From Desktop to Mobile:

A Framework for function and content transformation

By

Geert Dirk Jurgens

2011

From Desktop to Mobile: *A Framework for function and content transformation*

By

Geert Dirk Jurgens

Submitted in fulfillment of the requirements for the degree

Magister Technologiae

in

Information Technology

in the

FACULTY OF ENGINEERING, THE BUILT ENVIRONMENT AND INFORMATION TECHNOLOGY.

of the

NELSON MANDELA METROPOLITAN UNIVERSITY

Supervisor: Professor Reinhardt A Botha

January 2011

Declaration

I, Geert Dirk Jurgens with student number 20517971, hereby declare that the dissertation for qualification, Magister Technologiae: Information Technology my own work and that it has not previously been submitted for assessment to another University or for another qualification.

_____

Geert Dirk Jurgens

6 January 2011

Abstract

The use of mobile phones and other mobile devices are becoming widespread and almost all of these mobile devices have some sort of mobile Internet access. Due to the increase in mobile Internet usage, many websites need altering in order to become mobile compatible. Creating a mobile compatible version of a website is challenging due to formatting and capability restrictions imposed by the mobile device.

Currently, one of the popular methods of creating a mobile compatible website involves the creation of a new, dedicated mobile version of the website. However, this approach can prove to be expensive, and repetitive, since a fully functional desktop version of the website often already exists. A second method involves the use of a transformation proxy to transform the traditional website into a mobile compatible version.

This research develops a transformation framework that enables a web developer to create a single set of source files that can be used to render output compatible with both traditional and mobile devices. In developing this framework, capabilities and restrictions of the mobile device were examined. Furthermore, current mobile web development guidelines and best practices were discussed. This resulted in the development of a method to identify and outline areas of a traditional website for transformation into a mobile friendly format. Furthermore, a transformation engine that allowed processing of the traditional website into a mobile compatible website was developed. This transformation engine extracted the outlined areas, and rendered the extracted content, all while maintaining the website's original functionality.

The development of a prototype verified that the transformation concepts are valid, and provided for the development of guidelines and recommendations.

The development of a framework to enable the web developer to create a website once, and enable it to adapt its output for mobile devices, will have a positive impact on the development of content for the mobile web.

Acknowledgements

It is a pleasure to thank those who made this research possible:

My supervisor, Professor Reinhardt Botha for his support, encouragement, and for going the extra mile, without which this work would not have been possible.

My parents, Geert and Pam Jurgens for their unwavering support, motivation and patience.

My sister, Jenette Jurgens for her motivation and assistance.

The lectures in school of ICT at NMMU for their input.

Table of Content

List of Figures

List of Tables

Chapter 1. Introduction

Mobile devices are becoming more commonplace, and as a result, mobile Internet access is becoming a reality for many users. A recent survey showed that most 1827 year olds use their mobile phone to access the Internet more often than traditional devices, such as laptops and desktops (Opera Software, 2010).
However, despite their wide use, mobile devices are restricted in their capabilities to display webpages primarily designed for the traditional desktop browsers.

These restrictions are mainly due to the limitations imposed by mobile devices on screen size, input capabilities and processor power (Passani, 2010). The use of GSM and GPRS as mobile data transport bearers and the technical limitations imposed by these technologies create further challenges for web developers.
(Talukder & Das, 2010).

In order for web developers to make a mobile compatible website, they can follow a variety of approaches. One common approach used by web developers, is the creation of completely new websites for the mobile device; however, this approach can be an expensive undertaking (Christos, Giorgos, & Ioannis, 2007). A second approach for creating mobile compatible websites is to transform the markup code on the server to a layout that is capable of being displayed on the mobile device. (Christos, Giorgos, & Ioannis, 2007). It is foreseen that this second approach can reduce the cost in developing websites for the mobile web. Thus, the potential cost saving resulting from this approach makes it an attractive alternative for web developers and content providers.

1.1 Problem statement

Due to the increased number of mobile device users accessing the Internet, many web developers need to create mobile compatible websites.

As discussed above, one of the common methods of developing content for the mobile web is to re-create an existing website, so that it will be compatible with mobile devices. This approach may result in additional cost, time and resources for the web developer. To avoid the recreation of an existing website, this research will investigate methods to enable a single website to adapt its output rendering to fit the requesting device.

1.2 Research question

In order to delineate between function and content, this research will define them as follows:

> **Function: Actions that cause processing or some type of response from the website, such as a submit button or navigation link.**
>
> **Content: Any information and data displayed on or by the web page, such as a newsfeed or body text.**

In order to implement a usable transformation of the functions and content of a traditional web application to the mobile web, this research will answer the following question:

> **"How can current mobile web guidelines, practices and technologies be implemented to transform the function and content of web applications developed for desktop browsers to fit the capabilities of mobile web browsers?"**

In order to answer the above question, the following sub-questions have arisen and need to be reviewed.

- **What are the capabilities of mobile phones compared to the desktop environment?**
- **How can desktop to mobile content transformation be accomplished?**

- **How can desktop web application function be transformed to the mobile web?**

If the above questions can be answered, it is believed that an environment would be created, whereby functionality and content on traditional web applications could be seamlessly transformed for mobile web use.

Therefore, we can state the following objectives of this research.

1.3 Research objectives

The main objective of this research is to:

> **"Develop a framework for content and functionality transformation of desktop based web applications to the mobile web."**

In order to achieve this objective, three distinct sub-objectives must be achieved.

- **Develop a set of guidelines for the selection of critical content and functionality that must be transformed from the traditional web application to the mobile web.**
- **Develop a set of guidelines for the implementation of the selected content in the mobile web.**
- **Develop a set of guidelines for the implementation of function in the mobile web.**

In a research context, these objectives can only be argued to be valid if a sound research methodology is followed.

1.4 Research methodology

In order to develop a framework for the transformation of function and content from desktop web browsers to mobile web browsers the following research methods were used.

A review of current mobile communication technologies, mobile operating systems and application software was conducted.

This was followed by a review of current mobile web capabilities, recommendations and guidelines. Furthermore, an investigation into current mobile web design and implementation techniques was conducted.

Based on the findings found during the review of current mobile web capabilities, a proposed solution and conceptual model were defined. This solution and model formed the basis for the development of a prototype.

The development of the prototype ran concurrently with the theoretical literature study of current guidelines, practices and technologies. This enabled the researcher to use aspects of the theoretical literature study that are relevant at a particular point in the prototype development. The development of the prototype used an incremental prototype model that improved in design and function with each increment. By following this approach, the researcher was able to test the prototype at various stages of development. The lessons learned during development of the prototypes contributed to the development of a framework for the practical implementation of current best practices, guidelines and technologies in a real world scenario.

The interplay between the prototype development, the theoretical study and the framework development is shown in Figure 1-1.

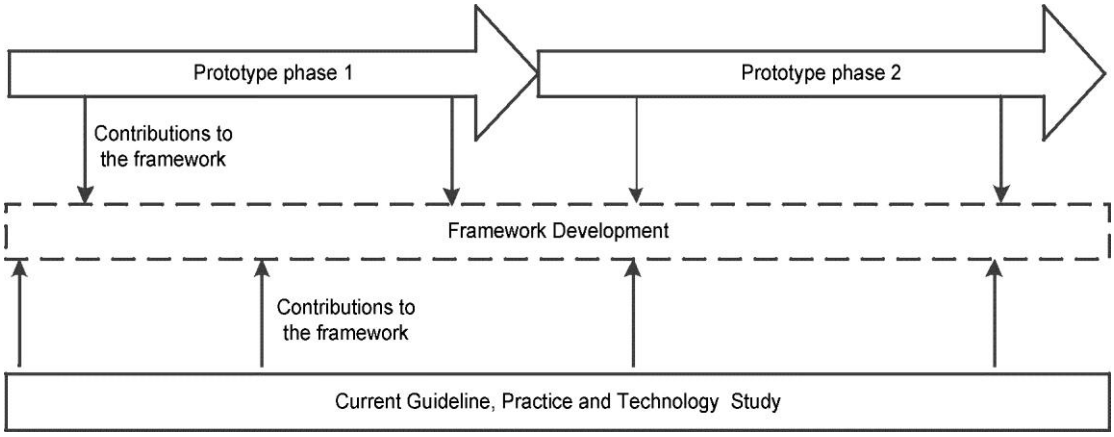


Figure 1-1 Research methodology

This research methodology is similar to the Design Science methodology described by Henver, Ram and March (2004) in that it produces similar outcomes. Henver, Ram and March (2004) described a set of guidelines that characterize "good" design science. These guidelines are as follows:

Guideline 1: Design as an Artifact

The research must produce an artifact in the form of a construct, model, method or instantiation.

Guideline 2: Problem Relevance

The objective of the research is to develop a technology-based solution to a relevant business problem.

Guideline 3: Design Evaluation

The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.

Guideline 4: Research Contributions

Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.

Guideline 5: Research Rigour

Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.

Guideline 6: Design as a Search Process

The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.

Guideline 7: Communication of Research

Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

1.5 Dissertation layout

The dissertation layout is structured as shown in Figure 1-2. Chapter 1 introduces the problem, provides an overview of the research question, objectives, research methodology, and sets the scene for the rest of the document.

Chapter 2 reports on the challenges faced when developing content for mobile devices. This report outlines the limitations and capabilities of mobile devices compared to desktop-based computers. The results of a study into the current mobile standards, best practices, guidelines, development and implementation used by mobile web developers are discussed in Chapter 3.

Chapter 4 proposes a conceptual foundation for the development of a transformation framework, which will assist in the transformation of traditional webpages to those that, can be utilized with mobile devices. Chapter 5 builds on the foundation laid out in Chapter 4 by developing experimental prototypes, which were validated with a small case study.

By reviewing the findings of Chapter 4 and Chapter 5, Chapter 6 formulates a framework for mobile transformation.

Finally, the conclusion chapter will summarize the findings and provide insight and recommendations for further study of this topic.

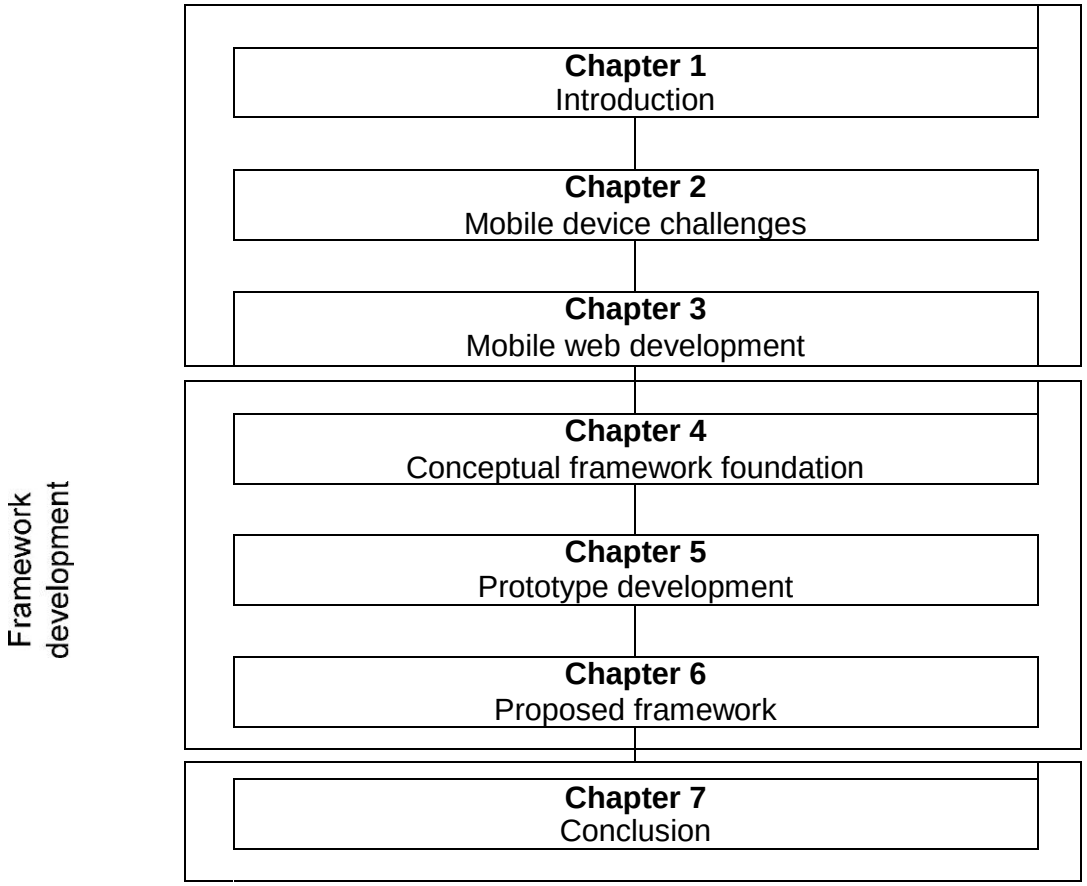


Figure 1-2 Chapter outline

Chapter 2. Mobile device challenges

Chapter 1 described an overview of this research with the goal of developing a framework for mobile transformation. Furthermore, it was discussed how currently separate websites are often developed for mobile devices, despite the existence of a fully developed website. Therefore, this research will investigate the approach of adapting a single website to display on both a mobile device and traditional desktop in hopes of reducing the development cost of webpages.

In order to understand why mobile devices are restricted in their ability to handle traditional websites, it is important to understand the challenges faced by users and developers alike. Furthermore, it is also important to understand how mobile device software and communication methods can present development challenges. Therefore, the rest of this chapter expands on the challenges of developing for mobile development.

2.1. Usability

The widespread use of mobile devices has resulted in users wishing to experience the Internet regardless of which device they are using (Paterno, 2003). This has resulted in challenges for web developers, since mobile devices **have unique operational characteristics that can directly affect the users experience (Parush & Yuviler-Gavish, 2004). Compounding the problems associated with the usability of the mobile web, is the desire to allow the user to access the same information and perform similar tasks to that of fixed, regular-size screen platforms, such as the desktop computer (Buchanan, Jones, Thimbelby, Farrant, & Pazzani, 2001).**

An increase in the development of dynamic webpages compared with the development of traditional static webpages persists (Tian, Voigt, Naumowicz, Ritter, & Schiller, 2007). This increase in the development of dynamic webpages correlates with more users accessing the dynamic content. To access this dynamic

content, users must input form data, such as name, surnames and other information. This can prove time consuming for users of mobile devices because of the limited text input capabilities (Rukzio, Noda, De Luca, Hamard, & Coskun, 2008). Many times this is often due to the restrictions imposed by the use of a mobile device keyboard. The keyboard on mobile devices is physically small and compact, which makes it difficult to locate a target key without pressing neighboring keys. For example, a typical keypad has twelve keys and there are normally three letters on each key. Therefore, a user has to distinguish different letters on the key by pressing the same key several times, which makes typing slow and error-prone (Chen, Yesilada, & Harper, 2010). In an attempt to improve text input on mobile devices, the T9 predictive text algorithm was developed. This algorithm reduces the need to press the same key multiple times to create a word (Nuance Communications, 2007).

Besides keyboard related challenges, input is further challenged by the common use of a joystick type interface for vertical and horizontal cursor movement (Chen, Yesilada, & Harper, 2010). This type of pointing interface on a mobile device is less efficient than a mouse on a traditional computer as it can only move a cursor in four directions. In addition, movement is incremental, so a user cannot "jump" from one position on a screen to another (Chen, Yesilada, & Harper, 2010).

In order to provide the same functionality available on traditional PC devices to mobile devices, content, such as documents, webpages and maps, are becoming readily available on small screen devices (Burigat, Chittaro, & Gabrelli, 2008). The common form factor of mobile devices constrains screen size to a small fraction of what is available on a desktop. For example, a typical 240 x 320 pixel display of a mobile device is less than one-sixteenth the display area of a typical 1280 x 1024 desktop display. Such limitations make it extremely difficult for users to navigate content that does not fit on the screen of their mobile devices (Burigat, Chittaro, & Gabrelli, 2008).

These and other challenges greatly affect the usability of traditional websites when they are accessed using a mobile device. Besides the usability challenges that mobile devices present, mobile devices also exhibit technical limitations, which can affect how a website is rendered. These technical challenges will be discussed in the next section.

2.2 Technical challenges

The technological capability of mobile phones is advancing at a rapid pace. Similarly, the software that powers them follows suit and becomes more powerful and more capable. However, the capability of the mobile devices is still greatly dependent on the type of mobile device used.

Mobile phones can be separated into two categories, smartphones and low-end cellular devices (LECD). LECD's are primarily used for voice communication, but have limited processing capabilities. Smartphones provide voice communications as well as powerful processers for running advanced applications. Smartphones often make use of sophisticated mobile web browsers, for running web applications. (Willemse, 2009, p.3). Smartphone use grew 96% in the third quarter of 2010 compared to the same quarter in 2009, which resulted in smartphone sales making up 19.3% of mobile phone sales in the third quarter of 2010 (Gartner, 2010). The strong growth in smartphone sales is driving the development of mobile phone operating systems and subsequent applications. Since smartphones are more capable than LECD's to access advanced web content, it is important to understand the software and hardware powering these devices. Furthermore, the limitations and capabilities of the software and hardware need to be understood by web developers in order to develop suitable applications for use on traditional computers as well as smartphones.

2.2.1 Mobile operating systems

An increase in user demands on mobile devices is resulting in the development of mobile devices that are similar in functionality to traditional computers. Therefore, the development of mobile device versions of popular desktop-based operating systems is becoming commonplace.

Google Android - The Android operating system (OS) is a popular open source mobile device operating system. The open source nature of Android has resulted in its adoption by many mobile devices. Thus, Android was the second most popular mobile device operating system during the third quarter of 2010 (Gartner, 2010). Android evolved from the Linux 2.6 Kernel, this kernel provides core system services and acts as an abstraction between the hardware and software (Android, 2010). As the Linux operating system was originally developed for traditional computers, Android is a good example of how parts of a traditional OS can be adapted for mobile use.

The Android platform provides the software stack for mobile devices, which includes an operating system, middleware and key applications. The Android platform is developed with a Software Development Kit (SDK) and allows developers to create applications that can run on the Android platform (Android, 2010). By leveraging the power of the Android SDK, developers have been able to create applications that can run on a variety of Android powered devices. While Android is popular, there are many limitations and bugs that have been encountered with the Android Platform (Google Code, 2010). While these limitations will likely be rectified in future releases, content developed for or delivered to these devices may encounter functionality problems. It is important to understand that some of these problems may be exclusive to Android powered devices. This may result in content working correctly on another operating system, but fail when run on Android. While content functionality problems occur with many operating systems, the popularity of

Android powered devices requires developers to pay special attention to optimising their content for Android.

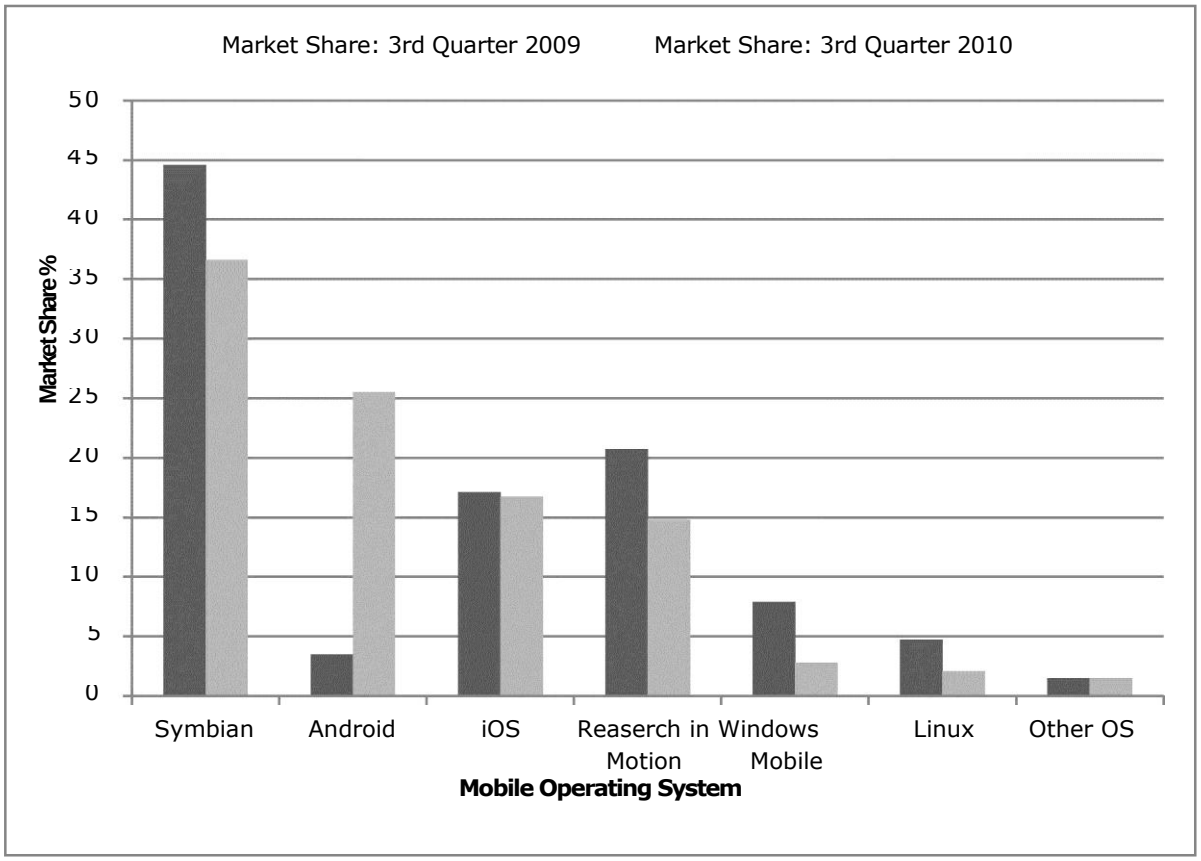


Figure 2-1 Chart showing mobile operating system market share.

Figure 2-1 shows a comparison of the operating system market share between the third quarter of 2009 and the third quarter of 2010 (Gartner, 2010). It is interesting to note the increase in the adoption of Android powered devices between 2009 and 2010. This increase shows the growing popularity of Android powered devices. Therefore, it is important for web developers to develop applications to be compatible with Android.

Blackberry OS - The Blackberry OS is the propriety operating system developed by Research in Motion (RIM) for the Blackberry line of smartphones.

The popularity of these devices has resulted in RIM becoming rated as one of the top five mobile device manufacturers worldwide. This has enabled the Blackberry OS to become the fourth most popular mobile OS worldwide (Gartner, 2010). The propriety nature of the Blackberry OS, limits the use of this OS to RIM devices. This limitation has resulted in other manufactures of mobile devices adopting the open source Android OS, thereby increasing the device coverage of Android. Like Android and other mobile operating systems, Blackberry provides a SDK for developers to develop applications for the Blackberry smartphone, (Blackberry, n.d). Since these applications are developed exclusively for the Blackberry device, they are optimised for these devices. The Blackberry line of mobile devices is primarily targeted towards the corporate market, and therefore has tight integration with corporate systems, such as Microsoft Exchange server, which provides "Push" email functionality among other communication capabilities (A J Gold Associates, 2008). This has resulted in the Blackberry becoming popular among business users. Developers interested in creating business orientated applications need to take Blackberry device limitations and capabilities into account.

Apple iOS- Apple iOS is the propriety operating system developed by Apple for the iPhone and iPad mobile devices. Like RIM devices, the propriety nature of this OS limits the use of this OS to Apple created devices. However, the popularity of the iPhone, and more recently the iPad, has resulted in iOS becoming the third most popular operating system worldwide (Gartner, 2010). Like Android and RIM, Apple provides a SDK for third party developers to develop applications for these devices (Apple inc, n.d). Apple iOS is shipped with a number of default applications, such as a web browser and email client. As the Apple iOS is propriety, it has been developed to take advantage of the multi-touch capabilities and other features of the mobile device (Apple inc, n.d).

However, the limitations of iOS with Flash content support can present difficulties to developers when creating web content.

Symbian - Symbian is the world's most popular mobile phone platform, and is used in a wide variety of mobile devices. Symbian is flexible and able to be used with simple mobile devices to high end smartphones (Nokia, n.d). This flexibility and high adoption rates have resulted in Symbian powered devices having a 36.6% market share in the third quarter of 2010 (Gartner, 2010). Like the other mobile OS's discussed above, Nokia has released a SDK for the Symbian platform allowing developers to develop applications for mobile devices. Developers creating applications for the Symbian platform have extensive support for audio and visual capabilities and an extensive range of open application programing interfaces in C, C++ and JAVATM languages (Nokia, n.d). The use of Symbian on both high end phones and simple devices presents a challenge to developers. Therefore, when developing Symbian based applications, developers must take the target device capabilities into account. As a result, many different versions of the same application targeted to different phone capabilities may need to be developed.

Windows Mobile - Windows Mobile is a mobile operating system developed by Microsoft for mobile devices, which is based on the Windows CE kernel and aimed at use with personal desktop assistants (PDA) and mobile phones (MSDN, 2010a). Recently, Windows Mobile was re-developed and re-launched as the Windows Phone 7 Series. This operating system is anticipated to replace the Windows Mobile brand in the future (Koh, 2010). Mobile devices running Windows Mobile can account for a 2.8% market share in the third quarter of 2010 (Gartner, 2010). While not as popular as the other mobile operating systems discussed above, its tight integration with other Microsoft products makes it important for developers to take this OS into account. As with other mobile operating systems, Windows Mobile and Windows Phone 7 Series

provide a SDK for developers to create their own applications for mobile devices.

Table 2-1 shows a comparison of basic operating functionality between the iOS 4, Android 2.2 and the Windows Phone 7.

Table 2-1 Comparison of mobile operating systems (German & Cha, 2010)

	Apple iOS 4	Google Android 2.2	Windows Phone 7
Consumer availability	June 21 2210	Will vary by device	Fourth quarter of 2010
Multitasking	Yes, only iPhone 3GS and iPhone 4	Yes	Yes, but limited
Notifications	Yes, one at a time	Yes, multiple notification bar	Yes
App folders	Yes	Yes	Apps arranged in 'hubs'
Tethering	Yes	Yes, built in support	Unknown
VoIP	Yes	Yes	Unknown
Gaming	Many available	Many	Xbox Live integration
Flash support	No	Yes	Yes, but not at launch
Number of titles in App Store	iTunes App Store: more than 225 000	Android Market: more than 30 000	Windows Phone Marketplace: not launched yet
Voice commands	Yes	Yes	Unknown
Mass storage	No	Yes	Unknown
Universal search	Yes	Yes	Yes
Music store	iTunes Music Store	Amazon MP3	Zune
Web browser	Safari	Android	Internet Explorer
Copy-Paste	Yes	Yes	No
Pinch and zoom multi touch	Yes	Yes	Yes
Video Calling	iPhone 4 only	Yes, depending on model	Not yet

2.2.2 Web browsers

Many mobile operating systems provide a SDK for developing applications for use on a particular platform. These SDKs enable programmers to leverage the capabilities and work within limits imposed by the mobile OS and device. Mobile web browsers are of particular interest as they render websites or web applications that are designed with no particular operating system in mind.

Web browsers become the user's method of interaction to a website. Therefore, if a web browser fails to render a page correctly, usability of the website may be compromised. It is important to look at technologies that directly affect the display and functionality of the website and browser. Similar to mobile phone operating systems, mobile web browsers use components of web browsers developed for traditional desktop-based computers. One of the major components of a web browser is the rendering or layout engine. Browsers' rendering engines or layout engines are algorithms that process the HTML and other mark-up code from a requested website and renders or "paints" the user's browser window with content. This is done according to an interpretation of standards, best practices and rules. Three common rendering engines are discussed below.

Gecko - Gecko is an open source layout engine developed by the Mozilla project and is therefore used in applications, such as the Mozilla Firefox web browser and Thunderbird email client. Gecko is designed to support open Internet standards such as HTML 4, CSS (version 1 and 2), W3C DOM, XML and JavaScript. Due to the efficiency and speed of the Gecko engine, it is used to create the user interface of some applications, such as the scroll bar (Mozilla Foundation, 2010). Gecko is made of many different components that each have their own function and work together to generate a web page. According to the Mozilla Foundation (2010), the Gecko engine is made up of the components shown in Table 2-2. The open source nature of the Gecko engine enables constant development and improvement of the code. Furthermore, it allows any developer to take advantage of the Gecko engine and incorporate it into their own applications.

Trident - Trident, also known as MSHTML, is a rendering engine similar to Gecko, developed by Microsoft for the Internet Explorer browser.

Trident has come under criticism for its lack of global standards support, resulting in rendering and layout issues with some web pages. Microsoft has committed to improving support standards in future versions of Trident (Microsoft Cor peration, 2008). The Trident layout engine is used by the Internet Explorer mobile browser and included in the Windows Mobile series of operating systems for use with mobile devices.

Table 2-2 Components of Gecko engine

Document parser - handles HTML and XML **Layout engine with content model** **Style system - handles CSS** **JavaScript runtime - using the SpiderMonkey engine** **Networking library - using the Necko engine**
Platform-specific graphics rendering and widget sets for Win32, X, and Mac
User preferences library **Mozilla Plug-in API (NPAPI) to support the Navigator plug-in interface**
Open Java Interface (OJI), with Sun Java 1.2 JVM
Security library (NSS)

Webkit - Webkit is an open source rendering engine, similar to the Gecko engine discussed above. Webkit is popular in that it is used as the rendering engine for many Apple Mac OS X developed products, such as the Safari Web browser, Dashboard and Mail (Webkit.org, n.d a). During 2005, Nokia ported the Webkit rendering engine to their Symbian S60 platform, which resulted in the development of the "S60 Webkit project". The use of Webkit as the rendering engine for the S60 web browser resulted in every S60 mobile device containing the Webkit rendering engine (Webkit.org, n.d b).

Furthering the adoption of the Webkit engine for mobile device browsers, it was announced at the 2010 Mobile World Congress that the Blackberry OS 6 will make use of the Webkit rendering engine for use with its mobile web browser. This will enable the Blackberry browser to render pages faster and make full use of technologies, such as JavaScript and Ajax (Kirkup, 2010).

2.2.3 Functionality expansion

Web browsers provide the rendering of content and basic functionality for a website. Functionality can be determined as the interactivity a user has with the site. For example, when a hyperlink or button is clicked, an action must occur. This action can range from loading a new webpage to performing server side processing. The functionality provided by a website can further be enhanced when developers use technologies, such as JavaScript and Adobe Flash, to provide dynamic user interaction with the webpage.

JavaScript – JavaScript is used by web developers to expand and enhance webpages by allowing for dynamic elements, such as partial page updates, popup dialog boxes and other user effects. Due to the limited space and capabilities of some mobile devices, JavaScript is not always supported or has limited functionality in the mobile device browsers.

Flash - Adobe Flash Player is a cross-platform that allows the web developers to create rich interactive applications for a website (Adobe, n.d). Adobe Flash Player is used to provide content for websites, such as YouTube and other multimedia sites. In order for the web browser to display Flash content, the Adobe Flash Player plugin must be installed. The use of Adobe Flash on mobile devices is slowly becoming possible; however, Apple Inc. has stated that Apple mobile products, such as the iPhone and iPad, will not support Adobe Flash. This is mainly due to the fact that Apple is concerned about the propriety nature and apparent stability issues of the Flash Player (Bilton, 2010). The lack

of Flash support on some of the major mobile platforms can present design and implementation challenges to the web developer. Table 2-3 shows a functionality comparison between popular mobile web browsers.

Table 2-3 Mobile browser comparison chart- (GSM Arena , 2010)

	Mobile Browser					
Functionality	iOS (iPhone 3G)	IE Mobile 6	Opera Mobile 9.7	Android 1.5	Android 1.6	Android 2.0
Rendering Engine	Webkit	Trident	Presto	Webkit	Webkit	Webkit
Page Loading Speed / sec	5.22	9.36	14.3	18.17	13	11.06
JS Benchmark score	30	17	12	16	12	26
Acid 3 Test Score	98/100	5/100	100/100	93/100	93/100	93/100
JavaScript	yes	yes	yes (some issues)	yes	yes	yes
Flash	no	yes (partial)	no	no	no	no
Full screen Support	no	yes	yes	yes	yes	yes
Landscape Support	yes	yes	yes	yes	yes	yes
Auto Rotate	yes	yes	yes	yes	yes	yes
Text Reflow (fit in screen)	no	yes	yes	yes	yes	yes
Address Suggestions	URL Only	URL Only	URL Only	URL Only	URL Only	URL Only
Password Manager	yes	no	yes	yes	yes	yes
Auto fill form fields	No	no	no	yes	yes	yes
Back Button	one step	one step	one step	one step	one step	one step
Multiple Pages	yes	no	yes	yes	yes	yes
Open page in new window	yes	no	yes	yes	yes	yes
Copy Text	yes	yes	yes	yes	yes	yes
Download Manager	no	no	yes	yes	yes	yes
Popup-Blocker	yes	n/a	yes	yes	yes	yes

2.3 Mobile communication

As discussed in the previous sections, mobile device operating systems and mobile browsers have limitations on functionality.

This results in mobile application developers having to take these limitations and capabilities into account when developing their applications.

The methods that the mobile device communicates with the mobile service provider, and therefore the Internet, have their own set of limitations. These limitations are of particular importance to developers of mobile web pages as they can affect factors, such as loading time and data throughput. Mobile devices make use of various wireless technologies to provide connectivity solutions. It is important to understand that many mobile devices provide connectivity to both the mobile network providers' network and to a Local Area Network (LAN). Some of these technologies will be discussed in the following section.

2.3.1 Mobile network data technologies

Mobile network providers are currently making use of the following technologies to provide mobile connectivity using their mobile network infrastructure. Dependent on the technology used, the transmission speed of data can affect the ability of websites to be rendered with sufficient speed.

GPRS Networks - GPRS (General Packet Radio Service) makes use of the existing GSM radio infrastructure to provide packet based data transmission. By utilising this technology, it allows the network to process both data and call services at the same time. The use of packet switching technology allows the resources to only be used while transmitting data. This has the benefit of not tying up resources by providing a dedicated resource for each user (Willemse, 2009, p. 40). While the GPRS does provide access to data connections, this GPRS technology is slow.

EDGE Networks - EDGE (Enhanced Data rates for GSM Evolution) is an enhancement to the GPRS technology. By changing the frequency modulation of the radio signal, a faster speed is achieved when compared to that provided

by GPRS (Willemse, 2009, p. 40). EDGE is marketed as a "bolt on" technology that can be used on any existing GPRS network. This makes it easier for existing network operators to upgrade to this faster technology.

3rd Generation networks (3G)- 3rd Generation networks (3G) is an advancement in mobile network technology that results in significantly higher data rates and capacity than the older second generation or 2G networks (Willemse, 2009). 3G is the common term given to a set of technologies that make up a 3rd generation network. These technologies are briefly described below (Asif, 2007).

CDMA 2000 is a set of standards that govern and define the radio interface that makes use of CDMA technology to accomplish a 3G network.

WCDMA (Wideband Code-Division Multiple Access) is a technology that allows single users to make use of multiple channels within a signal range to achieve data rates of over 2Mbps.

HSDPA (High Speed Download Packet Access) evolved from the WCDMA technology and makes use of adaptive modulation and coding techniques to achieve a faster throughput with reduced delay. Peak data rates of 14.4Mbps can be achieved using this technology.

Third generation networks are becoming commonplace and development of the fourth generation mobile specification is underway. These advancements in mobile data networks are enabling more content to be available to the user, which is driving the rapid development of Internet dependent mobile applications and websites.

2.3.2 LAN technologies

Mobile devices are becoming increasingly more advanced. This has resulted in the ability of mobile devices to connect to home and office networks. The ability of a

mobile phone to connect to a Local Area Network (LAN) provides access to documents and other applications. Furthermore, Internet connections can often be established over a LAN connection and therefore provide an additional connectivity method for the mobile device. Mobile devices commonly make use of Wi-Fi network technology to accomplish this. Wi-Fi technologies will be briefly discussed below.

Wireless Local Area Networks - Wireless local Area networks (WLAN) are a common method to connect devices, such as laptops, mobile phones and other devices to a home or office network. The use of WLAN networks enables the user to move around the coverage area, without losing network connection. The development of WLAN capabilities on mobile phones has enabled high-speed access to corporate networks, and Internet connections without the need to connect to the mobile network provider's data network. This can have significant cost advantages for the user.

The ability of mobile phones to connect to WLAN networks has resulted in the development of specific technologies for taking advantage of the connectivity and cost benefits that this connection can provide. Software, such as Fring, allows the user to make and receive voice over IP VOIP calls for little or no cost from their mobile phone, thus bypassing the mobile network provider (Fring, 2007).

LAN connectivity has greatly improved the capabilities of the mobile device to connect to other nearby devices and networks without being reliant on the mobile network provider's communications network.

2.4 Conclusion

The rapid progression of mobile technology and capabilities, combined with the increased ability of mobile networks to provide high-speed data to mobile devices, is resulting in the development of more demanding applications.

Because of the increased capabilities, mobile device users are starting to demand that applications they use on traditional computers also be available on their mobile device. This is evident by programs, such as word processors, Skype and many others, being developed for mobile devices. The wide range of mobile operating systems can result in developers needing to create multiple versions of the same mobile application for each operating system. Furthermore, the features and instability of each mobile operating system has to be addressed. The support and development of each version of the software can result in increased cost, and is often not a viable option; therefore, web based applications for providing content and functionality though a device's web browser is proving to be a viable alternative. However, the challenge for web based application developers is going to be the successful porting of traditional web applications to mobile devices, while maintaining equal functionality and content as possible.

The following chapters will look at the development of a possible solution to this challenge. In order to gain a better understanding of how the mobile device interprets the web, a review of previous work will be presented in the next chapter.

Chapter 3. Mobile web development

The previous chapter discussed the limitations and capabilities of the mobile device. It was further discussed how the transformation of function and content to the mobile web is a challenge. In order to provide a better understanding of current mobile web development, this chapter will look at the current guidelines, best practices and methods employed.

Web development for both traditional and mobile web applications can be divided into several developmental stages, as shown in Figure 3.1. In each stage, there are guidelines, practices and methods that can influence its development.

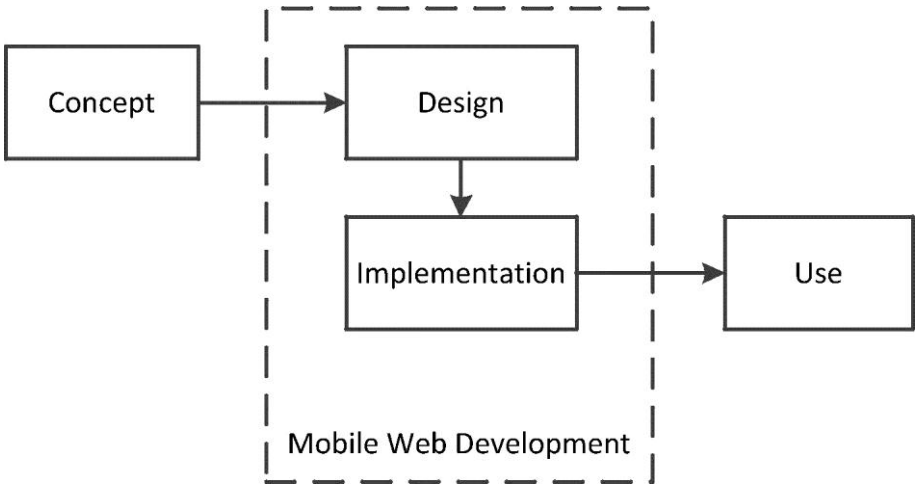


Figure 3-1 Stage layout

During the concept stage, an idea and the functionality requirements are decided upon. The design stage involves development of the web application to meet the requirements put forward in the concept stage.

The implementation stage involves how the application is put into operation. Finally, following successful implementation, the use stage is when users begin to utilise the application.

While the concept and use stages are similar between traditional web development and mobile development, the design and implementation stages in mobile development are influenced by mobile specific challenges.

These challenges have resulted in the development of specific mobile guidelines, best practices and methods. The design phase and the implementation stage are discussed below.

3.1 Design stage

During the design stage, various aspects need to be considered in order to develop an effective mobile site. There are many design guidelines and best practices available for the developer to follow in order to make the best use of the mobile device. In an attempt to encourage consistent development and compatibility across mobile devices, the World Wide Web Consortium (W3C) developed a set of best practices for mobile web development.

The World Wide Web Consortium (W3C) is an international community that produces guidelines, standards and best practices for the World Wide Web (W3C, 2008). The guidelines, standards and best practices, put forward by the W3C are highly regarded. They play an integral part in the operation and development of the Internet and its components.

As discussed above, the W3C has put forward a set of best practices that provide a foundation for the development of mobile compatible websites. These best practices are set out in the W3C Best Practices Document 1.0 (2008) and discussed in this section.

The W3C Best Practices Document 1.0 (2008) section 3.1, discusses the concept of the "One Web" approach to the design of mobile web sites. According to the W3C Best Practices Document 1.0 (2008) section 3.1, the "One Web" approach entails making as far as is reasonable, the same information and services available to users, irrespective of the device that they are using.

The W3C Best Practices 1.0 (2008) "One Web" approach to development points out that it is important to consider that website content should be available and accessible regardless of browser platforms used by the end users. However, the W3C Best Practices 1.0 (2008) section 3.1, also states that it is important to note that the "One Web" approach to web design does not mean that the same information has to be available in exactly the same way across all devices. This is largely due to constraining factors, such as type of device and type of network, which will affect the representation of the content.

It is further noted in W3C Best Practices 1.0 (2008) section 3.1, that some services and information are targeted at a specific group of users. Therefore, it is important for application developers and service providers to strive for the best user experience of the designed application. An extract of some of the best practices listed in W3C Best Practices 1.0 (2008) section 5, are described Table 3-1.

During design of the mobile website, it is important for the developer to consider factors that affect the functionality of the website. The W3C Best Practices 1.0 (2008) document highlights these factors and provides a best practice suggestion for the developer to take into account. These aspects are discussed in the following sections.

Table 3-1 Best practices for mobile web development.

Item	Description
Thematic consistency	All contents provided by accessing a mobile site must provide a thematically coherent experience when accessed from multiple mobile devices.
Capabilities	Individual mobile device capabilities must be exploited to provide an enhanced user experience when using the site.
Deficiencies	Take reasonable steps to work around deficient implementations.
Testing	Testing must be done on actual devices as well as mobile device simulators.
URL	All the URLs of the site entry points must be kept sort.
NAVBAR	Provide only minimal navigation at the top of the page.
Balance	A balance must be taken into account to determine whether a trade-off between having too many links on a page and asking the user to follow to many links to reach what they are looking for.
Access Keys	Assign keys to functions that are access frequently.
Image Maps	Do not support image maps unless you know the device supports them effectively
POP UPS	Do not cause popups or other windows to appear.
Auto Refresh	Do not use auto-refreshing pages, unless you have informed the user, and have a means of stopping it.

The lack of screen space on mobile devices can present challenges to the design of mobile webpages. Therefore, it is important for the developer to create a website that is presented well on a mobile website. The W3C Best Practices 1.0 (2008) document recognizes the importance of good presentation and provides guidelines on how to achieve good presentation; these are discussed below.

Consistency of style - The ability of the website to maintain consistency of style throughout the website needs to be considered. The W3C Best Practices 1.0 (2008) section 5 encourages a consistent style to be used across the mobile site, as this will aid the user in creating a mental image of the site's layout.

Scrolling – As discussed above, mobile devices are limited in screen size and this limiting factor needs to be taken into account during design. As a result of the limited screen size, a mobile webpage can require considerable scrolling by the user. The W3C Best Practices 1.0 (2008) section 5, reminds developers to take this aspect into consideration during design of the mobile website.

Action Feedback - the ability of a website to provide user feedback is an important aspect to consider during design. Providing sufficient user feedback on a mobile device from a mobile website is a challenge. W3C Best Practices 1.0 (2008) section 5, notes this important aspect and makes a recommendation on how a developer can structure the site. Special attention is paid to the position of navigation and the use of page refresh as a form of user feedback.

Identification of the user goals - During the design of the mobile website, it is important to identify what the user is intending to achieve by using the site. W3C Best Practices 1.0 (2008) section 2.4 discusses this aspect and presents some guidelines that developers can follow in order to identify this goal. The document identifies that mobile users typically have different goals compared to desktop based users. Furthermore, it is noted that mobile users tend to have a more immediate goal directed intention. The intentions of most mobile users are often to find specific pieces of information related to their current content. It has been found that mobile users are typically less interested in lengthy documents and more interested in immediate results. Developers need to take this into account when developing mobile applications and websites, since usability will be affected if the application does not provide what the user requires.

Choice of user experience - Following on the "One Web" approach to website design, it is important to understand that the website content must be similar regardless of the type of device used.

This is an important consideration for web designers to take into account during the design phase. The W3C Best Practices 1.0 (2008) section 3.6 recommends that the design of the website should incorporate a method to identify the device type being used. The website should be able to determine the devices capability's accordingly. While this is a consideration that must be taken into account during the design phase, it is also discussed in the implementation section discussed below.

Cascading Style sheets (CSS) - provides style information to the browser on how to render design aspects of the site. CSS provides the ability to apply different style sheets for different media, such as handheld devices, voice agents and others. A single webpage can have multiple style sheets defined to allow effective display on various media devices (Ragget, Le Hors, & Jacobs, 1999). This can prove useful to developers of mobile websites as CSS provides support for mobile devices though the *@media="handheld"* attribute. This allows the mobile web developer to define special design aspects specifically for a mobile device. However, the use of the mobile CSS still requires the web developer to spend resources creating a new style sheet for the mobile device.

In order to design a mobile website that is both usable and appealing to the users is a constant challenge. However, by following the design best practices set forth in the W3C Best Practices 1.0 (2008) document as a starting point the developer can create an effective, well-presented mobile site. Furthermore, these design best practices can be incorporated into the design of a transformation engine. Enabling the transformed mobile website to display and function correctly on a mobile device.

3.2 Implementation stage

The previous section discussed design considerations that need to be taken into account when developing for the mobile web. Following on the design aspects discussed above, this section will discuss best practices, considerations and techniques used during mobile website implementation.

3.2.1 Mobile data network consideration

Chapter 2.3 discussed the types of network technologies mobile devices use to access the Internet. It is important for the web developer to take note of how the mobile devices are using the mobile data network to run their website. The correct use of the mobile network can play a large role in the usability of the website. The W3C Best Practices 1.0 (2008) document provides some network considerations that the web developer must take into account when implementing the mobile web. Some of these aspects are discussed below.

Speed - Mobile Data Networks can be slow when compared with fixed line data connections and often have a higher latency. This can lead to long page load times, and it is further affected by pages that require a lot of navigation to reach the requested content. Therefore, developers must determine methods for optimising the page load times.

Cost - The cost of mobile data transfer is a factor that developers must take into account. Mobile sites must have optimised content to reduce page size and therefore attempt to reduce the data transfer cost. It is also important to note that mobile devices frequently only support limited types of content. Therefore, it is important to only deliver content to the device that can be interpreted by the device. Delivering content to the site that the device cannot read can result in wasted data cost.

Format – It is important for developers to take note of the data format that mobile devices can interpret. Incompatible formats can directly influence the user experience of the site, and can often result in an unsatisfactory experience.

Content- Developers must take note of the type of content that is provided by the website. Often the webpage may contain content that the user did not request, such as advertising, large images and so forth. This can directly influence the usability of the mobile site and can often result in a bad user experience or perception of the site. During implementation of the website, developers need to look at various methods of reducing this unrequested or unnecessary content in order to optimise the user's experience on the mobile device.

3.2.2 Mobile input

As discussed in Chapter 2, the input of mobile devices is often difficult when compared to the input of desktop devices equipped with a mouse and keyboard. This aspect is addressed by the W3C Best Practices 1.0 (2008), and point out important problems that can be encountered by the user. While this aspect can be classified as a design factor, it can be argued that it also plays an important role as an implementation factor. This can be attributed to the influence of server function and device browser function. An extraction of these points discussed in the W3C Best Practices 1.0 (2008) are reviewed below.

Address- During implementation, it is important that page and file names are short, and consideration must be made if the server makes use of URL rewriting or other functions. This is an important consideration since URLS are difficult to type on a mobile device.

Input- The limitations of mobile device screens can present a problem with form input and navigation order of the form fields.

This must be taken into consideration as devices might have different methods for inputting form data. Furthermore, the implementation of form validation may be affected depending on the mobile device's browser rendering engine.

Navigation- It is important to consider that many mobile devices do not provide back navigation functionality. The lack of this functionality can make it hard to recover from errors, broken links and so forth. This must be taken into account during both the design phase and the implementation phase. The design phase needs to pay attention to how errors are going to be handled by the site; this must spill over into the implementation phase where appropriate server configuration is done.

The designs of traditional websites do not take into account the restrictions of the mobile device. Therefore, in order for the traditional website to be transformed to work on a mobile device, these aspects need to be considered.

A user may access the website though a desktop or mobile format. The selected device should be able to identify what type of device it is and any special features or capabilities the application can use. The W3C Best Practices 1.0 (2008) section 3.7 recommends that if the application is not able to identify the requesting device, a standard set of rules should be followed. This set of standard global rules is known as the Default Delivery Context (DDC) and will be discussed in the next section.

3.2.3 Delivery context

As discussed above, if a device is unable to identify the requesting device then a Default Delivery Context (DDC) should be used. This DDC should define a set of global rules for site functionality.

W3C Best Practices 1.0 (2008) document defines a DDC with elements that should be consistent across service providers, in order to provide a baseline experience for users without making use of device specific adaption. The required elements that a default DDC should contain as recommended by the W3C Best Practices 1.0 (2008) document are shown in Table 3-2.

Table 3-2 Default Delivery Context

Attribute	Description
Screen width:	120 pixels minimum.
Mark-up Language support:	XHTML Basic 1.1 delivered with content - type application/ xhtml + xml.
Character encoding:	UTF-8
Image format support:	JPEG, GIF 89a
Maximum total page weight:	20 Kilobytes
Colors	256 colors
Style Sheet Support:	CSS Level 1. CSS Level 2
HTTP:	HTTP 1.0 or the more recent HTTP 1.1
Script:	No support for client side script.

3.2.4 Device capabilities identification

As discussed, it is possible to design the mobile website to adapt to the capabilities of the mobile device. Therefore, it is important for the website to identify the device and possible capabilities. Various methods used by web developers to identify the mobile device and determine its capabilities are discussed below.

HTTP Headers - HTTP Headers are sent during HTTP requests and responses. They are used by devices and programs to transmit their operating parameters.

This capability is taken advantage of by web developers and is one of the more common methods of identifying mobile device capabilities to the requesting application (Mahmoud, 2004). When a browser (User-Agent) requests a resource from a webserver, an identification string is provided. This string is read by the webserver and the content can be created accordingly.

Figure 3-2 shows an example of User-Agent request header string. In this example, the requesting browser identifies itself as a Mozilla browser and shows it's compatibly information.

```
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows
98; YComp 5.0.0.0)
```

Figure 3-2 User-Agent Request Header (Mahmoud, 2004)

The user agent request header is further supplemented by a series of accept headers. These HTTP headers define the acceptable types of content that the device or user agent can support. The providing server is able to read this accept header and provide supported content to the device or user-agent. According to Mahmoud (2004), HTTP 1.1 provides support for the following four data fields shown in Table 3-3. Furthermore, Figure 3-3 shows an example of how the Accept-Headers are used in the HTTP Request.

Table 3-3 Accept-Header fields

Attribute	Description
Accept:	**The MIME (media) types accepted by the User-Agent**
Accept-Charset:	**The preferred character set**
Accept-Encoding:	**The encoding for the User-Agent**
Accept-Language:	**The language set used**

```
GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
  application/vnd.ms-powerpoint, application/vnd.ms-excel,
  application/msword, */*
Accept-Language: en-ca
Accept-Encoding: gzip, deflate
```

Figure 3-3 HTTP Request Accept Header (Mahmoud, 2004)

As the mobile device becomes more feature rich, the capabilities of HTTP headers to transmit device information are becoming limited. This results in the HTTP header not being able to supply full or correct browser identification and capability information (Mahmoud, 2004). The limitations imposed by these current methods have resulted in the development of the CC/PP (Composite Capabilities / Preference Profiles) structure discussed below.

Composite Capabilities / Preference Profiles structure - In order to overcome the limitations associated with using Request/Accept header methods to identify browser capabilities, the W3C has developed a framework for content adaption and contextualization. This framework has resulted in the development and use of the CC/PP (Composite Capabilities / Preference Profiles) structure (Klyne, et al., 2004). The CC/PP provides a description of a devices capabilities and user preferences and can be used to guide the adoption of content presented to the requesting device (Klyne, et al., 2004).

In order to take advantage of the CC/PP, Mahmoud (2004) states that a developer should design the mobile content once, then use tools to customize and deliver the content in formats that specific mobile devices support. As a result of this development approach, users can access the same content from any device and be confident that will work on that device.

The objective of the CC/PP framework is to overcome the limitations imposed by the Request / Accept header methods by allowing the application to use the framework to discover information about the requesting mobile device (Mahmoud, 2004). The CC/PP framework is based on the RDF (Resource Description Framework), which is a W3C recommendation for describing web resource metadata (Klyne, et al., 2004). Figure 3-4 shows an extract of an example of a CC/PP profile for a Blackberry 7750 mobile device. The annotations shown in Figure 3-4 are described as follows:

(a) Identifies that the information below is related to the devices hardware capabilities.

(b) Defines the bits per pixel supported by the device.

(c) Defines that this device is capable of displaying colour.

(d) Defines the type of CPU in the device, in this case it is an ARM 7 processor.

(e) As this device is a Blackberry, it shows that its keyboard adhering to the ISO-8859-1 ASCII-based standard character encodings.

(f) Defines that this device is capable of displaying images.

(g) Defines the layout of the Keyboard, in this case it is a Blackberry device, so it makes use of a QWERTY keyboard.

(h) Defines the particular model of the device, in this case it is a Blackberry 7750.

```
<rdf:RDF>
<rdf:Description rdf:ID="DeviceProfileb
<I-- Hardware Platform Description -->
<prf:component>
(a)     <rdf:Description rdf:ID="HardwarePlatform">
        <rdf:type rdf:resource="http://www.openmobilealliance.org/tech/profiles/UAPROF/ccppschema-20021212#Ha rdwa re Platform" />
(b)     <prf:BitsPerPixel>16</prf:BitsPerPixel>
(c)     <prf:ColorCapable>Yes</prf:ColorCapable>
(d)     <prf:CPU>ARM7</prf:CPU>
        <prf:InputCharSet>
            <rdf:Bag>
(e)             <rdf:li>1S0-8859-1</rdf:li>
                <rdfili>UTF-8</rdfili>
            </rdf:Bag>
        </prf:InputCharSet>
(f)     <prf:ImageCapable>Yes</prfilmageCapable>
(g)     <prf:Keyboard>Qwerty</prf:Keyboard>
(h)     <prf:Model>BlackBerry 7750</prf:Model>
        <prf:NurriberOfSoftKeys>0</prf:NumberOfSoftKeys>
        <prf:OutputCharSet>
            <rdf:Bag>
                <rdfili>150-8859-1</rdfili>
                <rdfili>UTF-8</rdfili>
            </rdf:Bag>
        </prf:OutputCharSet>
```

Figure 3-4 Blackberry 7750 CC/PP profile (Research in Motion, 2010)

Note that Figure 3-4 is not a full CC/PP, but merely an extract showing the hardware capabilities of the device. This CC/PP continues to describe the software capabilities of the device. As demonstrated in the above example, the CC/PP framework approach provides much more detailed information about the device than the HTTP header approach.

3.2.5 Website transformation to mobile devices

As shown in the preceding section, it possible for a webserver to identify the requesting device and capabilities through methods, such as HTTP headers or a CC/PP profile.

Currently, there are various methods available to create a mobile compatible site. A commonly used approach to developing websites for mobile devices has been for content providers to develop a separate mobile site with device specific interfaces. However, developing a separate mobile specific site can be an expensive exercise for a content provider (Christos, Giorgos, & Ioannis, 2007). Furthermore, this solution can result in further technological and device support complexity for the content developers.

A second approach for creating mobile compatible websites is to transform the web content, as it exists on the server to a layout that is capable to be displayed on the mobile device. This can be accomplished through the use of techniques, such as transcoding or transformation proxies (Christos, Giorgos, & Ioannis, 2007). Schilit & Bickmore (1997) proposed the transformation proxy system called the "Digester". According to Schilit & Bickmore (1997), the "Digester" system is designed to automatically re-author a web site to display on small screen devices. In order to achieve this, the Digester is implemented as a HTTP Proxy, which dynamically re-authors the requested webpages using a heuristic planning algorithm and a set of structural page transformations to achieve the best layout for the screen size. Therefore, Schilit & Bickmore (1997) state that the "Digester"

approach is able to re-author any arbitrary designed document from the web for the desktop, read the characteristics of the target display and re-author the document to display appropriately on a mobile device.

3.3 Conclusion

The widespread use of mobile devices to access webpages has resulted in many challenges in the development of mobile sites. In order to assist web developers in addressing the challenges imposed by mobile devices, various best practices and guidelines have been developed. This chapter divided these best practices and guidelines into these developmental stages: Concept, Design, Implementation, and Use. As discussed, the Concept and Use stages of web development are similar between mobile and traditional website development. However, the Design and Implementation stage differ when developing for mobile web and these stages are addressed by mobile specific best practices, guidelines and methods.

The review of the best practices provides an understanding of important aspects of mobile web development. The best practices document discussed in this chapter stressed the importance of creating a consistent user experience throughout the mobile website. Furthermore, techniques for the identification of device browser type and capability are described. This identification of browser type and capability plays an important part in determining how the mobile website renders for the requesting device.

The approaches, techniques and guidelines explored in this chapter provide a starting point for the development of a mobile web transformation, which is discussed in the next chapter.

Chapter 4. Conceptual framework foundation

The previous chapters discussed the problems and challenges imposed by mobile devices. These mobile device challenges prevent websites designed for traditional media, such as desktop computers, from displaying or functioning correctly on mobile devices. Furthermore, these chapters investigated best practices and guidelines that web developers can use to develop websites that will be displayed properly on mobile devices. Currently, web developers wishing to have a mobile version of a website create a separate mobile version of the existing site. This is often unviable for many developers as additional resources will be required for recreating a website that essentially already exists. Furthermore, the web developer might not have the skill or understanding to develop an effective mobile website.

Therefore, this dissertation can now move on to address the problem of how a web developer should create a single website, which adapts its output depending on if it is displayed on a mobile device or a traditional computer. In order to do so, a conceptual framework foundation will be discussed in this chapter. This foundation will then be refined through prototyping. This will contribute to the development of the framework and accompanying guidelines for mobile transformation.

As described above, this chapter will define the conceptual foundation for a framework to transform a single web source file for both mobile and traditional device use. This conceptual idea will explore the concept of using a set of structure elements and a transformation proxy to achieve website transformation.

4.1 Structure of content

In order to structure the transformed traditional web content into a readable and comprehensible form on mobile devices, it is important to understand how people read and respond to text and images on small screens.

Dalal, Quible, & Wyatt (2000) found that a user's comprehension of a webpage is determined by how well the document aids in the process of creating a mental model of the site. This mental model can be defined as the user's mental representation of the objects and sematic relations of parts of the document. Furthermore, it was found that users of mobile devices are able to read and comprehend text if it is displayed in chunks rather than in a continual stream. The use of this method of displaying text on a mobile display only resulted in a minor reading performance degradation compared to reading text displayed on a traditional screen. However, it is noted that the biggest reduction in effectiveness occurs in mobile browsers with limited display width. However, it is noted that the biggest reduction in effectiveness occurs in mobile browsers with limited display width (Jones, Marsden, Mohd-Nasir, Boone, & Buchanan, 1999). Jones et al (1999) work further cites two interesting effects observed by users of smaller displays. Firstly, users interacted with the small display more than users with a larger display. Users were found to be able to navigate forward and backwards more on a small screen than a large one. Based on this, it was suggested that this significant scrolling movements could be related to the users trying to orientate themselves to the layout. Secondly, it was found that a large percentage of users indicated that they would rather have a larger screen size suggesting that despite device performance, first-time users might perceive that mobile device platforms to be inferior to conventional systems (Dillon, Richardson, & Mcknight, 1990 as cited in, Jones,Marsden, Mohd-Nasir, Boone, & Buchanan, 1999).

The posed solution for mobile transformation concept has similar aspects to the "Digester" re-authoring approach, described in Chapter 3.2.5.

Therefore, it is beneficial to review some of the lessons that Schilit & Bickmore (1997) documented during the development of the Digester.

Use of Images - It was found that keeping some of the original images in the document to maintain look and feel of the original document was important. A common technique is to keep the first and last image (bookend images) while removing the rest. An image resizing rule of thumb is to reduce images by a standard percentage, dictated by the ratio of the original screen size and the size of the requesting device. However, it should be noted that images containing numbers or text can only be reduced by a small amount before they become illegible.

Section Headers - It was noted that section headers are not used correctly in web design and therefore cannot be used to provide structural outline.

White Space - Whitespace is at a premium in mobile devices; therefore, sequences of paragraphs (p tags) or breaks (BR tags) can be removed and the content collapsed into one. Furthermore, lists (UL, OL and DL tags) take up valuable space and can be reformatted into simple text blocks with spaces between items.

By reviewing the techniques and recommendations discussed in Chapter 3 and the lessons documented by Schilit & Bickmore (1997) during the development of the Digester re-authoring method, a foundation for the development of a website transformation method was developed.

4.2 Overview of design

As discussed in Chapter 2 and Chapter 3, mobile devices are limited in what they can display. It is understood that a traditional website design may incorporate aspects and technologies that are not suitable for a mobile device.

Some of these aspects can include, flash elements, JavaScript and unnecessary pictures or content.

This conceived approach will enable the transformation of a traditional website into a mobile ready format. In order to accomplish this, elements of a traditional website need to be selected for mobile use. These selected elements are intended to provide site information and functionality to the mobile website browser. Elements that are not selected are deemed not suitable for mobile transformation and are removed during the mobile transformation process. The envisaged result of this tagging process would be a website that has had unnecessary data "filtered" out resulting in only mission critical data and function displaying on the mobile device. In order to select these elements of the traditional site, it is proposed that a set of custom HTML tags are developed. These tags would be embedded within the standard mark-up of a HTML page by the web developer. These tags would surround the selected data to define the aspects and areas and functionality of the traditional page that would be usable on a mobile site.

The use of custom HTML tags to define the mobile areas of the site, takes advantage on the ability of the browser not to render unrecognized HTML tags (MSDN, 2010b). Therefore, should a traditional browser render the page, the custom tags will be ignored during render and have no influence on the page outlay. Once the set of tags has been embedded into the structure of the HTML page, it is envisaged that an engine will intercept the output stream of the HTML page. This engine will then "read" the custom HTML tags and render the mobile elements accordingly. This results in a single web source file being used for both traditional and mobile device output. The approach of using embedded tags to define elements is based on the approach used by common web languages, such as PHP and ASP, to define server tags in an HTML file.

4.2.1 Custom structure tags

As defined above, the proposed method of element selection is to develop a set of custom tags to define elements that can be transformed to the mobile web.

In order to define what type of elements would likely be selected in a traditional site for transformation. The analysis of a traditional websites has shown that there are distinct "areas" that replicate across most webpage designs. They can be defined as follows:

Page Content Area–This defines the main content of a webpage.

Page Navigation Area– This area defines blocks of hyperlinks that provide website navigation.

Page Item Area – This area defines elements that make up the page or provide functionality, but do not fall within the Page Navigation or Page Content outline.

Page Name Area - This area defines the name of the page being viewed.

To demonstrate this concept, Figure 4-1 shows, how a web page can be broken down into "areas". The following annotations describe the identified "areas".

A - Page Content Area.

B - Page Item Area.

C - Page Navigation Area.

D - Page Name Area.

In order to identify these "areas" set of custom HTML tags can be developed to identify these areas in HTML mark-up code. However, in order to develop these custom HTML tags, it is important to understand how HTML tags are structured.

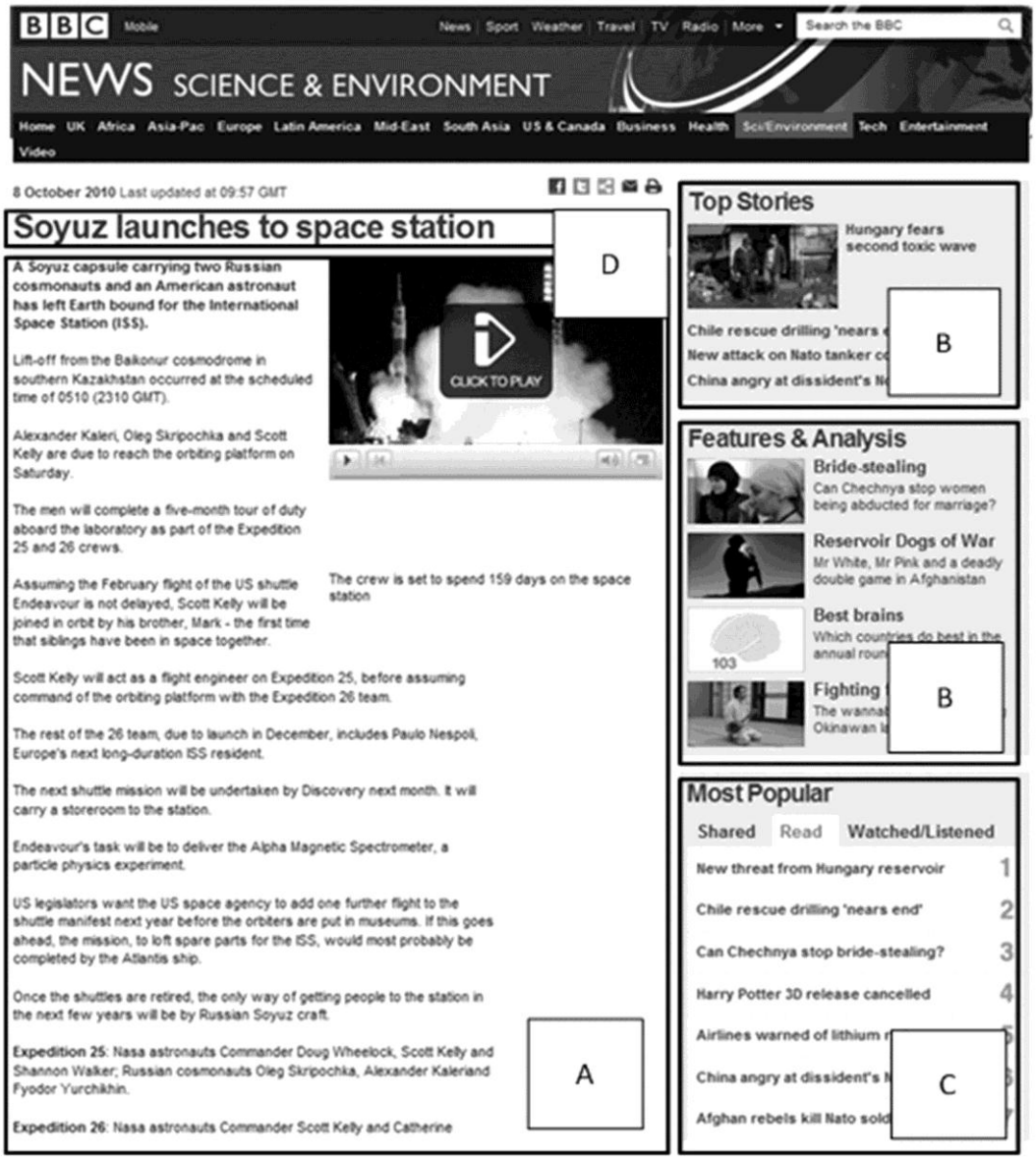


Figure 4-1 Screen shot of a BBC webpage showing proposed page areas.

In order to comply with the HTML specification put forward by the W3C, an HTML element must have a start delineator and an end delineator. This must surround the text that the author wishes to manipulate (Berners-Lee & Connolly, 1995). Figure 4-2 shows an example of a standard Level 2 heading HTML tag, showing the start and end delineators.

```
<H2>Header Text</H2>
```

Figure 4-2 HTML Example showing element structure

Based on the required structure of an HTML element, a custom set of HTML tags can be developed to define the elements selected for mobile transformation. Using the element areas described above, as a starting point, the following set of custom HTML structure tags are proposed do define mobile elements. Figure 4-3 shows the proposed set of custom HTML tags to define mobile areas of the HTML site. It is proposed that all custom tags used in this transformation method will have the prefix "mobi" in the element name. The use of this prefix is to differentiate the tags from other HTML tags that may be embedded in the page HTML mark-up.

```
a)    <mobicont> </mobicont>
b)    <mobinav> </mobinav>
c)    <mobiname> </mobiname>
d)    <mobiitem> </mobiitem>
```

Figure 4-3 Proposed custom HTML Tags

The following annotations describe the custom HTML tags defined in Figure 4-3.

a) **The *"mobicont"* element that will define the content sector of the site that the author wishes to be rendered for mobile devices.**

b) **The *"mobinav"* element will define the navigation sector that contains site navigation elements.**

c) **The *"mobiname"* element will define the name or main heading of the page.**

d) **The *"mobiitem"* element will define the sectors of the site that are not part of the main site content but can provide additional information or profile additional site functionality.**

The envisaged design of this system will allow traditional HTML elements, such as, list tags, heading tags and so forth, to be defined within the custom HTML elements. This will allow the web developer to take advantage of browser specific interpretations of these standard HTML tags.

Figure 4-4 shows the envisaged use of the custom HTML tags in a page HTML mark-up. It is important to note the standard HTML elements embedded between the start and end delineators of the custom HTML tags.

4.2.2 Development of the rendering algorithm

As discussed above, it is proposed that custom HTML tags are developed and used to delineate areas of a traditional website for mobile use.

Once these areas are defined, it is envisaged that the page will pass though code that will read and extract the defined elements of the HTML page. The envisaged design of the transformation code is described as follows. In order to fulfill the requirement of making the webpage accessible to both desktop based browsers and mobile device browserS, a mechanism must be devised that can provide a

standard version of the webpage to traditional web browsers, and provide a mobile ready version when delivered to mobile devices.

Approaches to development of this mechanism will be investigated during prototype development and discussed in the next chapter.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title></title>
   <link href="StyleSheet.css" rel="stylesheet" type="text/css" />

</head>
<body>
<div id="PageWrap">
<mobiname><div id="Header">Site Name</div></mobiname>

<div id="LeftCol"><div id ="LeftItem"><mobiitem><b>Search Site</b><br /><input type="text" /
><input type="button" value="Search" /></mobiitem></div><div id="LeftItem"><mobinav><h3>Site
Navigation</h3><ul><li><a href="#">Navigation Link </a></li><li><a href="#">Navigation Link </
a></li><li><a href="#">Navigation Link </a></li><li><a href="#">Navigation Link </a></li></ul></
mobinav></div><div id="LeftItem"><b>Advert</b><br />Etiam eu mi neque, ullamcorper faucibus
ligula. Praesent molestie scelerisque massa, eget eleifend dolor ultricies sed. </div></div>

<div id="Content"><mobicont><h3>Page Content</h3>Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Vestibulum orci tellus, rutrum eget imperdiet eu, mattis quis nulla. Etiam eu mi
neque, ullamcorper faucibus ligula. Praesent molestie scelerisque massa, eget eleifend dolor ultricies
sed. Vestibulum quis nibh a eros

   <ul>
      <li>Maecenas non felis velit, ac dictum lorem.</li>
      <li>Mauris quis erat sed odio ultrices placerat.</li>
      <li>Suspendisse sed mi sed ipsum mattis congue vitae eget erat.</li>
   </ul>

Nulla venenatis libero nec lacus fringilla posuere. Sed mattis, felis eget consectetur condimentum,
metus turpis mattis turpis, in rhoncus turpis leo non erat. Vestibulum ante ipsum primis in faucibus
orci luctus et ultrices posuere cubilia Curae; Maecenas sit amet justo erat, ac cursus dui.<br />
   <br />
   <img alt="" class="style1" src="pap%20park.jpg" /><br />
   <br /></mobicont>
   <div id="AdvertBar"><b>New Product</b><br /><br />ornare a mollis ultrices, tincidunt eget nunc.
<br />Maecenas commodo risus nibh. Ut ornare, tortor nec posuere bibendum, massa urna
malesuada  </div>
   </div>

</div>
</body>
</html>
```

Figure 4-4 Envisaged use of the custom HTML tags.

In order to create a mobile ready version of the webpage, interception or redirection of the page request must occur in order for the transformation code to read and process the elements of the custom tag delineated webpage.

It is envisaged that the transformation code will parse the delineated sections of the webpage into specific blocks. These blocks will consist of all the defined elements as shown in Figure 4-5.

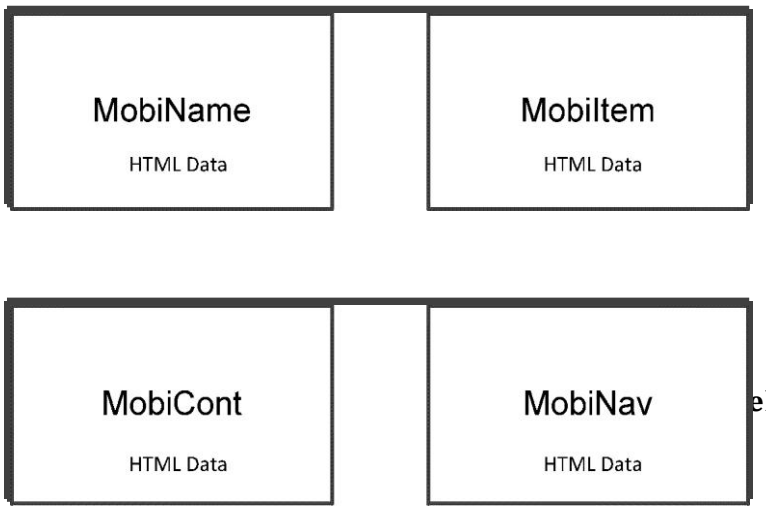
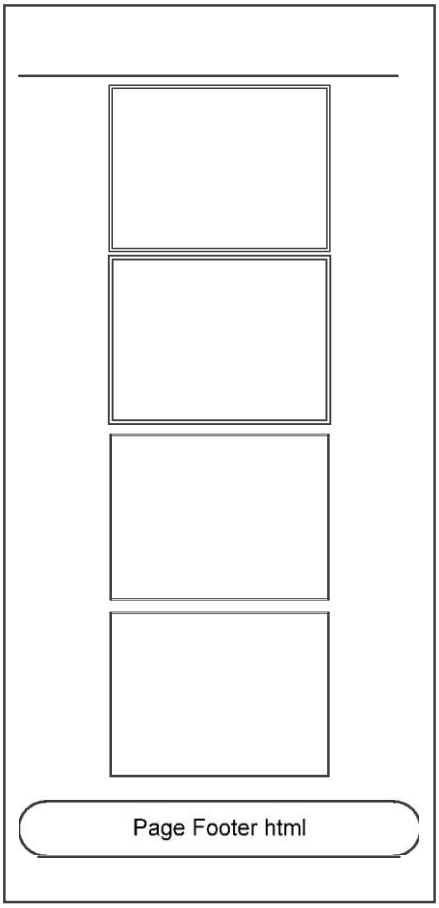


Figure 4-5 Graphical depiction of element blocks

It is envisaged that once the creation of the HTML blocks are complete, various algorithms will be run to transform the raw HTML elements for mobile use. These algorithms will process factors, such as excessive white space and navigation.

Once the above algorithms are run, it is envisaged that a dynamically created HTML page will be constructed. This HTML page will be constructed by rendering or "building" the HTML elements blocks in a specific order. This order is developed with the page name and navigation elements near the top to aid in usability. Elements identified by the *< Mobiitem >* tag, such as search boxes, will be rendered below the navigation elements. Lastly, the selected content, such as text, is rendered. If there is more than one block of identified elements in a page, they

will be rendered beneath one another on a first read first out basis. HTML page header and footer code is added or maintained to create a valid HTML markup.



This build order is shown in Figure 4-6.

Figure 4-6 Envisaged HTML build order.

4.3 Conclusion

The increasing use of mobile devices has resulted in the need for traditionally developed websites to be displayed in a mobile format. As described in the previous chapters, the approach of developing a separate mobile version of an existing website is often not viable. Therefore, this chapter defined a foundation for the development of a mobile transformation framework. This chapter

described how a traditionally developed website is analyzed and areas are defined to be displayed on a mobile version of the website are delineated by a set of custom HTML tags. Furthermore, it was also described how a transformation code parses the delineated HTML document and renders a mobile ready website.

It is envisaged that this transformation framework would allow web developers the capability to create a single set of source files, for use by both mobile and traditional devices. Furthermore, it is envisaged that existing websites can be converted for mobile use by implementing this approach.

The following chapter will incorporate the lessons learned from Chapter 2 and Chapter 3 relating to mobile development and build upon the conceptual foundation that was proposed in this chapter.

Chapter 5. Prototype development

Chapter 4 proposed a foundation for the development of a framework for mobile transformation. It was proposed that a set of custom HTML tags are used to delineate sections of the traditional website for mobile transformation. It was envisaged that the transformation would occur when the site is requested by a mobile device.

This chapter will describe the development of a prototype in which the transformation method proposed in Chapter 4 will be built upon.

Following the research methodology discussed in Chapter 1, two prototypes will be developed. The successes and failures found during the development of the first prototype will directly influence the development of the second prototype. While each prototype may follow a different approach, the principle goal is to determine a solution for intercepting an original HTML page, filter it and render a mobile ready page, while maintaining the webpage's functionality. Furthermore, the developed solution must allow for the display of the unmodified HTML page should it be requested by a non-mobile device.

5.1 Development environment

The prototypes will be developed using the Microsoft ASP.Net platform with C# as the code behind and .NET 4.0 frameworks. Furthermore, the webserver will be running the Microsoft Internet Information Service (IIS). While it might be possible to develop this solution using other webservers and languages, such as Ruby on Rails or the Apache platform, such development is out of the scope of this research.

In order to develop the prototype, it is important to understand the request flow between a requesting browser and the webserver where the website resides.

Figure 5-1 shows a basic overview of a standard request flow between the client and the webserver.

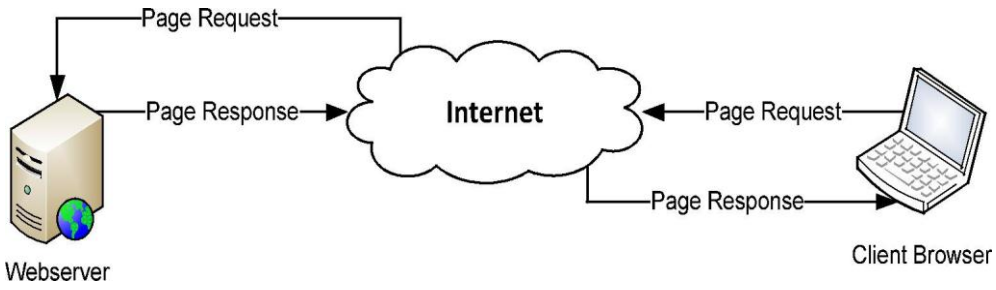


Figure 5-1 High-level overview of a web request and response

Figure 5-1 shows that the client browser creates a page request and the webserver creates a response with the requested HTML web page. This scenario makes no distinction between device types and will deliver the same HTML page to both the desktop-based browser and mobile device browser.

In order to allow modification of the webpage for mobile devices, it is proposed that the server page response is modified. This must occur after the page is processed, but before it is transmitted to the client browser. The proposed location of this request modification is shown in Figure 5-2.

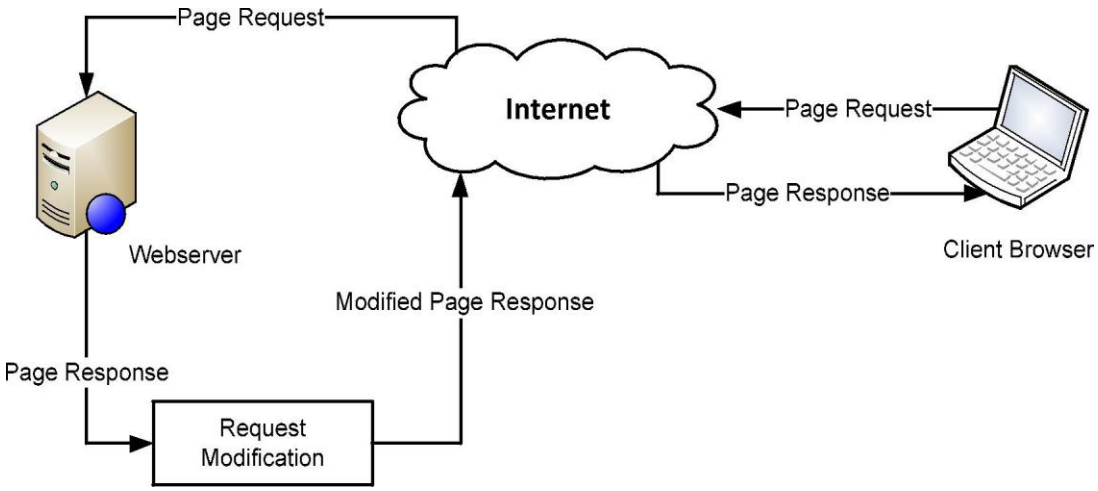


Figure 5-2 Location of request modification.

5.2 Approach A – A physical proxy server

As discussed above, a method needs to be developed to modify the server response before it is received by the client browser. The modification must occur without interfering with the original HTML page should it be requested by a non-mobile device.

Upon study of the requirements and request flow, a solution was devised whereby a different "request pathway" was developed for mobile devices. It was determined that this request should be isolated in order not to interfere with a traditional browser request. In order to achieve this, a separate server instance on a separate domain or sub-domain was created. The mobile transformation code for this approach resides on this new server instance and processes the HTML page when requested by a mobile device.

Figure 5-3 shows how this scenario's web request and response flows. Note that there are two separate server instances, Webserver A and Webserver B.

The first server instance: Webserver A, is located at the domain *www.sitename.com* and is the location of the requested website code. In order to retrieve the original unmodified site, a user would request the domain *http://www.sitename.com* and Webserver A would respond with the requested data, shown by annotation 1 and 2.

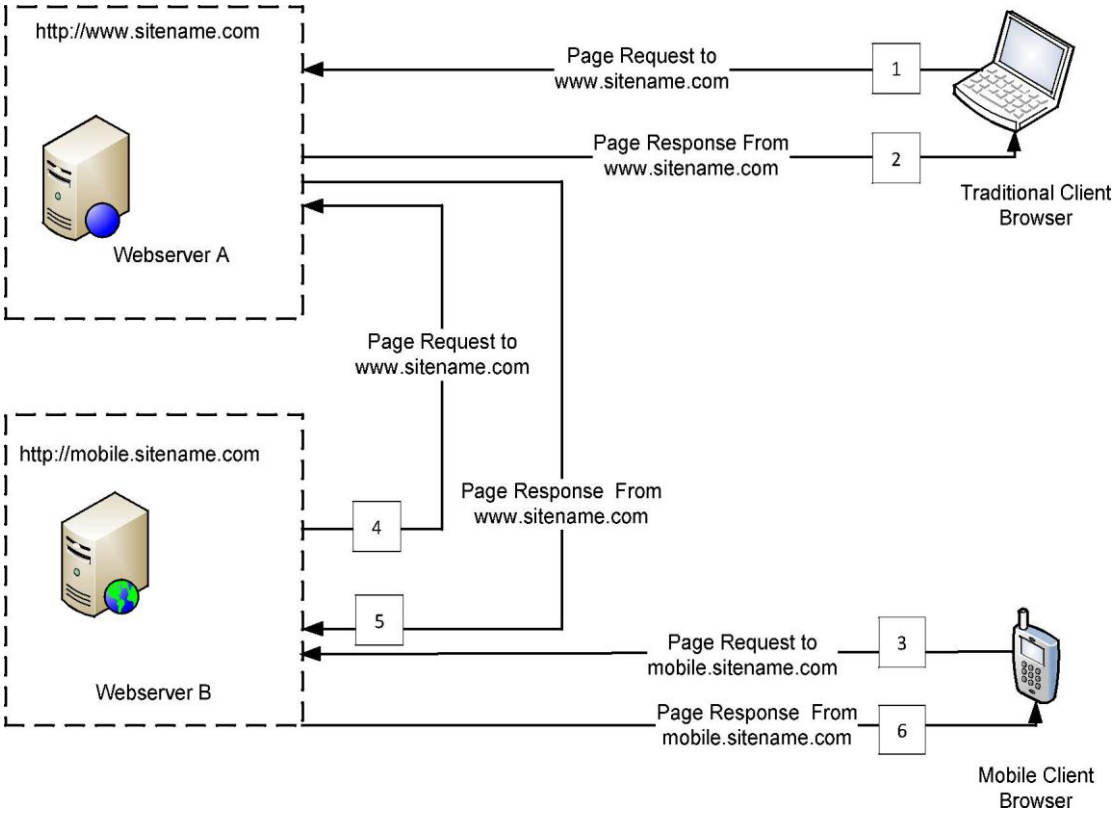


Figure 5-3 Request flow with separate server instance

The second server instance: Webserver B, located at a subdomain with a URL of *http://mobile.sitename.com*. This instance held the appropriate code to modify and render the mobile site version. A user from a mobile device browser would therefore request a URL of *http://mobile.sitename.com* and would be resolved though the standard DNS procedures to webserver B (annotation 3).

The transformation code in webserver B would then execute and initialize a new HTTP request of the original page from webserver A (annotation 4). The proxy code in webserver B would then receive the incoming HTTP response from Webserver A (annotation 5). The mobile rendering engine would then transform and modify the original HTML page and forward the modified page content to the mobile device (annotation 6). The mobile only request flow is shown in Figure 5-4.

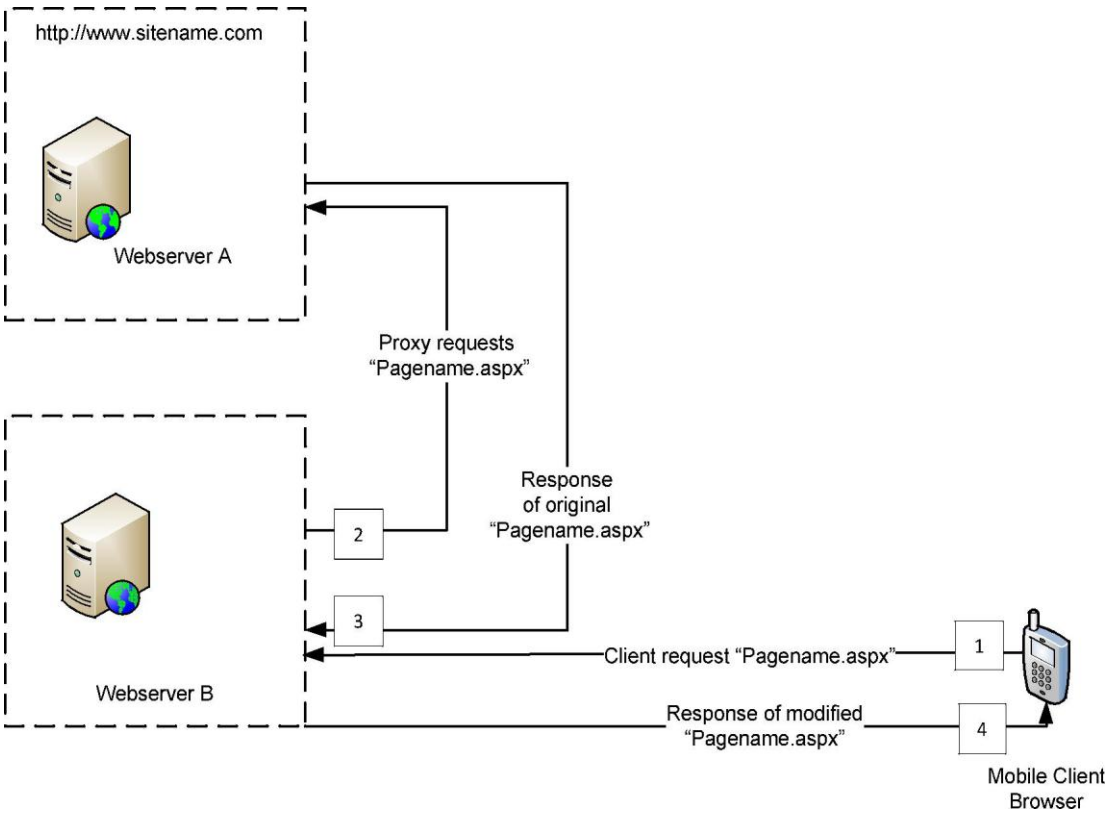


Figure 5-4 Diagram showing flow of a page request from mobile device

During prototype development and testing of this approach, it was found that having two separate server instances used in this approach encountered functionality problems. These problems, as well as potential solutions, are discussed in the following sections.

5.3 Lessons learned from Approach A

Valuable lessons were learned from creating the prototype in Approach A. The next sections illuminate these lessons.

5.3.1 External resource files

During development of this prototype, it was found that external resource files such as the cascading style sheet (CSS), presented problems. In order to understand the cause of this problem and to develop a possible solution, it was important to investigate the use of CSS and other external resource files.

External resource files are used by the browser to provide functionality and visual styles to the webpage. These files are developed by the web developer and referenced in the HTML page code. While CSS styles can be defined in the page itself, it is recommended that CSS files are located externally for enhanced flexibility (Bos, Çelik, Hickson, & Wium lie, 2009).

```
<link href="StyleSheet.css" rel="stylesheet" type="text/css" />
```

Figure 5-5 HTML reference to external style sheet

Figure 5-5 shows a typical HTML page reference to an external style sheet. The path of this style sheet is a relative URL to the root of the requesting page. A relative URL does not contain any machine or protocol specific information, but refers to a path of a resource on the same machine as the current document (World Wide Web Consortium, 1997).

The use of a relative URL for external resource files presented a problem during page rendering at Webserver B. After investigation, it was determined that the CSS file was on webserver A, and could not be accessed by webserver B using the supplied relative URL.

In order to address this problem, a solution was developed to automatically scan the HTML code received from Webserver A, detect a relative URL and rewrite the output code to become an absolute URL. An absolute URL is the full path needed to access a file resource; this is shown in Figure 5-6.

```
<link href="http://www.sitename.com/StyleSheet.css"
rel="stylesheet" type="text/css" />
```

Figure 5-6 Relative URL rewritten to become an absolute URL

During development of the prototype, it was found that the following components were also affected by the above relative path problem.

- **Image references.**
- **External JavaScript libraries.**
- **Navigation URLs to other documents on the webserver.**

The solution discussed above to correct the CSS file paths, was further adapted to correct all relative paths in the page.

After the above solution was implemented, the proxy server did render the requested page correctly on mobile devices as shown in Figure 5-7.

Despite the successfully visual rendering of the site on the mobile device, functionality problems were encountered. These problems are described in the following section.

5.3.2 Functionality problems

Figure 5-7 shows that the prototype does visually render the test site in a mobile outlay; however, upon testing it was found that no event functionality was rendered in the HTML page. Investigation into the cause of this issue resulted in the discovery of a major flaw in the approach of having a proxy server to render the content.

Figure 5-7 Nokia N70 Emulators rendering test page

In order to better understand why this problem occurred, it is important to revert to our initial design goals. As stated, we do not wish to modify the application logic of our site, but merely the interface to suite a mobile device. Therefore, it is important that the "code behind" remain untouched. Investigation into the cause of this problem concluded that the final rendered page received by the mobile client, did not relay the "event" or inputted data successfully via Webserver B to Webserver A. This resulted in no response from Webserver A. This failed event flow is shown in Figure 5-8.

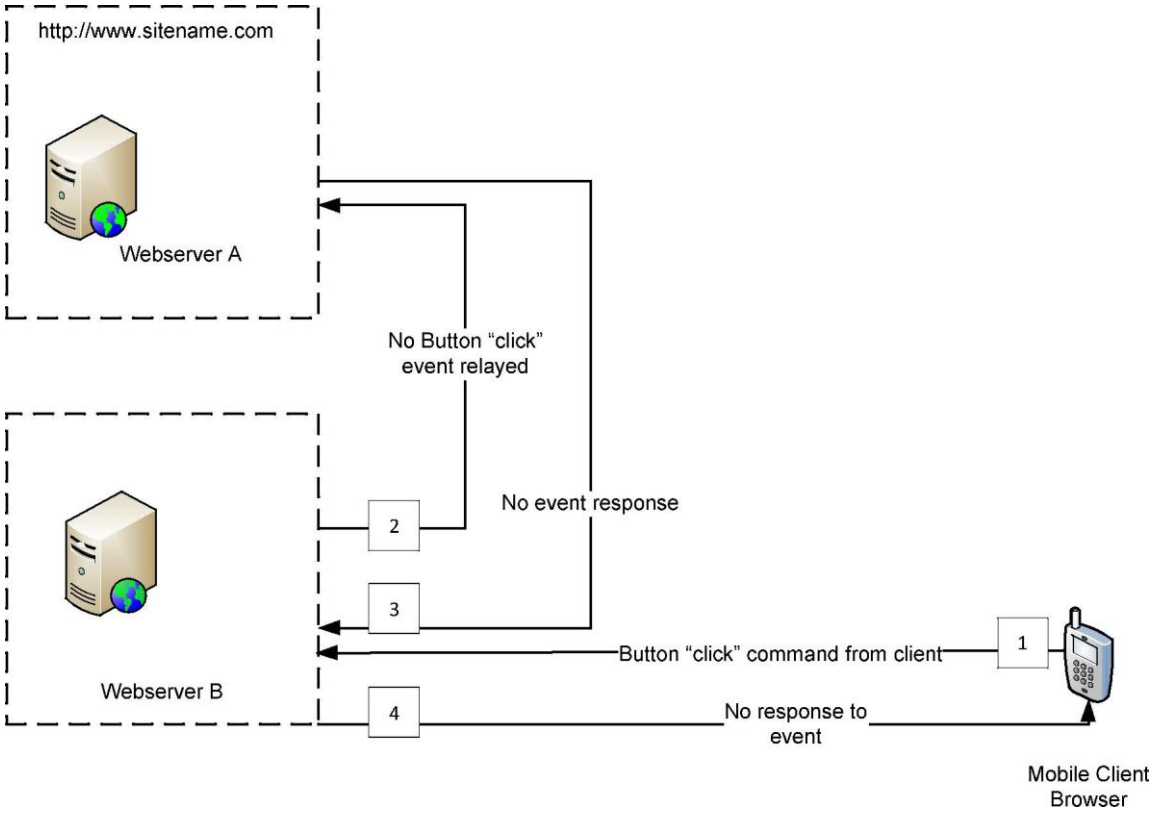


Figure 5-8 Button "click "event flow

This approach to designing the mobile transformation proxy only fulfilled the original goal of "transformation of content". Furthermore, it was foreseen that other server functionality, such as session state and other functions that require

server post-back, would encounter problems. Therefore, this approach was deemed not feasible and this approach was abandoned.

Based on the lessons learned from the above implementation attempt, it was determined that further study was required to understand how the ASP.Net server handles pages. A second approach was therefore proposed and tested. This approach is discussed in the next section.

5.4 Approach B – Interception of HTTP stream

As shown in Approach A, the use of a separate proxy server to process the transformation request was not viable. This resulted in further investigation into how web requests are handled by ASP.Net with the intention to find an alternative solution. Investigation into the operation of the ASP.Net framework resulted in the review of the ASP.Net pipeline process. Mitchell (2008) describes the ASP.Net pipeline process as follows:

> **"When a web browser requests an ASP.Net page from a webserver, the ASP.Net engine processes the request though a number of steps that generate the resulting mark-up that is returned to the requesting browser. This process is known as the ASP.Net pipeline, and is also responsible for tasks like authentication and authorization".**

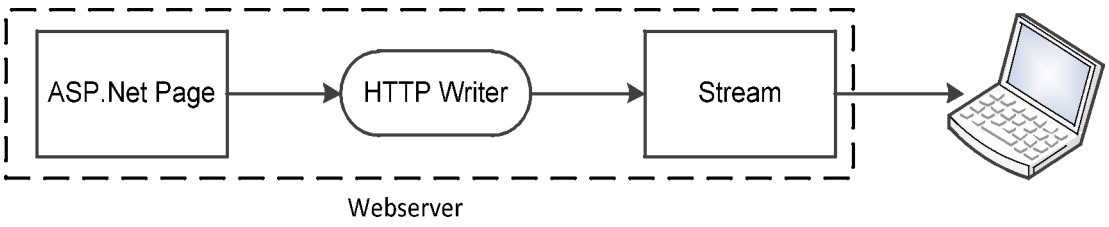


Figure 5-9 ASP.Net pipeline (Mitchell, 2008)

Figure 5-9 shows a high-level overview of ASP.Net pipeline, described below.

- **The ASP.Net rendered page mark-up is sent to the HTTP Writer.**
- **The HTTP Writer Buffers the data and writes it out to a stream**
- **The Webserver reads from the stream and sends it to the client.**

Mitchell (2008) describes how a web developer can intercept, inspect and modify the rendered mark-up before it is sent to the requesting web browser by making use of the *response.filter* function of the ASP.Net pipeline.

Based on this, it is possible through the use of a custom implementation of the ASP.Net Response Filter, construct a custom stream and insert it into the pipeline. This capability was further expanded upon by adapting the mobile transformation code to intercept and modify the stream. In order to do so, the transformation code read the incoming stream, ran the transformation code and relayed the new mobile ready HTML to the outgoing stream. This modified outgoing stream is then provided to the requesting browser. This process is described in Figure 5-10 .

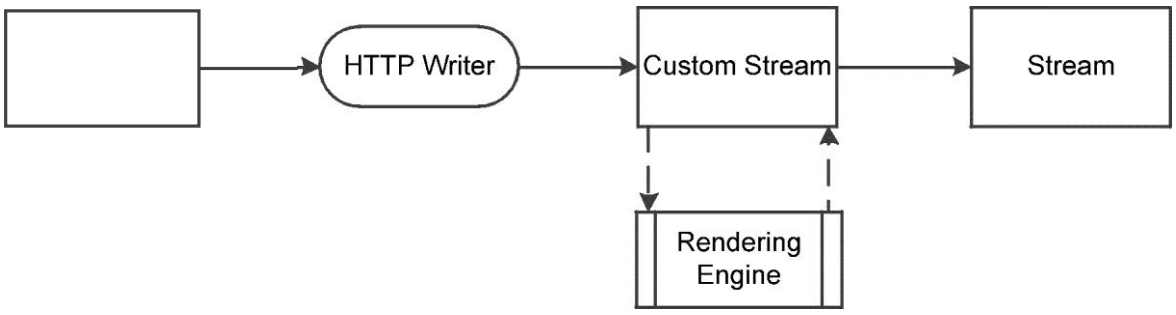


Figure 5-10 Stream interception and modification

In order to implement the transformation code using this approach, a method must be developed that will allow this solution to be added to an existing ASP.Net server, with as little modification as possible.

5.4.1 HTTP module

In order to maintain simplicity and ease of use, it is important to design the transformation engine to be able to "plug" into existing ASP.Net projects and servers. Furthermore, the transformation engine must be able to be placed into a position whereby it has access to the ASP.Net streams in order to modify them.

Investigation has shown that using the HTTP Modules and HTTP Handlers provided by ASP.Net, are best used for this purpose, due to the tight integration they have with ASP.Net (Microsoft Corporation, 2007).

Microsoft Corporation (2007) describes that a request is processed by multiple HTTP modules, such as the authentication and session module. Thereafter, the request is passed to and processed by a single HTTP handler. Once the handler has completed processing the request, the processed data is passed back to the HTTP modules.

Figure 5-11 shows how the request flows though the HTTP modules before the HTTP handler executes. These modules enable developers to intercept, participate in or modify each individual request before it is returned back to the requesting device. (Microsoft Corporation, 2007).

It is important to integrate the transformation engine into the request flow without disrupting the intended operation of the website. This integration can be accomplished by developing the transformation engine into an HTTP module.

In order to develop a HTTP module, it is important to understand what methods a developer can use to interface the HTTP module events within provided ASP.Net framework.

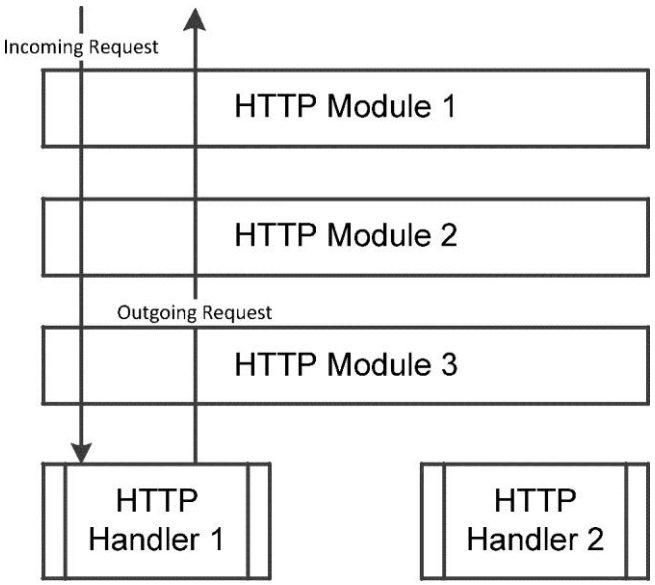


Figure 5-11 HTTP Modules and HTTP Handler function (Siddqui, 2002).

The HTTP module is an implementation of the *IHTTPModule* interface found in the Microsoft .NET *System.web* namespace. Furthermore, an *HttpApplication* class provides the developer with programmatic access to module events (Microsoft Corporation, 2007).

Table 5-1 lists the events exposed by the IHTTP Module as well as a brief explanation of their function.

The events exposed by the ASP.Net framework, makes it possible to place the transformation code within the HTTP module, to be executed when the *BeginRequest* event is fired. The code to accomplish this is shown in Figure 5-12.

Table 5-1 Events accessible in HTTPModule (Microsoft Corporation, 2007)

Module Method	Description
BeginRequest:	**Request has been started. If you need to do something at the beginning of a request (for example, display advertisement banners at the top of each page), synchronize this event.**
AuthenticateRequest:	**If you want to plug in your own custom authentication scheme (for example, look up a user against a database to validate the password), build a module that synchronizes this event and authenticates the user how you want to.**
AuthorizeRequest:	**This event is used internally to implement authorization mechanisms (for example, to store your access control lists (ACLs) in a database rather than in the file system). Although you can override this event, there are not many good reasons to do so.**
ResolveRequestCache:	**This event determines if a page can be served from the Output cache. If you want to write your own caching module (for example, build a file-based cache rather than a memory cache), synchronize this event to determine whether to serve the page from the cache.**
AcquireRequestState:	**Session state is retrieved from the state store. If you want to build your own state management module, synchronize this event to grab the Session state from your state store.**
PreRequestHandlerExecute:	**This event occurs before the HTTP handler is executed.**
PostRequestHandlerExecute:	**This event occurs after the HTTP handler is executed.**
UpdateRequestCache:	**This event writes output back to the Output cache. If you are building a custom cache module, you write the output back to your cache.**
EndRequest:	**Request has been completed. You may want to build a debugging module that gathers information throughout the request and then writes the information to the page.**

```
public class MobiMod : IHttpModule
{
    private System.Web.HttpApplication app;
    public MobiMod()
    {
        //
        // TODO: Add constructor logic here
        //
    }

    public void Dispose()
    {

    }

    public void Init(HttpApplication application)
    {
        application.BeginRequest += new System.EventHandler(BeginRequest);   ( A )

        app = application;


    }

    public void BeginRequest(Object sender, EventArgs e)
    {
        if(app.Request.Browser.IsMobileDevice == true)                        ( B )
        {

        HtmlFilter OutputFilter = new HtmlFilter();

        app.Response.Filter = new ResponseFilter(app.Response.Filter,app.Request.Browser);   ( C )
        }



    }
```

Figure 5-12 Code extract showing HTTP Module code

Figure 5-12 shows how the HTTP module has been used to initialize the *Response.Filter* function of ASP.Net . The annotations below describe the important functions of this code example.

(A) The *BeginRequest* event is wired-up to the initialization sequence of this module.

(B) In order to achieve our goal of enabling both traditional bowsers and mobile browsers to access the site, this conditional statement relies upon the built in capabilities of the ASP.Net browser to detect whether the browser is a mobile device. If the requesting device is mobile, then the entire mobile rendering engine will execute and modify the output.

If however, the browser is detected as being a traditional desktop-based browser, then the conditional statement will return false and the mobile rendering engine will not execute, and the page will display as normal.

(C) The ASP.Net response.filter is defined in the Microsoft .net framework as a get/set function, providing both reading and writing functionality to the output stream. In this case, a new instance of the *response.filter* class is initialized with two inbound parameters, the current unmodified Response stream from the ASP.Net pipeline and an instance of the *BrowserCapabilities* class containing the requesting browsers information. In order to modify the HTTP Stream, a custom implementation of the *response.filter* class is created. This custom class inherits HTTP Stream functionality from the *System.IO.Stream* class. Furthermore, in order for the transformation engine to write the modified HTML to the outgoing stream, *System.IO.Stream.Write()* method class is overridden. This custom implementation of the *System.IO.Stream.Write()* method, contains the functionality that enables ASP.Net pipeline stream to be read, modified and returned to the pipeline. Thereafter, the transformed HTTP Stream is sent to the requesting browser.

Figure 5-13 shows the custom implementation of the System.IO.*Stream.Write* method. This method has been modified to pass the incoming stream to the mobile transformation engine.

```
public override void Write(byte[] buffer, int offset, int count)
   {
      //Get text from response stream.
      string originalText = System.Text.Encoding.UTF8.GetString(buffer, offset, count);          (A)

      SBuilder.Append(originalText);
      if (SBuilder.ToString().Contains("</html>"))                                               (B)
      {
            //Alter the text.


            HtmlFilter HtmlFilter = new HtmlFilter();

            string OutputHtml = HtmlFilter.StripHtmlToElements(browser, SBuilder.ToString());   (C)

            //Write the altered text to the response stream.
            buffer = System.Text.Encoding.UTF8.GetBytes(OutputHtml);                             (D)
            this.baseStream.Write(buffer, 0, buffer.Length);

      }
```

Figure 5-13 Custom stream write method

The following annotations describe important aspects shown in Figure 5-13.

(A) The incoming stream from the ASP.Net pipeline is converted to a string, using the specified encoding format.

(B) ASP.Net stream writer does not write a steady stream to the response filter. This code solution is further discussed in section 5.1.1.

(C) The original HTML from the ASP.Net pipeline is passed to the mobile rending engine (*HtmlFilter.StripHtmlToElemnts()* method), once processed, the modified HTML is passed back and strung into a string variable.

(D) The modified HTML is converted from a string to an output stream and written to the ASP.Net pipeline for delivery to the requesting browser.

As described above, the interception of the HTTP stream is directed to the transformation engine, for processing. This transformation engine is described in the sections below.

5.4.2 Transformation engine

The mobile transformation engine provides the mobile ready HTML output to the mobile device. In order to accomplish this, the transformation engine reads incoming HTML, separates the areas defined by the custom mobile tags, and rebuilds the HTML for passing to the mobile device. The mobile transformation engine makes use of the HTML Agility Pack, an open source library. The use of this library allows the identification of the custom HTML elements described in Chapter 4.2.1. Furthermore, it allows HTML between these custom tags (nodes) to be extracted.

```
public string StripHtmlToElements(HttpBrowserCapabilities Browser, string Rawinput)
{
    HttpServerUtility server = HttpContext.Current.Server;

    BrowserCap = Browser;

    ////////// Load HTML into Parser /////////////////

     HtmlDocument alldoc = new HtmlDocument();

    alldoc.LoadHtml(Rawinput);

    HtmlNode MobiWrap = alldoc.DocumentNode.SelectSingleNode("//mobiwrap");

    ////////////////// defined mobile area/////////////////

     HtmlDocument doc = new HtmlDocument();

    doc.LoadHtml(MobiWrap.InnerHtml);
    ////////// Define Custom MobiTags ///////////////

    HtmlNodeCollection MobiName = doc.DocumentNode.SelectNodes("//mobiname");

    HtmlNodeCollection MobiNav = doc.DocumentNode.SelectNodes("//mobinav");

    HtmlNodeCollection MobiCont = doc.DocumentNode.SelectNodes("//mobicont");

    HtmlNodeCollection MobiItem = doc.DocumentNode.SelectNodes("//mobiitem");

    ////////////////// Build HTML /////////////////
```

Figure 5-14 Identification of the custom HTML tags

Figure 5-14 shows how the custom HTML tags are defined, as there can be multiple blocks of each area, for example many defined content areas in an HTML page. Each identified "block" is placed into a collection variable *HtmlNodeCollection*.

During prototype development, it was found that the custom HTML tags are required to be nested within a parent element. Therefore, the *mobiwrap* custom tag was developed. The *mobiwrap* tag is designed to be placed around the areas of the HTML page that contain mobile elements.

Once the "blocks" of mobile elements have been identified and extracted, methods are run to iterate the collections of mobile "blocks" and rebuilds them into a single sequence of HTML. Thereafter, the newly created sequences of HTML are built into the output HTML document. References to CSS files and other resources are maintained and incorporated into the headers and footers of the output HTML document. This HTML is then passed back to the HTTP stream for delivery to the mobile device.

5.5 Problems encountered

During prototype development of this approach, various problems and challenges became evident. These problems and challenges, along with solutions are described below.

5.5.1 ASP.Net pipeline stream

During the rendering of an ASP.Net page, the resulting output was written to an instance of the HTTPWriter. The HTTPWriter object takes the content and buffers the output, sending it to the stream in chunks (Mitchell, 2008). This "chunking" of the output presented a problem in that, if only a "chunk" and not the entire output passed to the mobile transformation code, the transformation of the HTML would

fail. This was demonstrated during prototype development, when inputs of only the first few hundred characters were found. Therefore, the mobile transformation code failed when it could not locate the closing tags of HTML elements.

In order to solve this problem, an instance of the *StringBuilder* class was needed to append these "chunks" as they are provided from the ASP.Net pipeline to form a complete data string. This appended data string is then validated to ensure that the complete HTML stream is passed to the mobile rendering engine. In order to ensure that the entire HTML document is present before transformation, various techniques were considered and tested.

Stream Length – The approach to count the amount of characters in the string and compare to a previously defined value was considered. However, due to the varying lengths that a HTML document can be, it is not feasible to use this approach.

Termination Characters – During prototype type development, it was found that the mobile transformation code required the HTML document end tag *</html>* in order to function correctly. Therefore, a conditional statement was developed to determine if the appended string contained this termination character; if so, the string was declared to be complete and passed to the mobile rendering engine. The use of this technique is shown in Figure 5-13 - annotation B.

5.5.2 Browser definition file

The initial design of this technique is to enable both a traditional browser and a mobile browser to request and view the same HTML page. Therefore, it is important for our transformation code to determine if a requesting device is a mobile browser or a desktop-based browser. The identification of browser type must occur in order to allow the mobile transformation to execute. In order to

identify the browser type the ASP.Net *BrowserCapiblities* class is used. The *HttpBrowserCapabilities* of the Microsoft .Net framework allows detection of the type of requesting browser and its capabilities by making use of the User-Agent string in the HTTP headers (Shepherd, 2005).

During prototype development and subsequent testing, it was found that some mobile device browsers were not identified as mobile devices. This resulted in the prototype providing the unmodified HTML page to these mobile devices. Investigation showed that the *HttpBrowserCapabilities* component of ASP.Net relies on a list of browser definitions, to which it matches the requesting browser. Due to the rapid development of mobile phones, this browser definition file was out-dated at the time of development, as some of the test mobile devices were not defined in the browser definition file. Further investigation revealed that an updated browser definition file is periodically provided by Microsoft. This updated browser file would need to be updated on the web server to ensure that the latest mobile devices are provided for.

5.6 Case study

In order to determine if the interception of the HTTP Stream described in section 5.4 and the subsequent mobile transformation was effective, testing was conducted on the prototype code. Furthermore, these tests were conducted in order to determine how the transformation techniques could be used on a real world webpage. Therefore, an HTML template was selected from the Internet and used as a case study. The traditional website template used in these examples is a fully standards compliant CSS template that has been "fitted" with the custom tags. These "fitted" custom tags around HTML elements followed the ideas put forward in Chapter 4.

For testing proposes, the website was hosted on a private ASP.Net server and the transformation code module described in Chapter 5.4.1 was "plugged" into the ASP.Net pipeline.

Confirmation of the successful installation of the website on the server was verified by a "visit" to the domain with the desktop Firefox web browser. The unmodified website displayed successfully in the browser as anticipated with no errors. This unmodified website is shown in Figure 5-15.

In order to test the functionality of the mobile transformation code and mobile device detection methods, the following tests were performed.

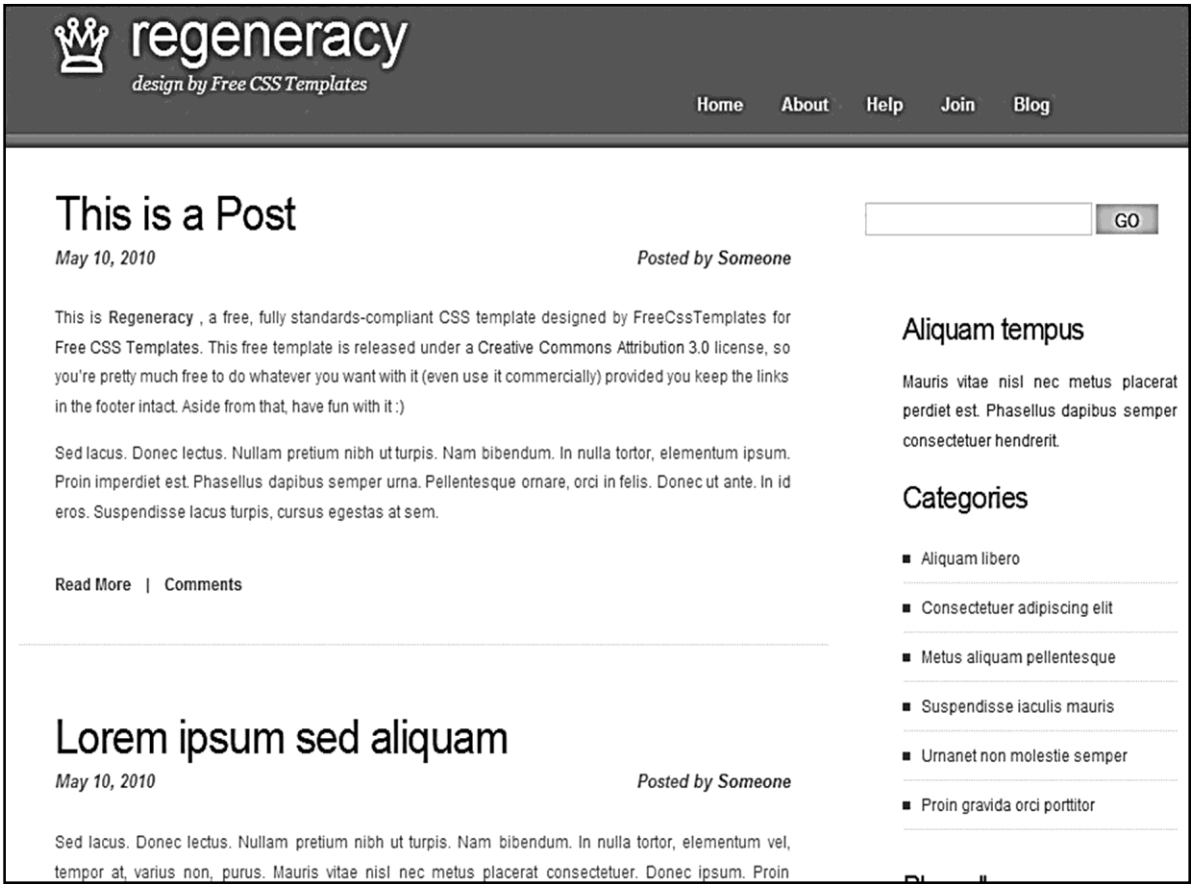


Figure 5-15 Unmodified website as seen in the Firefox web browser

5.6.1 Mobile device tests

The testing of this prototype consisted of accessing the website with a variety of mobile devices. The tests were conducted with a Nokia N75 mobile phone emulator, the Opera Mini mobile browser emulator and a Blackberry 9800 smartphone.

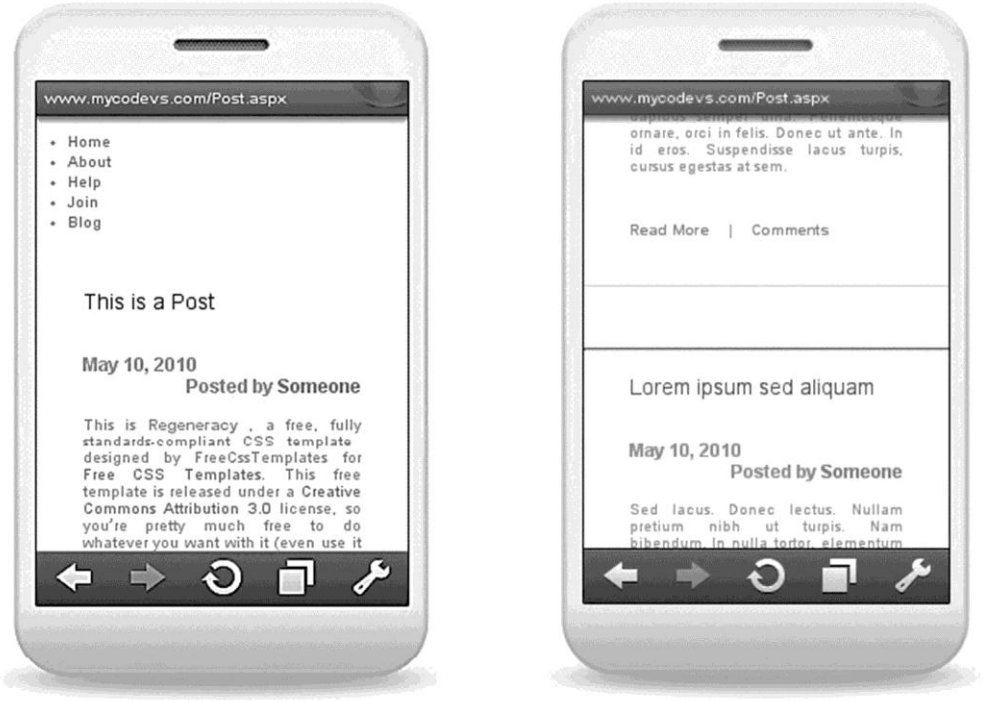


Figure 5-16 Website as viewed with Opera mobile device emulator

Figure 5-16 shows the automatically transformed website when viewed with the Opera mobile browser emulator. It is important to note that the URL entered is the same URL that was entered on the traditional browser. This shows that the mobile bowser detection did occur and the mobile transformation code was executed.

Furthermore, it should be noted in the rendered page how the "Element Blocks" described in Figure 4-5, were created and rebuilt according to the sequence in Figure 4-6. Furthermore, it is important to note that the categories navigation and search box elements shown on the unmodified site in Figure 5-15 are not rendered on the mobile site. These areas were not selected for mobile transformation and therefore verifies that the transformation code is working correctly.

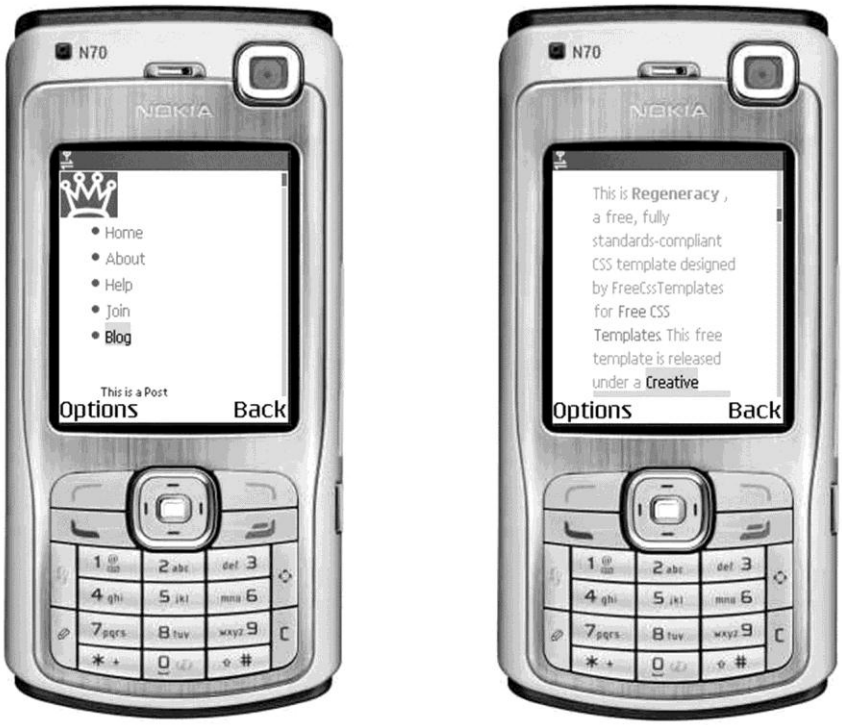


Figure 5-17 Nokia N70 Emulator showing Transformed Website

Figure 5-17 shows the transformed website in the *DotMobi* emulator of a Nokia N70 mobile phone. It is important to note that as described in Chapter 4.2.1, the developer can incorporate standard HTML elements within the custom wrap tags. Furthermore, it was described how some of the rendering is left to the mobile device browser.

This is evident in Figure 5-17 where the browser has rendered and resized the "crown" logo image, shown in the top left hand corner of this graphic.

Figure 5-18 Screenshot of a Blackberry 8900 showing the rendered page

Figure 5-18 shows the rendered site in a Blackberry 8900 smartphone, as this Blackberry has a more advanced browser than the above Nokia N70. This rendering demonstrates the adaptability of the outputted code to be manipulated by the device browser. It can be noted that HTML image references are kept in the output code, and can be shown on the browser, as seen in the above case.

Furthermore, site functionality was maintained, and this is shown in Figure 5-17 where the browser has recognized and highlighted the active hyperlinks. Testing of these links resulted in navigation to the other pages within the site; these new

pages were also actively transformed for mobile use. Therefore, the "click" event was recognized and correctly transferred to the server.

The transformation of a traditional website to a mobile website proved to be completely transparent to the user, requiring no extra effort to have the mobile version of the site displayed. Despite the successful transformation of function and content to the mobile device, further investigation and testing showed that the HTML page mark-up was not valid. This is investigated in the next section.

5.6.2 HTML mark-up validation

During further investigation, it was found that the use of custom HTML structure tags to define the elements of the traditional website resulted in the webpage to no longer be valid XHTML 1.0 Transitional.

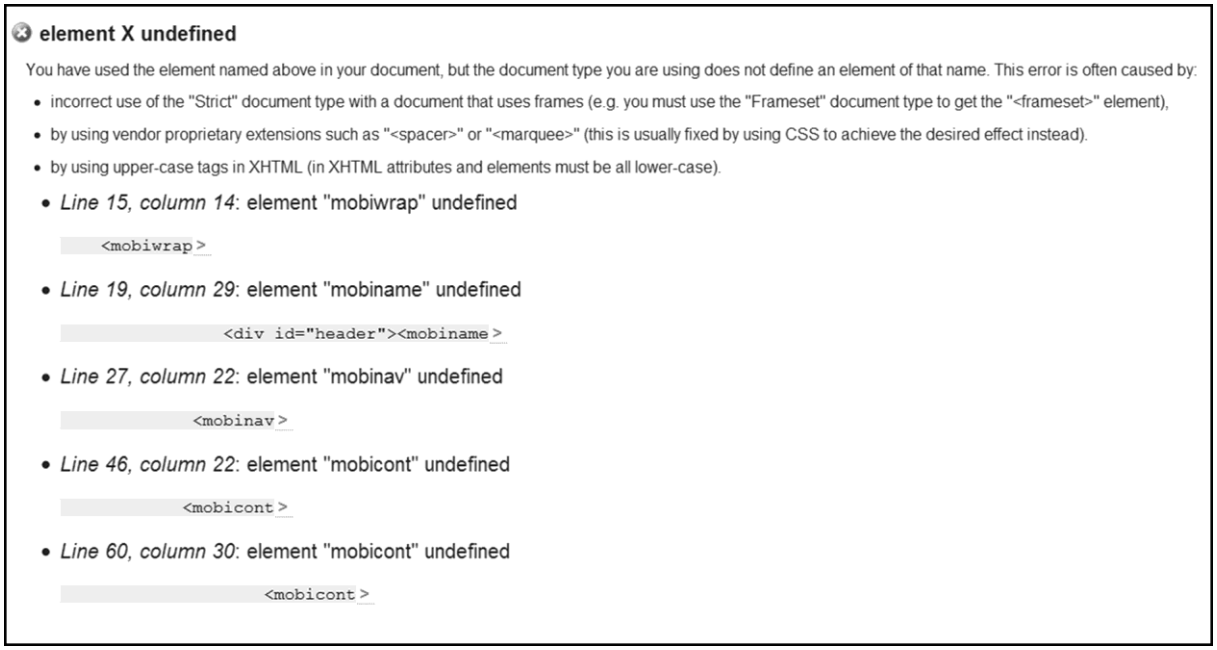


Figure 5-19 W3C validation result

The online validation tool provided by the W3C validated various prototype HTML pages. The validation failed on all pages using the custom HTML tags. The resulting error list described the custom structure tags as undefined according to the XHTML 1.0 Transitional requirements.

Figure 5-19 shows a screen shot of the validation result of the webpage when validated with the W3C validator located at http://validator.w3.org/#.

Therefore, the use of this mobile transformation system will invalidate the page mark-up. This must be taken into account if the webpage needs to be designed according to the strict XHTML 1.0 Transitional requirements.

During prototype testing, this failure did not affect the presentation of the page when viewed with a modern traditional browser.

5.7 Conclusion

The transformation of content from a traditional website to a mobile website presents a web developer with many challenges. Chapter 4 laid the foundations for a transformation method that allows a website to adapt its output if the requesting device is a mobile device. Furthermore, Chapter 4 discussed the use of custom HTML tags to allow the web developer to select appropriate content for mobile transformation. This chapter built on the foundations laid in Chapter 4, and developed a series of prototypes to further develop the concepts. It was clear that concepts proposed in Chapter 4, such as the custom tags, are valid.

During prototype development, the transformation engine was developed and refined. Furthermore, two approaches to implementing this engine were attempted. The resulting prototype was tested on a small case study and was shown to operate as envisaged.

The resulting successes from the prototype developments show that the transformation from a traditional website to a mobile website using an adaptive

output is possible. Furthermore, it was shown that an existing website could be modified to include this transformation method.

The next chapter will combine the foundations laid out in Chapter 4 and the prototype discussed in this chapter to develop a framework for transforming both content and function originally aimed at desktop browsers to mobile browsers.

Chapter 6. Proposed framework

Chapter 4 put forward a foundation for the development of a system whereby a traditional website can adapt its output rendering to be compatible with a mobile device. The solution consisted of using a set of custom tags. These tags allowed the web developer to define areas of a traditional website to be transformed. This technique allowed developers to prevent mobile incompatible components of the site transforming to the mobile device.

Chapter 5 built upon the foundation discussed in Chapter 4 by developing a series of prototypes. The prototypes facilitated the development of code that transformed traditional websites embedded with the custom tags to a mobile ready format. A small case study was conducted by implanting the concept with a pre-existing website template. During implantation, a series of guidelines and approaches were developed. The lessons learned, guidelines and approaches developed in Chapter 5 are combined with the original concept proposed in Chapter 4, to create a framework. The objective of this framework is to enable web developers to use and build upon the concepts described in Chapter 4 and Chapter 5 to transform a traditional website into mobile ready website.

6.1 Framework overview

In order for this framework to be able to assist web developers in transforming function and content to mobile devices, this framework requires that the following technical implementation requirements to be in place. This framework is based on the HTTP stream interception and modification method described in Chapter 5.4. While the transformation engine can be customized and further functionality added by the web developer, this framework requires that the overall processes follow the architectural overview shown in Figure 6-1.

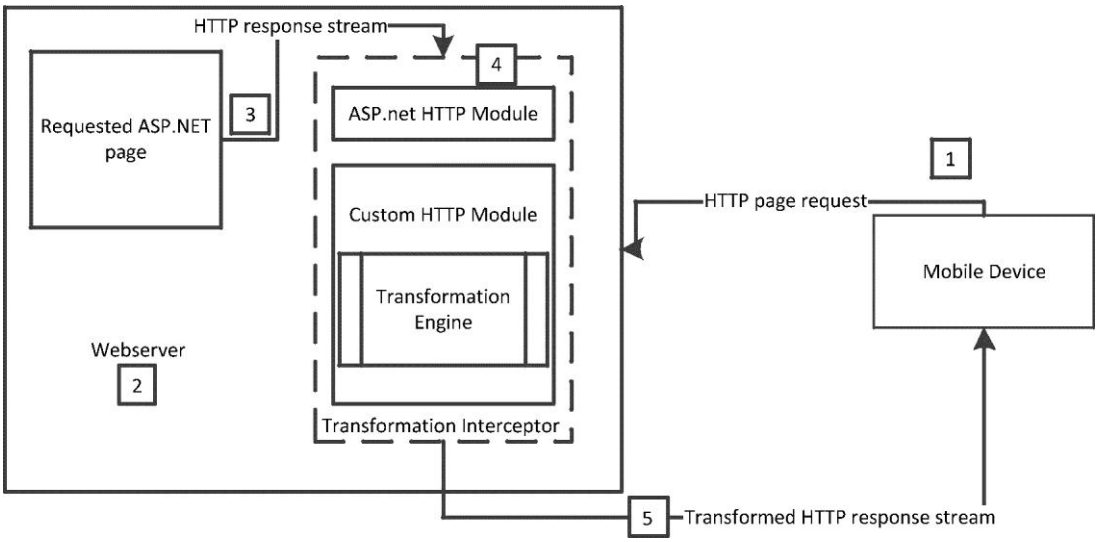


Figure 6-1 Architectural overview

Figure 6-1 shows the mobile device requesting an ASP.Net webpage from the IIS webserver (annotation 1). The webserver (annotation 2) processes the request and retrieves the requested page. This page is passed to the HTTP response stream with the intention of delivering it to the requesting device (annotation 3). The transformation interceptor intercepts the HTTP Response Stream and passes it to the transformation engine where the transformation will occur (annotation 4). The transformed HTTP response stream is returned to the mobile device (annotation 5). Based on the process overview shown in Figure 6-1, the framework discussed would be able to be used to assist in the transformation of function and content.

Figure 6-2 shows an overview of the mobile transformation framework. This framework is developed from the foundational concept presented in Chapter 4 and the prototype input attained in Chapter 5. The result is the development of phases that the development of the website will pass though in order to become mobile compatible. Furthermore, this overview shows which processes are

influenced by specific guidelines outlined in this research. These phases are described in the following subsections.

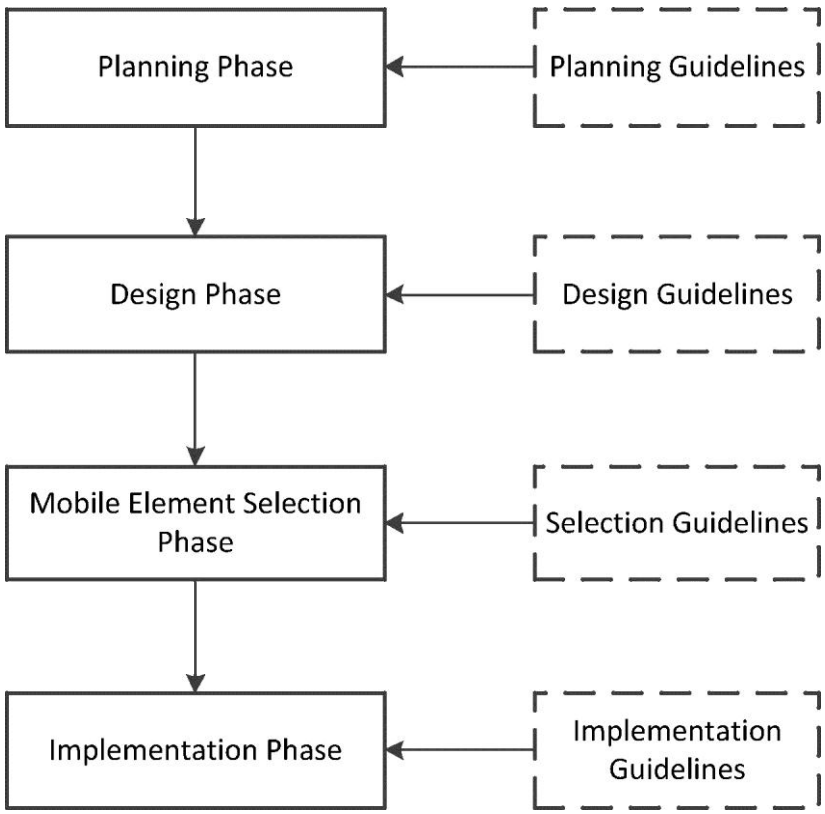


Figure 6-2 Mobile transformation framework overview

6.1.1 Planning phase

The planning phase is the initial phase that spawns the idea for a particular website. Website specifications, design and functionally goals are defined at this stage. Furthermore, a target audience is specified and delivery mechanisms, such as desktop, mobile device or both are decided upon.

6.1.2 Design phase

Once the conception phase is complete, the process moves on to the design phase.

The design phase incorporates the specifications and goals developed in the conception phase. Functionality and the user interface are developed during this phase. It is important to note the website is developed with the intention of displaying on a traditional desktop at this time. However, this phase is influenced by the following guidelines to facilitate a successful mobile transformation.

6.1.3 Mobile element selection phase

Once the website design phase is completed, and the guidelines have been considered, the process moves to the mobile element selection phase. The selection of the mobile elements is based upon the concept of custom HTML tags. These tags delineate areas of the website and this is discussed in Chapter 4.2. During the development of the prototypes, the following selection guidelines where developed.

6.1.4 Implementation phase

Once the appropriate mobile elements have been selected, the process moves on to the implementation phase. The implementation phase is the final phase in this framework and provides recommendations on how to implement the transformation code.

Each of these phases has a set of guidelines that can assist the web developer in developing each phase. These guidelines are discussed in the sections below.

6.2 Guidelines

6.2.1 Planning guidelines

These guidelines are intended to be reviewed during the planning phase of the website development.

6.2.1.1 Retrofitting an existing webpage

The conceptual design of this mobile transformation system allows for a web developer to covert a previously developed website. This ability is possible because of the use of custom structure tags. In theory, these tags are able to be inserted into a current document with no adverse effects on rendering of the website in traditional browsers. However, during development, it was found that some visual elements designed for use in a traditional browser did not work well on mobile screens. Therefore, if the original website author did not intend for the site to be displayed on mobile devices, modification may be necessary. This must be taken into account during the conception phase, as it will decide the approach going forward.

6.2.1.2 Mark-up validation

Prototype development showed that the use of custom tags violated HTML validation; this was discussed in Chapter 5. It is recommended that the web developer take this into account during development.

6.2.2 Design guidelines

These guidelines are intended to be reviewed during the design phase of the website development.

6.2.2.1 Cascading style sheets

Cascading Style Sheet (CSS) is a style language that allows authors to attach a style (eg: fonts and spacing) to a structured document, by separating the presentation style of the documents from the content of the document (Bos, Çelik, Hickson, & Wium lie, 2009).

Prototype testing showed that some mobile devices support CSS capability more than others do. In some instances, it was found that some browsers, such as those on the Nokia N70 did not support CSS fully. Furthermore, some instances were found that some modern phones, such as the Blackberry 9000 running firmware version 5 , displayed the CSS defined elements differently than the Blackberry 9800 running firmware version 6. While it is important to bring in as much visual style as possible from the traditional site, some visual style effects that work on traditional browsers did not work effectively on the mobile device

6.2.2.2 Dynamic elements

Many browsers have the ability to create dynamic visual elements. This ability is a functionality of CSS that allows web developers to create text animations, and other effects. The dynamic pseudo-class: hover applies styling to an element when a pointing device designates it but does not activate it (Bos, Çelik, Hickson, & Wium lie, 2009). In the case of the test site, this technique was applied to the navigation links, to display a different image when an element was hovered over by the user's mouse. During testing of this site on the Blackberry 9800, it was found that a delay occurred between the time an element was hovered over and when the image was displayed. While not detrimental to the function of the site, it does affect the perceived usability aspect of the mobile site.

Therefore, it is recommended that techniques that rely on dynamic images or elements to be loaded should be avoided by the web author in the design of the traditional website in order not to "filter" though to the mobile version.

6.2.2.3 Font

Font sizes and styles are frequently used to enhance visual appeal in web design. Fonts are often set in a CSS style sheet attached to the webpage. During prototype testing, it was found that some of the modern mobile device browsers, such as the Blackberry 9000 and Blackberry 9800, attempted to replicate and render the font styles and sizes as used on the traditional site. However, this resulted in some fonts being displayed properly, while others were not. Furthermore, testing showed that some browsers, such as the Nokia N70, ignored the CSS font styles and resorted to using browser default fonts.

As the design of the mobile transformation system attempts to bring as much visual style to the mobile device as possible, it is recommended that the web author try to avoid fancy, nonstandard font and sizes in the design of the traditional site. Furthermore, it is recommended that a process of trial and error be followed in order to find a compromise, between a visually appealing font on the traditional website and what renders as a "good" font on the mobile device.

6.2.2.4 Element widths

The design of the mobile transformation system allows for HTML to be embedded within the defined mobile element blocks. This gives the possibility for the mobile device browser to render visual styles based on some elements of the attached CSS. The rendering of many element positions and sizes within the browser are calculated with respect to the edges of a rectangular box called a containing block. This block is defined by the attached CSS (Bos, Çelik, Hickson, & Wium lie, 2009). Based on this, it is recommended that elements have a defined width that is a relative percentage of the containing block, rather than an absolute value. This will allow fluid resizing of the elements should the

mobile browser try to render the elements widths strictly to the defined CSS width attributes.

6.2.2.5 Multimedia

As discussed above, the design of the mobile rendering engine allows for HTML to be embedded within the mobile elements. This allows the mobile device browser to load images or multimedia as required. However, during prototype development, it was found that some images and media are resized by the mobile device browser. This functionality can result in small images being displayed on the mobile device. Furthermore, the capabilities to display animated images seem to be entirely device specific. The use of images are recommended, for information proposes and design purposes; however, it must be taken into account that some mobile devices may be incapable of displaying them correctly. Therefore, the selection of images in the traditional website for mobile transformation should be carefully considered.

Once the design phase of the website is completed, and the guidelines have been taken into account, it is recommended that basic functionality is tested on a traditional desktop computer. This is to ensure that the design is correct, before implementation of the mobile tags.

6.2.3 Selection guidelines

In order to guide the web developer in choosing elements that would be best suited for mobile devices, the following thought process has been developed. Following this thought process; the developer should be able to identify elements that stand the best chance of effectively rendering on a mobile device. Therefore, the web developer should ask the following questions about every element of the website.

Is this element a critical part of my sites business logic?

The web developer needs to determine if a candidate element of a webpage is critical to achieving the main goal of the website.

Does this element provide additional functionality?

Functions such as a search function are not critical elements but provide additional functionality.

Does this element rely on "non-standard" browser functionality?

Non-standard browser functionality often relays on third party elements such as JavaScript libraries or Adobe flash elements to function.

Does this element exist for "look and feel" function only?

Some web design elements such as animations or non-critical images do not affect the core function of the site.

Figure 6-3 shows the thought process as a flow chart that can be followed in order to assist in determining the mobile viability of an element.

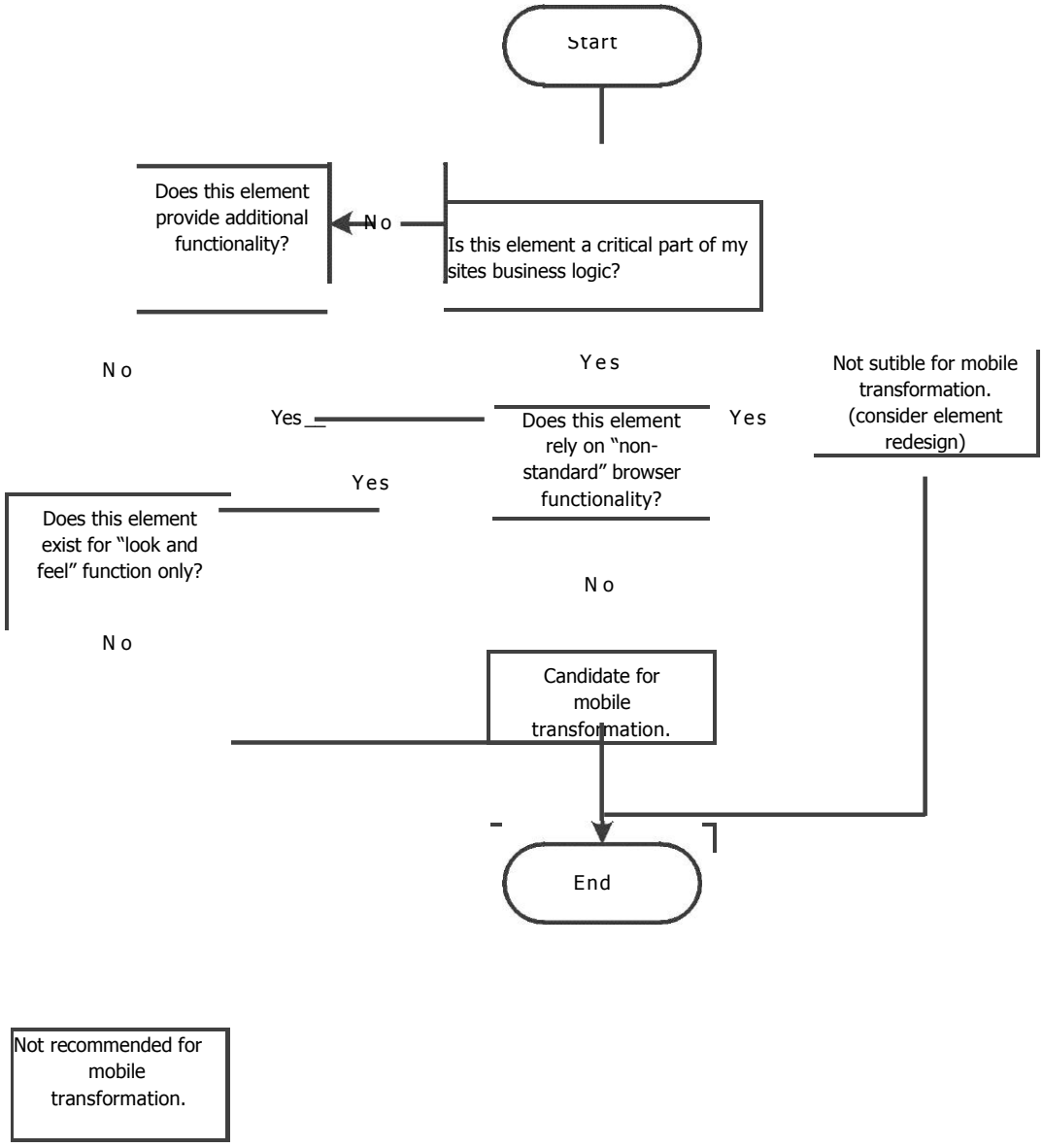


Figure 6-3 Flowchart showing element selection process

6.2.4 Implementation guidelines

As discussed in Chapter 5, the ideas presented in Chapter 4 provided the foundation for the development of prototypes further built upon by the development of prototypes. These prototypes were developed on the ASP.Net platform. Therefore, these guidelines are related to experiences encountered using this technology. However, should another technology be decided upon, these guidelines should be modified appropriately.

6.2.4.1 ASP.Net trust level

In order to provide application security, ASP.Net makes use of a trust level approach. Depending on the trust level of an application, it can access specified resources or execute specified processes. During prototype development, it was found that the transformation code was only able to operate if it was granted the "FULL TRUST" level on the server. Tests were performed on a server running under the "Medium Trust" or "Modified Medium Trust" level and were unable to run the transformation code. Therefore, it is important for the web developer to consider this aspect when implementing the transformation system.

6.2.4.2 ASP.Net Site integrity

The design of the transformation system modifies a HTTP stream from the server. The interception and subsequent modification to the HTTP stream may result in the integrity of the HTTP stream becoming lost. This may result in unexpected output to the mobile device. Therefore, it is recommended that the site be tested for correct operation on the mobile device, under a variety of conditions and situations.

6.2.4.3 Browser definition file

Chapter 5 discussed the use of the bowser definition file to provide mobile device browser detection capability. Due to the changing mobile landscape, new mobile devices are introduced. Therefore, it is recommended that the browser definitions are updated frequently to ensure correct mobile device detection.

Once the transformation code is implemented, it is recommended for the application to be tested on both the traditional desktop and a wide range of mobile devices. It is important to ensure that mobile device detection does

occur, and that the transformed site is indeed delivered to the mobile device as intended.

6.3 Conclusion

The process of creating a mobile compatible website is challenging; therefore, a method to transform a website to become mobile compatible was created. In order to assist the web developer in implementing this mobile transformation method, a framework was developed. The framework introduced four development phases that the web developer should follow during the development of the website. These phases consisted of various activities that would contribute to the development of the website. A series of guidelines and recommendations influence the development of a specific phase. These guidelines were developed upon the foundation of the transformation concept discussed in Chapter 4 and expanded upon by prototype development in Chapter 5. The guidelines and recommendations influencing a specific phase are intended to assist the web developer in the successful completion of that phase, and therefore contribute to a successful mobile transformation.

Chapter 7. Conclusion

The widespread use of mobile devices to access the Internet presents challenges to web developers. This has resulted in the need to enable a website developed for traditional media such as desktop computers to be able to be viewed and function on a mobile device. Current methods of developing separate mobile compatible websites are often not viable for web developers. Therefore, this research presented a concept for the transformation of function and content of a traditionally designed website to mobile devices. This research developed a set of custom HTML tags to select areas of a traditional website. These selected areas consisted of content that could be transformed to a mobile device. Prototyping of this method of mobile transformation resulted in the development of a framework consisting of guidelines. This framework is intended to assist web developers who use this approach of mobile transformation.

This chapter will conclude this research by reviewing the chapters, research questions and goals. Finally, this research will discuss areas that may be suitable for further study.

7.1 Chapter review

Chapter 1 provided the motivation behind this research. A brief introduction to the mobile device environment and restrictions mobile devices impose was presented. A research question was proposed and related sub-questions were discussed. Furthermore, the chosen research methodology was introduced.

Chapter 2 provided an overview of problems facing mobile devices, as well as discussing mobile operating systems. Furthermore, this chapter started to delineate the research towards the mobile web with the discussion of web specific technologies and development.

Chapter 3 discussed some of the challenges faced by mobile web designers. Current practices and guidelines for mobile web development were discussed. This chapter provided the foundation for the development of the mobile transformation concept discussed in Chapter 4.

Chapter 4 proposed a foundation for the transformation framework by using techniques to assist in the transformation of a traditional web site to the mobile web. Concepts, such as the custom HTML tags and the transformation engine, were proposed in this chapter.

Chapter 5 built on the foundation proposed in Chapter 4 by developing prototypes. These prototypes validated the concepts proposed in Chapter 4, furthermore; the transformation engine was tested using a small case study.

Chapter 6 created the framework that was developed from the results of Chapters 4 and 5. This framework proposed the development phases through which a website will pass. Furthermore, guidelines and recommendations were made for each phase.

7.2 Review of research questions

The primary question posed by this research was:

> **"How can current mobile web guidelines, practices and technologies be implemented to transform the function and content of web applications developed for desktop browsers to fit the capabilities of mobile web browsers?"**

In order to answer the primary question, this research had to answer the following sub-questions.

- **What are the capabilities of mobile phones compared to the desktop environment?**

- **How can desktop to mobile content transformation be accomplished?**
- **How can desktop web application function be transformed to the mobile web?**

Research into the first sub-question resulted in the study of mobile device capabilities and limitations. These capabilities and limitations were discussed in detail in Chapter 2, and further references to mobile device capabilities were made in Chapter 3. The remaining two sub-questions and primary question were answered though creation of the conceptual model and subsequent model verification through prototype development.

The analysis of these questions enabled the creation of a transformation solution that enabled traditional websites to be transformed to match the technological capabilities and restrictions of mobile devices.

It is believed that the resulting model and subsequent prototype have created an environment whereby the mobile transformation can be seamlessly achieved. Therefore, it can be stated that the primary research question and sub-questions have been answered.

7.3 Review of research objectives

As discussed in Chapter 1.3, this research set out to achieve the following objectives. The main objective of this research was as follows:

> **To develop a framework for content and functionality transformation of desktop based web applications to the mobile web.**

In order to accomplish this primary objective, the following sub-objectives needed to be achieved.

- **Develop a set of guidelines for the selection of critical content and functionality that must be transformed from the traditional web application to the mobile web.**

- **Develop a set of guidelines for the implementation of the selected content in the mobile web.**

- **Develop a set of guidelines for the implementation of function for the mobile web.**

The first sub-objective was analyzed during the development of the conceptual model as discussed in Chapter 4. This objective was further built upon during the prototype development in Chapter 5. Furthermore, the prototype development resulted in the creation of guidelines that achieved the second and third objectives. Chapter 6 combined these guidelines and this resulted in the development of a framework for mobile transformation, thereby achieving the primary objective.

7.4 Review of research methodology

The development of the framework followed the research methodology described in Chapter 1. In order to assess the research merit of the development it is appropriate to consider how the Design Science Guidelines described by Henver, Ram and March (2004) where met. Therefore, consider each of the guidelines in turn.

Guideline 1: Design as an Artifact

This research produced an artifact in the form of a framework. This framework consisted of guidelines and techniques for the transformation of function and content from a traditional website to mobile compatible website.

Guideline 2: Problem Relevance

The relevance of the business problem was clearly illuminated in Section 1.1. Essentially, current methods of developing mobile compatible versions of websites can prove to be unfeasible.

Guideline 3: Design Evaluation

Chapters 4 and 5 describe two prototypes that were developed in building and refining the resultant framework. Section 5.6 also describes a case study where it was shown that a traditionally developed webpage can be "fitted" with the tags described in Chapter 4 and transformed successfully.

Guideline 4: Research Contributions

The proposed framework represents the primary contribution of this research. In addition, the framework is supported by sets of guidelines that can aid in the implementation of the framework.

Guideline 5: Research Rigour

The experiences, successes and failures encountered during the development of the two prototypes were meticulously documented, interpreted and included in the final framework. Further exploration is possible, but the scope and time constraints on a master's level study did not allow for this.

Guideline 6: Design as a Search Process

The proposed framework was designed, based on two prototype iterations that commenced partially parallel to a literature survey. Furthermore, during development of this framework various options were considered, the hallmark of a search process.

Guideline 7: Communication of Research

This research is communicated in detail in this dissertation. In addition, an academic paper is in preparation, which will be submitted to an appropriate conference.

Based on this reflection, it can confidently be stated that this research adheres to the Design Science guidelines by Henver, Ram and March (2004). Therefore, this framework was developed through the use of a sound research methodology.

7.5 Contribution of this research

The primary motivation of this research was to develop an approach that allows a web developer to easily convert a website developed for a traditional PC to a mobile device compatible version. Presently, a mobile website required a separate development stream resulting in a separate mobile version of the website. This type of approach was often costly and time consuming. Furthermore, this was often an extra burden, as a traditional website was often already built and functional.

This research contributed a mobile transformation framework. The use of this framework enables the web developer to create a single source file for use by both mobile and traditional devices. Furthermore, this Framework consisted of mobile development phases and guidelines to assist the web developer in application of the transformation approach. It is envisaged that this framework be integrated during the initial website development in order to reduce the traditional resource cost associated with mobile device compatibility.

The further development and use of this framework will assist in more websites both large and small aching mobile device compatibility. This can result in these websites reaching a wider audience than was previously the case.

7.6 Areas for further research

This research concentrated on developing a framework for mobile transformation of traditional websites. These techniques were validated through the development of a prototype and the implementation of a small case study. It is believed that there is potential for more research and expansion into this approach of mobile transformation. Potential areas of further research and development are discussed below.

Dynamic image resizing - as more and more web sites are consisting of images, often in the form of photo galleries. Further research could be done to develop a mechanism for the mobile transformation code to dynamically resize images from a traditional website for mobile devices. Furthermore, this could be expanded to incorporate media elements, such as Adobe Flash and audio files.

Custom tags expansion – this concept makes use of a basic level of custom tags to delimitate mobile elements within a traditional site. This could be expanded to allow tags for other features and functions. Investigation could be done into the possible adoption of device specific tags, for example delineating elements for smartphones use only.

Mobile device detection – additional research could be conducted into the methods that the mobile transformation code could use to detect mobile devices. This could include support for the mobile transformation code to support a full CC/PP device profile as described in Chapter 3.

Request Security – design of the mobile transformation system relies on interception and modification of the outgoing request. Research can be conducted into the security and integrity of this intercepted request.

In concluding, it is hoped that this research will enable both new and existing websites to become mobile compatible, in order to embrace the exciting growth opportunities the mobile device presents.

References

A J Gold Associates. (2008). ***Wireless Push Email for the Smaller Bussiness: A Comparison - White Paper.*** Retrieved October 21, 2010, from http://na.blackberry.com/solutions/types/Push_Email_for_the_Smaller_Business_A_Comparison_Whitepaper.pdf

Adobe. (n.d). ***Rich Internet Applications.*** Retrieved November 3, 2010, from Adobe Flash Player: http://www.adobe.com/products/flashplayer/

Android. (2010, November 1). ***What is Android***. Retrieved November 2, 2010, from Android Developers: http://developer.android.com/guide/basics/what-is-android.html

Apple inc. (n.d). ***Apple iOS***. Retrieved November 1, 2010, from Apple.com: http://www.apple.com/iphone/ios4/

Asif, S. Z. (2007). ***Wireless Communications Evolution to 3G and Beyond.*** Artech House Publishers.

Berners-Lee, T., & Connolly, D. (1995, November). ***Hypertext Markup Language - 2.0***. Retrieved October 8, 2010, from http://tools.ietf.org/html/rfc1866

Bilton, N. (2010, April 29). ***Thoughts From Steve Jobs on Flash.*** Retrieved November 4, 2010, from New York Times: http://bits.blogs.nytimes.com/2010/04/29/thoughts-from-steve-jobs-on-flash/

Blackberry. (n.d). ***Blackberry - Why Blackberry ?*** Retrieved November 1, 2010, from Blackberry.com: http://na.blackberry.com/eng/developers/whyblackberry.jsp

Bos, B., Çelik, T., Hickson, I., & Wium lie, H. (2009, September 8). ***Introduction to CSS 2.1.*** Retrieved October 10, 2010, from The World Wide Web Consortium: http://www.w3.org/TR/CSS2/intro.html

Buchanan, G., Jones, M., Thimbelby, H., Farrant, S., & Pazzani, M. (2001). Improving mobile internet usability. ***10th International WWW Conference*** (pp. 673-680). New York: ACM Press.

Burigat, S., Chittaro, L., & Gabrelli, S. (2008). Navigation techniques for small-screen devices: An evaluation on maps and web pages. ***International Journal of Human-Computer Studies, 66***(1), 78-97.

Chen, T., Yesilada, Y., & Harper, S. (2010). What input errors do you experience? Typing and pointing errors of mobile Web users. ***International Journal of Human-Computer Studies, 68***(1), 138-157.

Christos, B., Giorgos, K., & Ioannis, M. (2007). A web content manipulation technique based on page Fragmentation. ***Journal of Network and Computer Applications, 30***, 563-585.

Dalal, N., Quible, Z., & Wyatt, K. (2000). Cognitive design of home pages:next term an experimental study of comprehension on the World Wide Web. ***Information Processing & Management, 36***(4), 607 - 621.

Dillon, A., Richardson, J., & Mcknight, C. (1990). The effects of display size and text splitting on reading lengthy text from screen. ***Behaviour & Information Technology, 9***(3), 215-217.

Duchicky, R. L., & Kolers, P. A. (1983). Readability of text on visual display terminals as a function of window size. ***Human Factors, 25***, 683 - 692.

Fring. (2007, January 24). ***How do i use fring over Wi-Fi?*** Retrieved November 10, 2010, from Fring Knowledge Base: http://www.fring.com/support/index.php?_m=knowledgebase&_a=viewarticle&kbarticleid=58

Gartner. (2010, November 10). ***Gartner Says Worldwide Mobile Phone Sales Grew 35 Percent in Third Quarter 2010; Smartphone Sales Increased 96 Percent.*** Retrieved November 23, 2010, from Gartner Newsroom: http://www.gartner.com/it/page.jsp?id=1466313

German, K., & Cha, B. (2010, April 10). ***Comparing smartphone operating systems.*** Retrieved November 29, 2010, from CNET.com: http://www.cnet.com/8301-17918_1-20002198-85.html

Google Code. (2010, November 22). ***Issues - Android - Project Hosting on Google Code.*** Retrieved November 22, 2010, from Google Code: http://code.google.com/p/android/issues/list

GSM Arena. (2010, Febuary 10). ***Touch web browser mega shootout.*** Retrieved November 20 November, 2010, from GSM Arena: http://www.gsmarena.com/browser_shootout-review-448p6.php

Henver, A. R., Ram, S., & March, S. (2004). Design Science in Information Systems Research. ***MIS Quarterly, 28***(1), 75-105.

Jones, M., Marsden, G., Mohd-Nasir, N., Boone, K., & Buchanan, G. (1999). Improving Web interaction on small displays. ***Computer Networks, 31***, 1129 - 1137.

Kirkup, M. (2010, Febuary 2). ***BlackBerry WebKit Browser Preview at Mobile World Congress***. Retrieved November 9, 2010, from Blackberry developers blog: http://devblog.blackberry.com/2010/02/blackberry-webkit-browser-preview-at-mobile-world-congress/

Klyne, G., Reynolds, F., Woodrow, C., Ohto, H., Hjelm, J., Butler, M., et al. (2004, January 15). ***Composite Capability/Preference Profiles .*** Retrieved August 12, 2010, from W3C Recomendation: http://www.w3.org/TR/CCPP-struct-vocab/

Koh, D. (2010, Febuary 18). ***Q&A: Microsoft on Windows Phone 7 Series.*** Retrieved November 4, 2010, from CNET - Asia: http://asia.cnet.com/reviews/mobilephones/0,39050603,62061278,00.htm

Mahmoud, Q. (2004, October). ***Getting Started With Composite Capabilities/Preference Profiles and JSR 188.*** Retrieved August 10, 2010, from Sun Developer Network: http://developers.sun.com/mobility/midp/articles/ccpp/

Microsoft Corperation. (2008, March 3). ***Microsoft's Interoperability Principles and IE8***. Retrieved November 9, 2010, from MSDN Blogs: http://blogs.msdn.com/b/ie/archive/2008/03/03/microsoft-s-interoperability-principles-and-ie8.aspx

Microsoft Corporation. (2007, Febuary 23). ***ASP.NET HTTP Modules and HTTP Handlers Overview.*** Retrieved October 12, 2010, from Help and Support: http://support.microsoft.com/kb/307985

Mitchell, S. (2008, December 3). ***Modifying the HTTP Response Using Filters***. Retrieved October 12, 2010, from 4GuysFromRolla.com: http://www.4guysfromrolla.com/articles/120308-1.aspx

Mozilla Foundation. (2010, Febuary 6). ***Gecko FAQ - MDC***. Retrieved November 10, 2010, from Mozilla developer center: https://developer.mozilla.org/en/Gecko_FAQ

MSDN. (2010a). ***Windows Mobile.*** Retrieved November 4, 2010, from Microsoft Developer Network: http://msdn.microsoft.com/en-us/library/bb847935.aspx

MSDN. (2010b). ***Using Comments Effectively***. Retrieved November 5, 2010, from Microsoft Developer Network: http://msdn.microsoft.com/en-us/library/aa164524%28office.10%29.aspx

Nokia. (n.d). ***Symbian Platfrom.*** Retrieved November 2, 2010, from www.forum.Nokia.com: http://www.forum.nokia.com/Devices/Symbian/

Nuance Communications. (2007). ***T9 Text input***. Retrieved December 5, 2010, from Nuance Communications: http://www.t9.com/us/learn/

Opera Software. (2010, November 24). ***Generation Y chooses the mobile Web.*** Retrieved November 26, 2010, from Opera Software: http://www.opera.com/press/releases/2010/11/24_2/

Parush, A., & Yuviler-Gavish, N. (2004). Web navigation structures in cellular phones: the. ***International Journal of Human-Computer Studies, 60***(1), 753-770.

Passani, L. (2010, March). ***Global authoring practices for the mobile web. Version 1.0.4***. Retrieved April 21, 2010, from Passani: http://www.passani.it/gap/

Paterno, F. (2003). Understanding interaction with mobile devices. ***Interacting with Computers, 15***(1), 473-478.

Ragget, D., Le Hors, A., & Jacobs, I. (1999). ***HTML 4.01 Specification.*** Retrieved November 20, 2010, from HTML 4.01 Specification.

Research in Motion. (2010, July 20). Retrieved August 16, 2010, from Blackberry.net: http://www.blackberry.net/go/mobile/profiles/uaprof/7750/3.7.3.rdf

Rukzio, E., Noda, C., De Luca, A., Hamard, J., & Coskun, F. (2008). Automatic form filling on mobile devices. ***Pervasive and Mobile Computing, 4***, 161-181.

Schilit, B., & Bickmore, T. (1997). Digestor: device-independent access to the World Wide Web. ***Computer Networks and ISDN Sytems, 29***, 1075 -1082.

Shepherd, G. (2005, January). ***The ASP Column - Determining Browser Capabilities in ASP.NET .*** Retrieved October 10, 2010, from MSDN Magazine: http://msdn.microsoft.com/en-us/magazine/cc300549.aspx

Siddqui, M. A. (2002, April 17). ***HTTP Handlers and HTTP Modules in ASP.NET .*** Retrieved October 10, 2010, from 15 Seconds: http://www.15seconds.com/issue/020417.htm

Talukder, A. K., & Das, D. (2010). Mobile web for under-privileged in developing countries. ***Telematics and Informatics, 27***, 350 - 359.

Tian, M., Voigt, T., Naumowicz, T., Ritter, H., & Schiller, J. (2007). Performance considerations for mobile web services. ***Computer Communications, 27***, 1097-1105.

W3C. (2008). ***About W3C***. Retrieved November 8, 2010, from World Wide Web Consortium: http://www.w3.org/Consortium/

W3C Best Practices 1.0. (2008). ***Mobile Web Best Practices 1.0.*** (J. Rabin, & C. McCathiNevile, Eds.) Retrieved October 20, 2010, from World Wide Web Consortium: http://www.w3.org/TR/mobile-bp/

Webkit.org. (n.d a). ***The WebKit Open Source Project***. Retrieved November 9, 2010, from WebKit Open Source Project: http://webkit.org/

Webkit.org. (n.d b). ***Welcome to the S60WebKit Project***. Retrieved November 9, 2010, from S60 webkit: http://trac.webkit.org/wiki/S60Webkit

Willemse, M. (2009). ***Incorporating Mobile Technology in Showroom Sales Enviroments.*** Nelson Mandela Metrolopitan Univeristy.

World Wide Web Consortium. (1997, July 8). ***HTML 4.0 Specification.*** Retrieved October 10, 2010, from World Wide Web Consortium: http://www.w3.org/TR/WD-html40-970708/htmlweb.html