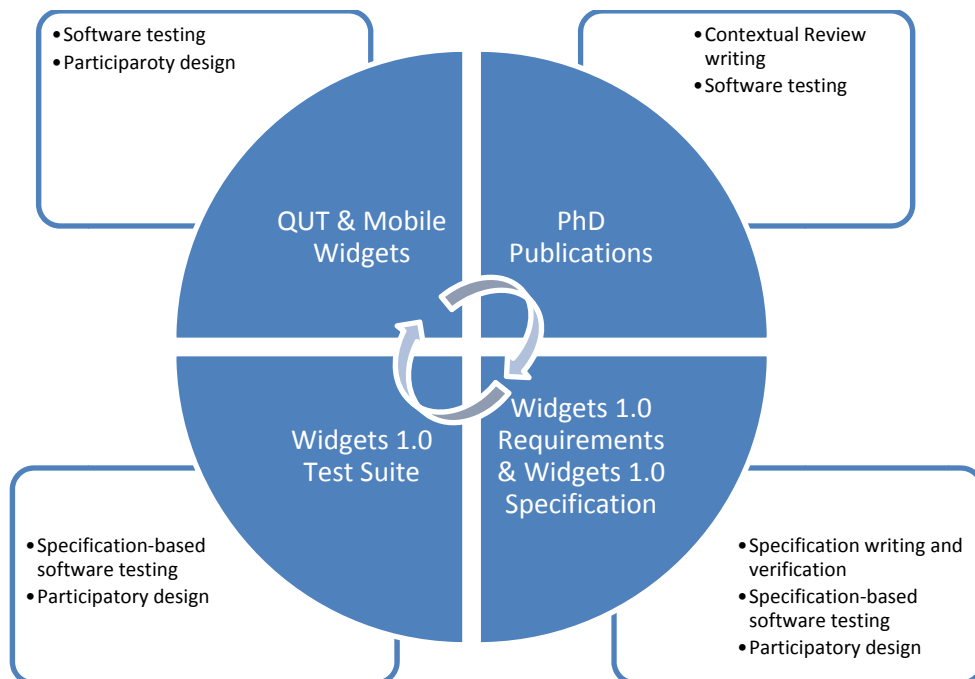


Figure 9. Methods I am applying to create the outputs of this PhD.

In summary, the methods I have been applying are:

- *Specification-based software testing.*
- *Specification verification and writing.*
- *Participatory design.*
- *Software Development (as research).*
- *Contextual Review & software testing.*

Selection Criteria for the Methods

In the beginning of this document, I presented the formal definition of a standard as defined by the International Standards Organisation (ISO) and the International Electronic Commission (IEC). I chose that particular definition because it contains a “Note” that implies the expected methodological processes that go into creating a standard:

“Note - standards should be based on the consolidated results of science, technology and experience, and aimed at the promotion of optimum community benefit.” (ISO/IEC, 2004) in (Vries, 2006, p. 4).

In relation to methods, my interpretation of this definition is:

- The research should be conducted using methods that involve people who use the technology in practice and in everyday life.
- The methods should be established and, where appropriate, scientifically rigorous: meaning that outputs must be verifiable, repeatable, and testable.
- All outputs should be royalty free, usable, accessible, and beneficial for their intended community and as well as for the wider community at large.

In the following sections, I describe how I have applied each the methods I have mentioned in the section above. Where directly applicable to the widgets standardisation process, I also explain how the methods meet my interpretation of the methodological criteria set by the ISO/IEC definition of a standard.

Specification Writing and Verification

The methods for writing a specification differ from other forms of writing in that it must make a clear distinction between content that is *informative* and content that is *normative*: “Normative content is the prescriptive part of the specification, whereas informative content is for informational purposes

and assists in the understanding and use of the specification." Furthermore, writing a specification requires the precise presentation of a *conformance model*. Dubost et al (2005) explain that,

"A clear presentation of conformance is crucial to successful interoperability of implementations. The conformance model and the language used for normative information determine the testability of a specification. They also influence the overall implementation landscape, ranging from a narrow conformance with few allowable variations in implementations to multiple conformance types, resulting in numerous variations in conforming implementations. The model must be chosen carefully, to produce the intended implementation range."

In order to assist editors write specifications, the W3C created the *QA Framework: Specification Guidelines* (Dubost, Rosenthal, Hazaël-Massieux, & Lofton, 2005). The guidelines present a methodology that "help W3C editors write better specifications, by making a specification easier to interpret without ambiguity and clearer as to what is required in order to conform. It focuses on how to define and specify conformance. It also addresses how a specification might allow variation among conforming implementations. The document presents guidelines or requirements, supplemented with good practices, examples and techniques."

The *Specification Guidelines* lists a set of requirements, good practices, and techniques that guide editors on how to write a specification that conforms to the W3C's Quality Assurance Framework. W3C specification must conform to the requirements of the framework in order to proceed to become a standard. The framework provides the requirements, good practices, and techniques for how to:

- Write a conformance clause.
- Define the scope of the specification.
- Identify who and/or what will implement the specification.
- Create a list of normative references.
- Clearly define the normative terms used in the specification.
- Clearly distinguish normative and informative sections of the specification by labelling and explicit use of a particular writing style.
- Indicate if a conformance requirement is mandatory, recommended, or optional.
- Write testable assertions.
- Identify deprecated feature, and how to define how products should handle or avoid those features.

- Define error handling mechanisms.

The W3C also provides a supplementary document, called *Variability in Specifications* (Hazaël-Massieux & Rosenthal, 2005), which provides methods to analyse “how design decisions of a specification's conformance model may affect its implementability and the interoperability of its implementations.”

As mentioned earlier, formally verifying the validity of conformance clauses in a specification is of critical importance for specification-based software testing (Stocks & Carrington, 1996). Significant literature and established methods exists on how to perform verification of a specification (Toroi & Eerola, 2006). As work continues on the *Widgets 1.0 Specification*, I will continue searching for a suitable method for verifying the normative clauses of the *Widgets 1.0 Specification*.

Meeting the ISO/IEC Methodological Criteria

Given that the W3C's *QA Framework: Specification Guidelines* has been applied in the creating of a large number⁷⁹ of successful W3C Specifications, it is within the scope of methodological requirements as prescribed by the ISO/IEC definition. There is also limited risk in adopting this methodology as it is widely adopted within the context of the W3C and the WAF-WG.

Specification-Based Software Testing

Specification-based software testing provides specific strategies for effective testing against formally written software specifications. Candidate methodologies for the *Widgets 1.0 Specification* include:

- The book, *Testing Applications on the Web: Test Planning for Mobile and Internet-based Systems* (Nguyen & Hackett, 2003).
- The paper, “Web-Based Conformance Testing for VRML” (Rosenthal, Skall, Bradly, Kass, & Mntanez-Rivera, 1997).
- The paper, “A Framework For Specification-Based Testing” (Stocks & Carrington, 1996).

Meeting the ISO/IEC Methodological Criteria

As just stated, research is forthcoming into which methodology will be most appropriate for creating *the Widgets 1.0 Test Suite*.

Participatory Design

Levinger defines participatory design as:

⁷⁹ 84 Recommendations and 19 Proposed Recommendations, as of the 17 of January 2007 (Dubost, The Matrix of W3C specifications, 2007).

“a set of diverse ways of thinking, planning, and acting through which people make their work, technologies, and social institutions more responsive to human needs. PD practitioners aim to improve conditions of work and the quality of life by involving workers, users, and community members in design and development. PD enables users, stakeholders, and other interested parties to play powerful roles in shaping technological and work outcomes to reflect their interests.” (Levinger, 1998)

So, participatory design attempts to involve all *end-users* at every stage of development. It is well-known that the concept of an end-user is problematic (Jakobs, Procter, & Williams, 1996, p. 184), particularly in a computing environment where actual interaction with software is always mediated via the operating system (Whittaker, 2003, pp. 3-16). For a specification, the definition of an end-user needs to be broad enough to accommodate both humans and machines (Dubost, Rosenthal, Hazaël-Massieux, & Lofton, 2005); (Jakobs, Procter, & Williams, 1996, p. 184). Particularly in the case of widgets, where the outputs of the widget standardisation process must be usable by those that:

- Generate widgets, either manually or automatically (developers⁸⁰ or software).
- Implement the specification as software (software engineers) .
- Do conformance testing of an implementation against a test suite (software testers and/or software).
- Check the validity of a widget (developers or conformance checker).
- Make the decision to allow widgets to be used in an enterprise setting (IT Managers).
- Use widgets on a regular basis (people or autonomous software agents).

The *W3C Quality Assurance Framework* puts in place the guidelines and requirements to address the needs of end-users in the specification. However, the framework is limited in that it does not actually provide any practical strategies to engage human end-users in the standardisation process; a gap that can be partly filled by participatory design.

To again quote Sherif (2006, p. 129), “the production of technical specifications intertwines technological considerations with business, social, interpersonal, political, and ideological aspects.” Participatory design can assist in addressing the needs of the developers, business, users, and the community at large by actively seeking out their direct involvement in the design and implementation of a specification: “there are two approaches to participatory design: 1. Bring the designers to the workplace. 2. Bring the workers to the design room.”

⁸⁰ Including enthusiasts and amateur authors.

On page 27 of Section 1, I listed how both I and other members of the working group are actively engaging the relevant communities. For me personally, my attempts to engage the relevant communities are guided by participatory design principles and strategies, particularly as outlined in (Muller, 2002) and (Kensing & Blomberg, 1998). For example, having workshops, presenting at companies where people are building widget related software, communicating with developers and other stakeholders as much as possible, etc.

Meeting the ISO/IEC Methodological Criteria

Participatory design is a well established methodology with a twenty year research tradition (Muller, 2002) (Kensing & Blomberg, 1998). I would argue that it is suitable for meeting the ISO/IEC methodological requirements, because every attempt is made to understand the needs of those who will be using the technology; hence, participatory design is, in a very practical and proven sense, geared toward achieving 'optimum community benefit.'

Mixing the Contextual Review and Software Testing

Gray and Malins (2004, p. 37) explain that conducting a Contextual Review is a two phase approach:

1. **Initial survey:** serves to "establish the proposal's rational... gain some background information around the proposed topic" and finding relevant literary sources or related materials.
2. **Critical review:** "involves the placement of these references/sources into a critical review of the research context to enable identification of your own particular research question and the development of an argument."

The methodology for conducting a Contextual Review provides an effective structure for critically reflecting on literature. However, the methodology on its own lacks the necessary tools to do any meaningful experimentation and reporting about widget engines.

To be able to test the various crucial aspects of widgets and widget engines, I have been applying the methodologies from software testing. For exploring the differences between widget engines, software testing provides a structured approach to test and report on (Whittaker, 2003); (Patton, 2006):

- Error handling.
- User interfaces.
- Configuration and Metadata.

- Network communications related aspects.
- Programming interfaces.
- Internationalisation and localisation.
- Security.

Preliminary documented outcomes of applying software testing can be found in the *Issues with Widgets: A Contextual Review* section on page 7 of this document, and in the Appendix of the *Widgets 1.0 Requirements* document. As stated earlier, the Contextual Review is an ongoing process.

Software Development

As stated earlier, actively creating widgets as a form of research may serve as an effective strategy to get an in-depth understanding of issues related to:

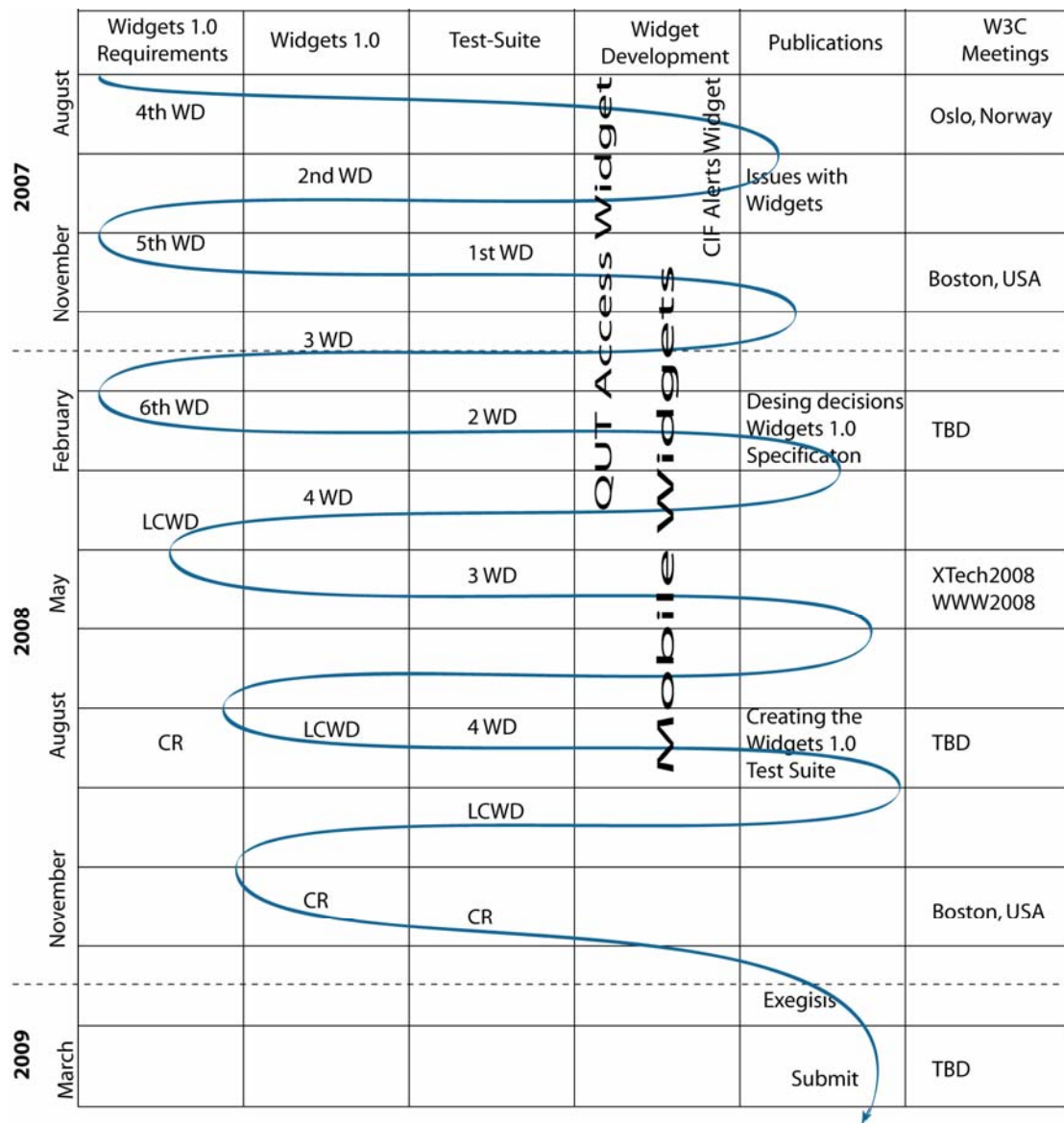
- How to build widgets.
- Differences between widgets engines.
- Limitations of widgets engines.
- Differences between widget engines and web browsers.
- Use cases for widgets in different settings (enterprise, mobile).
- Security issues.
- Etc.

The methodology I have employed to develop the widgets is the well-known *evolutionary prototyping* software development model (McConnell, 1996, p. 147). Although not as structured as other software development methodologies, such as the *spiral model* (McConnell, 1996, p. 141), evolutionary prototyping is well suited for small applications like widgets because it focuses on evolving an application based on user and stakeholder feedback. According to McConnell (1996, p. 147), the process is:

1. Come up with and Initial concept (based on user/stakeholder requirements).
2. Design and implement an initial prototype.
3. Get feedback from users and stakeholders.
4. Refine prototype, do step 3 again, or
5. Complete and release prototype.

Timeframe

The figure below shows what I'm going to produce over the twenty-one months remaining of this PhD. The path demonstrates the simultaneous development of the W3C publications, widgets and the Contextual Review, as well as the continuous cycle of peer and public review. The diagram also shows the when and where W3C Working Groups meetings are taking place. The publications cycle is mandated by the W3C's "heartbeat" requirement, which states that "each Working Group MUST publish a new draft of at least one of its active technical reports on the W3C technical reports index at least once every three months" (Jacobs, World Wide Web Consortium Process Document, 2005).



WD = Working Draft
 LCWD = Last Call Working Draft
 CR = Candidate Recommendation

Figure 10. PhD timeline, showing W3C publications, review periods and Working Group meetings.

Bibliography

adesklets. (2007). *Welcome to adesklets.sf.net*. Retrieved May 15, 2007, from aDesklets: <http://adesklets.sourceforge.net/desklets.html>

Alvestrand, H. (1995, March). *Tags for the Identification of Languages*. Retrieved from Internet Engineering Taskforce (IETF): <http://www.ietf.org/rfc/rfc1766.txt>

Apple Inc. (2007, January 9). *iPhone*. Retrieved April 23, 2007, from Apple Website: <http://www.apple.com/iphone/>

Apple Inc. (2007, March 19). *Apple Mac OS X Dashboard*. Retrieved March 19, 2007, from Apple Inc.: <http://www.apple.com/macosx/features/dashboard/>

Aughton, S. (2007, February 12). *W3C draws up widgets wish-list*. Retrieved June 1, 2007, from PC Pro: Computing in the Real World: <http://www.pcpro.co.uk/news/104307/w3c-draws-up-widgets-wishlist.html>

Barstow, A. (2006, May 24). *Declarative Formats for Web Applications*. Retrieved June 1, 2007, from World Wide Web Consortium: <http://www.w3.org/2006/Talks/0524-www-WAF.pdf>

Barstow, A. (2007, May 9). *Widgets & Web Application Formats*. Retrieved June 8, 2007, from World Wide Web Consortium: <http://www.w3.org/2007/Talks/WWW2007-WAF-May-09.pdf>

Berners-Lee, T., & Fischetti, M. (2000). *Weaving the Web: The Past, The Present and Future of the World Wide Web by its Inventor*. London: TEXERE Publishing Limited.

Berners-Lee, T., Fielding, R., & Masinter, L. (2005, January). *Uniform Resource Identifier (URI): Generic Syntax*. Retrieved March 17, 2007, from IETF: <http://tools.ietf.org/html/rfc3986>

Berners-Lee, T., Hall, W., Hendler, J., Shadbolt, N., & Weitzner, D. J. (2006). *Creating a Science of the Web. Science*, 313.

Besen, S. M., & Farrell, J. (1994). *Choosing How to Compete: Strategies and Tactics in Standardization. The Journal of Economic Perspectives*, Vol. 8 (No. 2), 117-131.

Bos, B., Çelik, T., Hickson, I., & Wium Lie, H. (2006, November 6). *Cascading Style Sheets, level 2 revision 1*. (B. Bos, T. Çelik, I. Hickson, H. Wium Lie, Editors, & W3C) Retrieved April 12, 2007, from World Wide Web Consortium: <http://www.w3.org/TR/2006/WD-CSS21-20061106>

Bradner, S. (1997, March). *Key words for use in RFCs to Indicate Requirement Levels*. Retrieved 6 5, 2007, from The Internet Engineering Task Force (IETF): <http://www.ietf.org/rfc/rfc2119.txt>

Braiker, B. (2006, December 30). *The Year of the Widget?* Retrieved May 2007, 12, from Newsweek: <http://www.msnbc.msn.com/id/16329739/site/newsweek/>

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., & François, Y. (2006, August 16). *Extensible Markup Language (XML) 1.0 (Fourth Edition)*. (T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, & Y. François, Eds.) Retrieved April 12, 2007, from <http://www.w3.org/TR/2006/REC-xml-20060816>

Bunsson, N., & Jacobsson, B. (2000). The Contemporary Expansion of Standardization. In N. Bunsson, & B. Jacobsson (Eds.), *A World Of Standards*. New York: Oxford University Press.

Caceres, M. (2006, July 26). *Packaging Web Applications: Requirements*. Retrieved June 1, 2007, from World Wide Web Consortium: <http://dev.w3.org/cvsweb/~checkout~/2006/waf/WAPF/WD-WAPF-REQ-20060726.html?rev=1.1>

Caceres, M. (2006, August 21). *Web Applications Packaging Format Requirements*. Retrieved June 2, 2007, from World Wide Web Consortium: <http://www.w3.org/TR/2006/WD-WAPF-REQ-20060821/>

Caceres, M. (2007, February 09). *Widgets 1.0 Requirements*. (M. Caceres, Ed.) Retrieved March 19, 2007, from World Wide Web Consortium: <http://www.w3.org/TR/2007/WD-widgets-reqs-20070209/>

Caceres, M. (2007, February 09). *Widgets 1.0 Requirements*. (M. Caceres, Ed.) Retrieved March 19, 2007, from World Wide Web Consortium: <http://www.w3.org/TR/2007/WD-widgets-reqs-20070209/>

Caceres, M., & Kesteren, A. v. (2006, November 9). *Widgets 1.0 Specification*. (M. Caceres, & A. van Kesteren, Eds.) Retrieved May 30, 2007, from World Wide Web Consortium: <http://www.w3.org/TR/widgets/>

Chisholm, W., Vanderheiden, G., & Jacobs, I. (1999, May 5). *Web Content Accessibility Guidelines 1.0*. (W. Chisholm, G. Vanderheiden, & I. Jacobs, Eds.) Retrieved June 12, 2007, from World Wide Web Consortium: <http://www.w3.org/TR/WAI-WEBCONTENT/>

Dubost, K. (2005, June 23). *Test Development FAQ*. Retrieved June 2, 2007, from World Wide Web Consortium: <http://www.w3.org/QA/WG/2005/01/test-faq>

Dubost, K. (2007, January 17). *The Matrix of W3C specifications*. Retrieved June 12, 2007, from World Wide Web Consortium: <http://www.w3.org/QA/TheMatrix>

Dubost, K. (2002, January 28). *W3C Quality Assurance Glossary*. Retrieved 4 11, 2007, from World Wide Web Consortium: <http://www.w3.org/QA/glossary>

Dubost, K., Rosenthal, L., Hazaël-Massieux, D., & Lofton, H. (2005, August 17). *QA Framework: Specification Guidelines*. (K. Dubost, L. Rosenthal, D. Hazaël-Massieux, & H. Lofton, Eds.) Retrieved 6 1, 2007, from World Wide Web Consortium: <http://www.w3.org/TR/2005/REC-qaframe-spec-20050817/>

Duerst, M., & Suignard, M. (2005, January). *Internationalized Resource Identifiers (IRIs)*. Retrieved March 17, 2007, from IETF: <http://www.ietf.org/rfc/rfc3987>

ECMA. (1999, December). ECMA-262: ECMAScript Language Specification. (3rd). European Computer Manufacturers' Association (ECMA). Retrieved from European Computer Manufacturers Association: <ftp://ftp.ecma.ch/ecma-st/Ecma-262.pdf>.

Egyedi, T. M., & Joode, R. v. (2006). Standardization and Other Coordination Mechanisms in Open Source Software. In K. Jakobs, *Information Technology Standards and Standardization Research* (pp. 71-90). Hershey: Idea Group Publishing.

Fielding, R., Gettys, J., Mogul, J., Frystyk Nielsen, H., Masinter, L., Leach, P., et al. (1999, June). *Hypertext Transfer Protocol -- HTTP/1.1*. Retrieved March 17, 2007, from IETF: <http://www.ietf.org/rfc/rfc2616.txt>

Garrett, J. J. (2005, February 18). *Ajax: A New Approach to Web Applications*. Retrieved June 1, 2007, from Adaptive Path: <http://www.adaptivepath.com/publications/essays/archives/000385.php>

gDesklets. (2007). *gDesklets: desktop eye candy*. Retrieved May 14, 2007, from gDesklets: <http://adesklets.sourceforge.net/desklets.html>

Gentry, C. (2003). Certificate-Based Encryption and the Certificate Revocation Problem. *Advances in Cryptology - EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques*. Warsaw, Poland: Springer Berlin / Heidelberg.

Goodwins, R. (2005, November 01). *The criminal costs of mobile data*. Retrieved June 12, 2007, from ZDNet UK: <http://news.zdnet.co.uk/communications/0,1000000085,39234845,00.htm>

Gray, C., & Malins, J. (2004). *Visualizing Research: A Guide To Research Process in Art and Design*. Aldershot, Hants, England : Ashgate Publishing Limited.

Hart, C. (1998). *Doing a literature review: releasing the social science research imagination*. London : Sage Publications.

Hazaël-Massieux, D., & Rosenthal, L. (2005, August 31). *Variability in Specifications*. Retrieved 6 11, 2007, from World Wide Web Consortium: <http://www.w3.org/TR/2005/NOTE-spec-variability-20050831/>

Henderson, L. (2005, September 6). *The QA Handbook*. (L. Henderson, Ed.) Retrieved June 11, 2007, from World Wide Web Consortium: <http://www.w3.org/TR/qa-handbook/>

Hickson, I. (2006, February 18). *Writing specifications: Kinds of statements*. Retrieved 6 11, 2007, from Hixie's Natural Log: <http://ln.hixie.ch/?start=1140242962&count=1>

Huseby, S. H. (2004). *Innocent Code : A Security Wake-Up Call For Web Programmers*. West Sussex: John Wiley & Sons Ltd.

ISO. (1998). *Code for the representation of names of languages International Standard*. International Organization for Standardization.

ISO. (1997). *Codes for the representation of names of countries and their subdivisions — Part 1: Country codes*. International Organization for Standardization.

ISO/IEC. (2004). *Standardization and related activities: general vocabulary* (8th Edition ed.). Geneva, Switzerland: ISO/IEC.

Jackson, D. (2005, November 15). *Web Application Formats Working Group Charter*. Retrieved June 1, 2007, from World Wide Web Consortium:
<http://www.w3.org/2006/appformats/admin/charter.html>

Jacobs, I. (2007, January 1). *About W3C: Goals*. Retrieved March 7, 2007, from W3C:
<http://www.w3.org/Consortium/mission>

Jacobs, I. (2005, October 14). *World Wide Web Consortium Process Document*. (I. Jacobs, Editor) Retrieved March 7, 2007, from World Wide Web Consortium: <http://www.w3.org/2005/10/Process-20051014/>

Jakobs, K. (2006). *Advanced Topics in Information Technology Standards and Standards Research* (Vol. Vol 1). (K. Jakobs, Ed.) Hershey: Ideas Group Publishing.

Jakobs, K., Procter, R., & Williams, R. (1996). Users and Standardization - Worlds Apart? The Example of Electronic Mail. *Standard View* , Vol.4 (4), 183-191.

Jaokar, A., & Fish, T. (2006). *Mobile Web 2.0: The Innovator's Guide to Developing and Marketing Next Generation Wireless/Mobile Applications*. London: futuretext.

Johnson, D., Menezes, A., & Vanstone, S. (2001). The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security Springer* , 1 (1), 36-63.

Kensing, F., & Blomberg, J. (1998). Participatory Design: Issues and Concerns. *Computer Supported Cooperative Work* (7), 167–185.

Kharif, O. (2007, March 27). *Widgets Gone Wireless*. Retrieved 5 21, 2007, from BusinessWeek Website: http://www.businessweek.com/technology/content/mar2007/tc20070327_532303.htm

Krechmer, K. (2006). Open Standards Requirements. In K. Jakobs (Ed.), *Advanced Topics in Information Technology Standards and Standardization Research* (pp. 27-48). Hershey: Ideas Group Publishing.

Lawrence, S. (2001). Online or Invisible? *Nature* , Volume 411 (Number 6837), 521.

Levinger, D. (1998). Conference preview: PDC '98: participatory design conference “broadening participation”. *interactions* , Volume 5 (5), 44 .

McConnell, S. (1996). *Rapid Development*. Redmond: Microsoft Press.

Mel, H., & Baker, D. (2001). *Cryptography Decrypted*. Boston: Addison-Wesley.

Microsoft Inc. (2007). *Windows Vista: Features Explained: Windows Sidebar and Gadgets*. Retrieved May 12, 2007, from Microsoft Corporation Website:
<http://www.microsoft.com/windows/products/windowsvista/features/details/sidebargadgets.mspx>

Mollins, R. A. (2003). *RSA and Public Key Cryptography*. Florida: CRC Press.

Muller, M. J. (2002). Participatory Design: The Third Space in HCI. In J. A. Jacko, & A. Sears (Eds.), *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications* (pp. 1051-1068). Mahwah, NJ, USA: Lawrence Erlbaum Associates.

NetVibes. (2007). *NetVibes' Universal Widget API (UWA)*. (NetVibes) Retrieved May 12, 2007, from Netvibes' Developers Network: http://dev.netvibes.com/doc/uwa_specification

Nguyen, H. Q., & Hackett, M. (2003). *Testing Applications on the Web : Test Planning for Mobile and Internet-based Systems*. John Wiley & Sons, Inc.

Nokia Inc. (2007, April 16). *Web Runtime: Going beyond Web browsing with widgets*. Retrieved May 17, 2007, from S60 Open To New Features:

<http://www.s60.com/business/productinfo/enablingtechnologies/webruntime>

Opera Software. (2006). *config.xml file syntax (Opera Widgets: Widget information file syntax)*. (A. Bersvendsen, Ed.) Retrieved June 1, 2007, from Opera Software:

<http://oxine.opera.com/widgets/documentation/widget-configuration.html>

O'Reilly, T. (2005, 9 30). *What Is Web 2.0*. Retrieved March 7, 2007, from O'Reilly :

<http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>

Patton, R. (2006). *Software Testing* (2nd Edition ed.). Indianapolis: Sams Publishing.

PKWare Inc. (2006, September 29). *.ZIP File Format Specification*. Retrieved March 17, 2007, from PKWare: <http://www.pkware.com/documents/casestudies/APPNOTE.TXT>

Poston, R. M. (1996). *Automating Specification-Based Software Testing*. Los Alamitos, California, United States of America: IEEE Computer Society Press .

QUT. (2006, November 17). *Queensland University of Technology Doctor of Philosophy Regulations (IF49)*. Retrieved March 9, 2007, from Manual of Policies and Procedures:

<http://www.mopp.qut.edu.au/Appendix/appendix09.jsp>

Raggett, D., Le Hors, A., & Jacobs, I. (1999, December 24). *HTML 4.01 Specification*. (D. Raggett, A. Le Hors, I. Jacobs, Editors, W3C, Producer, & World Wide Web Consortium) Retrieved May 14, 2007, from World Wide Web Consortium: <http://www.w3.org/TR/html401/>

Rosenthal, L., Skall, M., Bradly, M., Kass, M., & Mntanez-Rivera, C. (1997). Web-Based Conformance Testing for VRML. *StandardView* , 5 (3), 110-114.

Savourel, Y. (2001). *XML internationalization and localization*. Indianapolis: Sams.

Sherif, M. H. (2006). Standards for Telecommunication Services. In *Advanced Topics in Information Technology Standards and Standardizaion Research* (pp. 183-205). Hershey: Idea Group Inc.

Sherif, M. H. (2006). When is Standardization Slow? In K. Jakobs, *Advanced Topics in Information Technology Standards and Standardization Research* (p. 128). Hershey: Ideas Group Publishing.

Sivonen, H. (2006, May 25). *XTech 2006*. Retrieved June 12, 2007, from Henri Sivonen's pages: <http://hsivonen.iki.fi/xtech-2006/>

Sperberg-McQueen, C. M. (1999, March 19). *what does an editor do?* Retrieved 6 2, 2007, from World Wide Web Consortium Chairs Mailing List: <http://lists.w3.org/Archives/Member/chairs/1999JanMar/0056>

Stocks, P., & Carrington, D. (1996). A Framework For Specification-Based Testing. *IEEE Transactions on Software Engineering*, 22 (11), 777-793.

Swick, R. R., & Ackerman, M. S. (1988). "The X Toolkit: More Bricks for Building User-Interfaces, or, Widgets for Hire. *Usenix Winter 1988 Conference*, (pp. 221–228).

Toroi, T., & Eerola, A. (2006). Requirements for the Testable Specifications and Test Case Derivation in Conformance Testing. In A. Dasso, & A. Funes, *verification, validation and testing in software engineering* (pp. 118-35). Hershey: Ideas Group Publishing.

U.S. Dep't of Justice & Fed. (2007). *Antitrust Enforcement and Intellectual Property Rights: Promoting Innovation and Competition*. U.S. Department of Justice and the Federal Trade Commission.

Voas, E. (2006, November 10). *Re: [Widgets] Signing*. Retrieved 6 15, 2007, from W3C's public-appformats@w3.org Mailing List: <http://lists.w3.org/Archives/Public/public-appformats/2006Nov/0043.html>

Vries, H. J. (2006). IT Standard Topology. In K. Jakobs (Ed.), *Advanced Topics In Technology Standards and Standardization Research* (Vol. 1). Pennsylvania: Idea-Group.

Whittaker, J. A. (2003). *How To Break Software: A Practical Guide to Testing*. Boston: Person Education Inc.

Yahoo! Inc. (2007). *Yahoo! Widgets - Information*. Retrieved June 11, 2007, from Yahoo Widgets Engine: <http://widgets.yahoo.com/>

Acknowledgements

I would really like to thank everyone who has supported this work and taken the time to review this document. I want to particularly thank my partner, Debra Polson: without your genius, super-human abilities analyse and communicate concepts, and unending support, none of this would be possible. Thank you for enduring all tech-talk, reading draft after draft after draft, putting up with late-night/early-morning keyboard “tippa-tappa”, the loss of nights and weekends together, and being waken at ungodly hours to accommodate me taking part in Working Group teleconferences.

I would also like to thank my parents, Carlos and Maria, for taking the time read over the document and significantly improving its legibility.

I would like to thank members of the WAF Working Group, particularly the Working Group Chair, Art Barstow, and also Guido Grassel: thank you both for being so supportive of me and this work over the last year. I hope we can see the specs all the way through to *Recommendation*. I'd also like to thank Anne van Kesteren for answering all my irritating newbie questions, teaching me XBL2, and converting me into a HTML5 evangelist. I'd also like to thank Dean Jackson for always letting me know what the good TV shows to watch; hope you rejoin the group, Dean! Thanks also to Ian Hickson and members of the WHATWG, also for answering my silly questions and for making me more environmentally aware. Also, big thanks to Lachlan Hunt for coming to my aid with the *XBL Primer*!

I would also like to thank Oksana Selenko, who not only reviewed this document but at the eleventh hour gave me a whole bunch of life saving books on methodology to read. Thanks also to the people who took the time reviewed this document: particularly Cameron McCormack, Angela Button, and Gavin Sade.

Thanks also to Jamie Lonsdale, Danqing Zhang, and Tim Harrap (the best supervisors anyone could ever have!) for allowing me to create widgets for the Creative Industries Faculty (CIF). Your support in letting me work with Computing Services has been invaluable for this project. Thanks also to Tim Morris, who successfully hacked Yahoo!'s Widget Engine to deploy the Network Alerts Widgets.

Thanks to both the Faculty of IT and Faculty of Creative Industries for funding the WAF Working Group Face-to-Face Meeting here in Brisbane, in April 2007. Thanks to Michael Lawley, QUT's W3C Advisory Committee Representative, for your support in joining the WAF Working Group and for helping make the Face-to-Face meeting happen.

Thanks to all Google and Amazon engineers for saving me a thousand walks to the library! *Google Scholar* and *Google Books* totally rock!

Thanks to everyone who has contributed to the widget specs on WAF's public list.

Lastly, I'd like to thank Jeff Jones, and the *Australasian CRC of Interaction Design (ACID)*, who partially funded the development of this work in the form of a top-up scholarship. Jeff, best of luck with your future endeavours... hope I can be part of those too.