Web engineering for mobile devices

Kai Hendry

Helsinki 28th February 2005

M.Sc Thesis

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta/Osasto — Fakultet/Sektion — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Kai Hendry

Työn nimi — Arbetets titel — Title

Web engineering for mobile devices

Oppiaine — Läroämne — Subject

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
M.Sc Thesis	28th February 2005	62 pages

Tiivistelmä — Referat — Abstract

Unleashing information applications to a profound audience is proposed via the popular mobile device and de facto Web technologies. Incompatible application program interfaces (API) such as WAP are criticised in favour of unifying application development on the Web API with every device supporting a capable Web user agent. The consequences of using separate APIs for differing devices are found to result in complex and costly device dependent information applications.

Misconceptions of the limitations of the mobile device are explored, in order to show the mobile as a capable information device. Milestones are plotted beginning with WAP enabled phones, basic HTML enabled mobiles of today and to a future where a mobile Web UA effectively supports a generic Web mail application.

Different Web engineering methodologies for mobiles are evaluated, with recommendations for a device independent approach in the long term and UA support via device dependent proxies in the short term.

Further research is suggested with a emphasis on the need for a scalable bitmap image format. The thesis concludes by highlighting the shared responsibility of mobile stakeholders to propel a unified computing future through the Web API.

Instead of engineering applications for the device, the thesis presents argumentation to engineer for the Web medium.

Avainsanat — Nyckelord — Keywords
Accessibility, Mobile, user interface standards, WAP, Web engineering

Säilytyspaikka — Förvaringsställe — Where deposited
Kumpulan tiedekirjasto

Muita tietoja — övriga uppgifter — Additional information

Contents

Glossary

Accesskey	A markup mechanism for defining a keyboard shortcut
API	Application Programming Interface
Browser	See UA
Browsing based navigation	Hyperlinked document-based navigation where users move through an interface without a specific goal in mind
CC/PP	Composite Capability/Preference Profile. A RDF based language for describing a device
CHTML	Compact HTML. Also written as cHTML
CLDC	Connected Limited Device Configuration
CPU	Central Processing Unit
CSS	Cascading Style Sheets, which are used to define stylistic attributes to content
Desktop	Also known as a workstation. The familiar fixed networked personal computer, with powerful processors, large displays, sound output, large amounts of memory and persistent local storage.
ECMAScript	The javascript programming language defined by the ECMA-262 standard for performing browser-side actions.
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications
HDML	Handheld Markup Language
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
I-mode	Information Mode by NTT Docomo of Japan
Inline	A Web page component such as an image or a sound accompanying the page. They are automatically downloaded as part of the page rather than those shown only when you select their link.
IP	Internet Protocol
IT	Information Technology
MIDP	Mobile Information Device Profile

MIME Multipurpose Internet Mail Extensions (RFC 2045/2046/2047/2048/2049, IETF)

MMI Man Machine Interface, or just interface

MMS Multimedia Messaging Service

Mobile Small sized, light weight wireless computing device. Primary use is that of a phone. Extended battery life.

PC Personal computers, also know as desktop PC, or just desktop

PDA Personal Digital Assistants

RAS Remote Access Service, usually in context with a Internet dialup service with a modem

RDF Resource Description Framework

Scalability A technique that is as suitable for a small amount as it is for a large amount and vice versa.

SGML Standard Generalised Markup Language

SMS Short Message Service, used for instant text messaging

Tag soup The term "handled as tag soup" refers to how UAs in reality liberally parse HTML. The parser is neither an XML or SGML parser.

Transaction based navigation Multistep goal orientated navigation for performing a set task

UA The user agent (UA) is the client application that requests a document from an HTTP server. Browsers are examples of user agents, as are Web robots that automatically traverse the Web collecting information. [Wor01]

UAProf User Agent Profiles [KRW$^+$04]

URI or URL Uniform Resource Identifiers or Uniform Resource Locater

Validator An application which is used to check the validity of HTML markup, for example to detect bad or deprecated elements. A validator helps to ensure that the document can be parsed and used by all user agents.

W3C World Wide Web Consortium

WAE Wireless Application Environment

Web Accessibility Web access by everyone regardless of disability

WML Wireless Markup Language

WWW	World Wide Web or just Web
XHTML	The Extensible HyperText Markup Language is a family of current and future document types and modules that reproduce, subset and extend HTML, reformulated in XML.
XML	XML (Extensible Markup Language) is a meta-syntax, used to create new languages
XSL	Extensible Stylesheet Language (XSL) is a language for creating a style sheet that describes how data sent over the Web using the Extensible Markup Language (XML) is to be presented to the user.
XSLT	XSL specifies the styling of an XML document by using XSLT to describe how the document is transformed into another XML document that uses the formatting vocabulary.

1 Introduction

Accessing the Web from a mobile is a Web engineering problem. The Web's presentational markup is designed for the desktop PC, which makes Web content and services difficult to scale down to a mobile. This problem blocks an important avenue of information access for many people who are more likely to have a mobile device than a desktop device. If parts of the Web are unavailable to mobile users, it will fragment and devalue the notion of the universal information interface.

The original application of the mobile device is voice communication. However the mobile device does have a limited means of input, display and growing computing power to provide a platform for an information service. Today popularity of mobile devices is only matched by the Web, where trends indicate the Web being the most important information service space [Ree02]. Facilitating Web access from a more common wireless mobile device could yield an era of information accessibility to usher in the European Union's goals of an Information Society [Eur04]. These converging worlds of the wired Web and wireless mobile raise several research problems, from how to produce, retrieve, render and interact with information in order to make the mobile a viable, usable and accessible information appliance.

The thesis is chronologically organised to assess significant past and current mobile related developments in research and realised implementations. Each generation of the mobile device is defined in the context of its typical capabilities and operating environment. Mobile software implementations of technologies providing information services are evaluated along with their troubled relationships. These application technologies include WAP's WML, GSM's SMS, NTT Docomo's CHTML to W3C's XHTML and an analysis of how they are converging [HMK98].

Engineering device independent information services [Kir02] is an integral theme for solving the thesis question of how limited mobiles and the information services of the Web converge. Research offers differing approaches to mobile information access [Mar02] [Ran03] [MPS03] [BKGM$^+$02], the W3C and the mobile industry implementations of WAP/OMA. The W3C portfolio of mobile related technologies [DKMR03] [KRW$^+$04] [BIM$^+$00] [WDSR02] [CVJ01] and working groups [Wor01] will be the benchmark when evaluating different approaches for discerning Web content.

1.1 Scope

The application layer is the focus in the thesis and hence bearer networks, network protocols and bandwidth efficiency will not be covered in detail. The application layer will focus on the markup language of the information. Backend application server logic and Web applications will not be covered. The thesis will concentrate on the European Union mobile markets, as opposed to different and difficult environments abroad. Mobile device platforms will be biased towards the market leader vendor Nokia. Mobile computing devices such as laptops are excluded from the thesis by their similarities with desktops and their comparatively short battery life. Voice based information services provided by VoiceXML and other related speech technologies will not be included in the thesis. Other advanced telephony services such as location-based services will not be addressed. Multimedia applications and proprietary plugins will be largely avoided in order to concentrate on the basic yet challenging problem of delivering text based content to the mobile.

The thesis alternates between the terminology Web author, engineer, publisher, designer, developer and so forth. They all describe the person who contributes content or services to the Web via the Web API. We take the approach that there is no separation of concerns when it comes to developing for the Web.

Making the Web available on the mobile is related to the usability field, internationalisation efforts, accessibility of the disabled, affordability, content scalability, interoperability standards, multi-modality and device independence. Notice many of these fields overlap, as device independence is defined by the W3C as “Access to a Unified Web from Any Device in Any Context by Anyone”[Wor01] demands several disciplines. This thesis concludes by offering insight into past, current and future technological trends for realising the ubiquitous information application.

2 First generation of the mobile Web

There are a wide range of mobile devices such as Personal Digital Assistants(PDAs), book readers, smart watches, pagers and car navigation systems. However in this thesis, the focus is on the popular handheld wireless cellular mobile phone device or just mobile as it is referred to here. Mobiles equipped with information services are also called “enhanced phones” by the industry [Gar04]. The mobile device or simply mobile is defined as a small battery powered wireless Internet networked computing device. Fast paced technological progress has swept the terms multimedia and network PC aside and likewise small computing devices will evolve to make the term mobile out dated. For example the mobile we know now could be permanently fixed on a kitchen wall in the near future [BL04].

The mobile’s primary purpose is for voice communication across cellular networks. There were as many as 500 million GSM users worldwide in the year 2001 and now in 2004 more than 1000 million users worldwide according to the GSM association [EMC04]. Strategy Analytics says global sales of mobile phones hit a record of 516 million in 2003, with a predicted 13% growth expected in 2004. The mobile device market leader at 34.8% is the Finnish company Nokia [BBC04]. In Europe Mobile penetration is over 70% [Web04a], with the GSM association’s 147 European members claiming to have 425 million users in December 2003.

The World Wide Web (or Web) is the universal information space of the Internet. The Web’s open distributed architecture provides information services with global identifiers called Uniform Resource Identifiers (URIs) [Jac03]. The Web in contrast is accessed typically by a powerful desktop personal computer, which is referred to and described as the desktop computer. Determining the amount of desktop Web users to compare to mobile users is difficult. One can argue the boom of the IT bubble at roughly the same time period indicates explosive growth. The latest measurements indicate the American online Internet population is at least 200 million users strong [Nie04]. In Europe figures by eMarketer [Rya04] claim that Europe will have 255 million Internet users by 2004. However mobile subscriptions outstrip Internet subscriptions all over Europe according to Gartner in 1998. Despite these tentative market statistics, we can conclude that mobile users out number desktop users making the mobile the eminent consumer computing device.

The stakeholders involved with the mobile Web are:

- Mobile device manufacturer. e.g. Nokia or Motorola
- Operating system platform provider or vendor, e.g. Symbian or GNU/Linux
- Mobile operators or network service providers. e.g. Vodafone or Radiolinja
- Web developer or author. A Web application provider such as Google or Amazon.
- User agent or browser developer, such as Opera or Mozilla
- Web user, surfer or consumer. The person who requires all of the above via a user agent to access the Web
- Standards authorities. Such as the OMA, W3C and GSM association

On desktop systems the device and platform components are blurred between the PC hardware manufacturers such as Dell and the operating system Windows with a bundled user agent Internet Explorer. Nokia arguably fill these roles, with their mobile products providing the hardware device, their operating system and often including their own in house user agent for browsing the Web.

As mobile communications have not been a part of the Internet communications, these domains have diverged by rapid developments in recent years. If the Web can be accessed by the mobile device this would allow millions of more people to have convenient access to information. Bridging this divide between mobile and the Web is hence crucial for realising an Information Society [Eur04] by allowing millions of more people access to the information interface of the Web.

The mobile was seen by the industry as a viable platform for information services as early as 1997, whilst Internet users have been up to this point using a desktop computer to access information. The next section discusses the differences between devices at this early period and subsequent sections show how initial attempts to create a mobile information appliance began.

First generation of the mobile Web	
December 1996	W3C CSS, level 1
January 1997	W3C HTML 3.2
December 1997	WAP Forum founded
February 1998	CHTML Note to W3C
April 1998	WAP 1.0
December 1999	W3C stops working on HTML
June 2000	WAP 1.2.1
Second generation of the mobile Web	
December 2000	W3C XHTML Basic
January 2002	WAP 2.0 white paper
June 2002	WAP Forum consolidated to OMA
January 2004	Composite Capability/Preference Profiles

Table 1: Timeline of the mobile information appliance

2.1 Environment

In this thesis we are concerned with access to information services via the mobile. The mobile's typical characteristics however poses hardware restrictions, that need to be considered. First and foremost, a mobile is a telecommunication device for voice communication. Most mobiles are designed to be operated as a telephone handset, held in one hand between your ear and mouth. As the device is designed for voice communication, one might consider the device better suited for accessing informational voice content. Besides the technical demands of a voice operated mobile, character based or text content is a far more efficient way to access information for a variety of reasons. Text is not time dependent as the speech equivalent is. A user can naturally choose any speed from to re-reading slowly, or skim over large amounts of text at leisure. Text is also much more versatile and can by easily manipulated. For example text can be redirected to different outputs such as a braille terminal for the visually impaired to access. Voice interfaces are often intrusive or difficult to use in crowded and noisy mobile environments.

Therefore text displays complemented even the earliest mobiles, with battery and connectivity information and typical extras such as an address book. These displays evolved to a generic 2 to 5 line display from which the mobile information appliances originated.

Most of the specifications in table 2 are especially variable with the mobiles' hardware range. The screen size in pixels is probably the most significant difference between the mobile and desktop. Mobile screen resolution area are typically less than 5% of their desktop counterparts. Knowing the screen resolution does not necessarily allow you to conclude how many lines of text or how many rows and columns the device can render. A high resolution device might display one line of text, whilst a low resolution device could tightly pack a large amount of text on the screen. However a higher resolution screen does help the legibility of text, therefore

Characteristic	*Mobile*	*Desktop*
Screen resolution	50x30 to 150x100	640x400 to 1024x768
Screen Colours	Monochrome	16 bit
Fonts	Single	Several
Software upgradable	No	Yes
SMS	Yes	No
Input	Non-standard numeric based	Qwerty keyboard + mouse
Network	Non-IP	IP
CPU Power	Low	High
Data storage	Very limited (<1M)	Large and versatile
Power supply	Limited	Unlimited

Table 2: Broad comparison between first generation Web devices [Kam98]

a higher resolution screen indicates that more text can be rendered.

First generation mobiles' displays supported text and some simple graphics too. Those that did support graphics could only render them if they were within the resolution of the mobile screen and only in reduced 1 bit format. Another limiting feature of mobile graphics displays were that sometimes the pixels were not quite square, but rectangular so authors needed to remap their image to take this into account in order to make the image look correct. Styling elements using colours and differing fonts and sizes were not well supported at this stage, including more logical styles of emphasis were not implemented either. This generation of the mobile device typically had only support for a Western European character set, which usually required text in different languages to be transliterated. The desktop user agents at this stage had far better international support.

As mobiles have a history of being difficult for users to upgrade and maintain, they are often referred to as appliances instead of machines. An appliance has restrictions whereby you throw away and replace if something goes wrong, whilst a machine allows the freedom to be maintained and repaired.

The Short Message Service (SMS) is an instant message service available on the GSM network for mobiles and not part of the Internet domain. An interoperable device independent information messaging service would been of great benefit of unifying the desktop and mobile device and pave the way to multimodal access to small texts. At least SMS has shown the mobile as capable information device in section 2.3.

Inputs on mobile phones typically have several control buttons and the number buttons (0-9). The popular WAP phone pictured in figure 1 also has a proprietary NaviRoller(tm), which assists page scrolling. Some other mobiles had touch screen interfaces. Importantly there is no standard on what or how inputs are laid out on a mobile. Whilst on a desktop a qwerty keyboard and mouse have become a de facto standard.

GSM networks allowed mobiles to restrictive access to the Internet via dial-in servers

(GSM data). The wireless characteristics of a narrow channel made connecting to the Internet a slow, unstable and expensive operation [KAIM99] compared to desktop access. The network usually was a non-IP connection, although all these connectivity restrictions largely depend on underlying communication bearers.

Figure 1: Nokia 7110 WAP mobile phone

Data memory storage limitations were an acute problem with the first mobiles equipped with WAP. Each mobile generally has its own unique size limits. This makes it extremely difficult to develop for, as the developer may find their application unable to fit on the mobile whilst it did on another. Desktops computers of the same era had many times the memory size and other significant software advantages such as virtual memory provided by the operating system. Mobile CPUs are in the 1-10 MIPS class like many other embedded systems. Desktop CPUs have hundreds of times more CPU power.

Electrical power supply of a mobile is restricted to a battery, which degrades over time. Desktops are always connected to mains AC power. Notice that CPU hungry applications consume more power. Therefore mobiles with increased processing power, also require better batteries. Mobile displays are often illuminated only briefly to efficiently make use of the power. This often proves a nuisance when reading text on a mobile.

The mobile can be and is typically operated whilst on the move, whilst a fixed desktop is operated from a desk in a sitting posture. Reading and paging through information in text demands concentration, hence it is worth considering that a mobile user may have to assume a stationary stable posture when accessing certain content too. However for the most part, the mobile is operated with one hand and the informational content is just glanced upon whilst active or resting. A mobile is also used in a hands-free environment, and is a legal requirement in several European countries when driving. This mode of operation is referred to as “eye-less”. Compared to the desktop the mobile device exhibits a different paradigm for device usage in a dynamic mobile outdoors environment.

The output differences between a mobile and a desktop is an open scalability problem. The predominant information interface of the Web should be able to scale to any output. In practise scalability means that content has to be of relative dimensions in order to be resized correctly. If the Web is unable to scale to different outputs, then a new static medium is created specifically for each device (device dependence). Each new medium for each device type risks redundant information due to incompatible or different interfaces and being simply out of date with the original content.

Ideally when you browse the Web with any device, the information would be the same. For an example compare screenshots from different device user agents in figures 12, 13, 14 and 15 at the end of the thesis. They demonstrate how the information of the Web page scales between different devices. This demonstrates device independence with no loss of information.

The term device independence is used to describe methods in which content is to be widely deployed across any terminal or device. This in turn gives the Web greater access to people who are without expensive fixed desktop computers and hence part of the larger efforts of field of Universal accessibility of information. Device independence shares principles of Web accessibility initiative (WAI) and that is defined as Web access by everyone regardless of disability. Device independence might bring the information to the mobile, but if the information is not usable such as navigable or understandable then it is said to lack usability and accessibility. Accessibility is not only about ensuring that disabled people can access information, the meaning is often broadened to ensuring that the wide variety of users and devices can all gain access to information.

In 1997 the WAP Forum was founded by GSM industry players Ericsson, Motorola, Nokia and Unwired Planet. Their task was to standardise wireless information services. This was prompted by large network operators unwilling to accept various incompatible proprietary techniques for delivering services to mobiles offered by vendors at the time. The WAP Forum is the authoritative source for the WAP specification, a standard on first generation mobile information appliances.

2.2 WAP Forum

> The Wireless Application Protocol (WAP) is the de-facto world standard for wireless information and telephony services on digital mobile phones and other wireless terminals.
>
> Main Objective of the WAP Forum (1999)

The WAP Forum is unlike Internet standards organisations such as the Internet Engineering Task Force(IETF). The body charged a preventive entrance fee for members and operated a closed design process, effectively excluding Internet engineering veterans and small businesses. The WAP Forum objectives aimed to provide an application framework leveraging digital data networking standards of mobile networking technologies and Internet technologies such as XML, URLs and content formats. Their resulting product WAP is not a single protocol, but a list of protocols and specifications for specific devices in a specific environment.

The first objective from the WAP architecture specification [WAP98a] was to bring Internet content to digital cellular phones. Unfortunately at the time WAP was touted as the "Web in your hand", incorrectly implying that anything on the Web was now available on the mobile. This was damaging, as WAP does not directly combine Internet services with the mobile device. WAP uses the Wireless Markup Language (WML), not the typical HyperText Markup Language (HTML) of the Web and WML is not compatible with HTML. WAP Forum's efforts were on providing a channel for information services on the mobile device. In effect the WAP Forum defined a new medium as if one wanted to transfer information from a printed media source, email, the Web, one would have to transform it to fit into their newly devised architecture under WML.

The second listed objective was to create a global wireless protocol specification to work in differing wireless network technologies. WAP is a set of protocols reimplementing wireless counterparts of each member of the stack of Internet protocols for wireless optimisation. The WAP Forum has been criticised for developing a new set of protocols in a short space of time that increases the chances of design, implementation and security flaws, instead of concentrating on improving the Internet Protocols [Val00]. Recall from table 2 that mobile device software is not easy to upgrade, therefore a long and expensive software testing is essential to ensure quality. It is difficult to determine why exactly the WAP Forum did not use existing widely implemented Internet Protocol whilst others such as the Japanese company NTT Docomo did. Although the technical details of how exactly NTT Docomo did this are not publicly available. The WAP Forum departed from Internet standards, separating information space by creating an incompatible divide between mobile and Internet domains.

The third objective was to enable content creation that scales across a wide range of bearer networks and device types. Content authors should not have to worry whether or not their content will work on different mobile hardware. If they did, then most authors tend to support the most popular mobiles and as a consequence neglect others. Created content should just work by scaling across all devices, without having the author to intervene. After all WAP is a standard for information publication on mobiles, therefore it should prevent cases where authors are required to fine tune the display of their content for different mobile devices. However WAP Forum's WML does not scale for reasons addressed later in this section.

The fourth objective was to embrace and extend existing standards wherever appropriate. This is a point of contention. WAP Forum did embrace standards, by basing their WAP stack on similar IP elements however they extended the original standards with wireless optimisations without necessarily contributing back to the standard's process or ensuring compatibility. WML justifies this claim as a hybrid between Handheld Markup Language (HDML) and HTML standards, that broke compatibility with both formats.

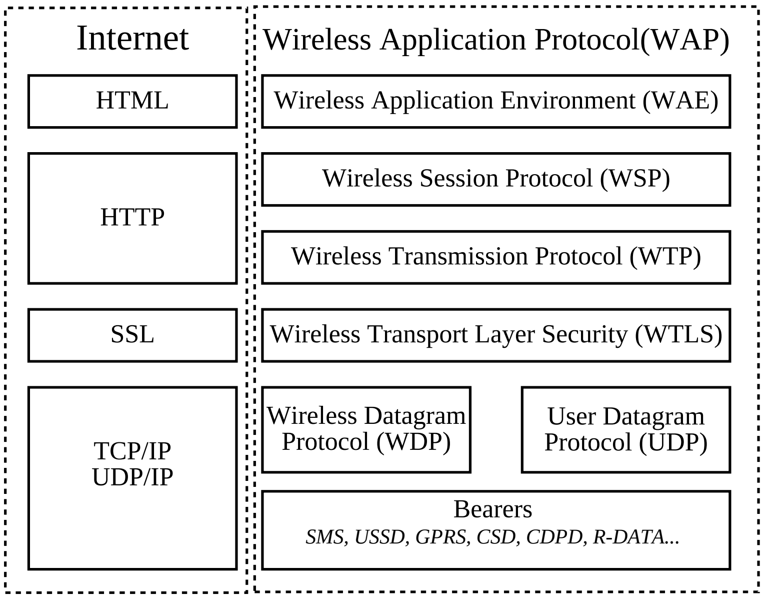

Internet
Wireless Application Protocol(WAP)
HTML
Wireless Application Environment (WAE)
HTTP
Wireless Session Protocol (WSP)
Wireless Transmission Protocol (WTP)
SSL
Wireless Transport Layer Security (WTLS)
TCP/IP
UDP/IP
Wireless Datagram Protocol (WDP)
User Datagram Protocol (UDP)
Bearers
SMS, USSD, GPRS, CSD, CDPD, R-DATA...

Figure 2: WAP architecture compared with the Internet's Web [WAP98a]

The WAP Forum defined the Wireless Application Environment (WAE) as WMLScript, Wireless Telephony Application (WTA, WTAI), content formats and Wireless Markup Language (WML). WMLScript is a complementary technology to WML, loosely based on a subset of ecmascript. WTAI offers telephony APIs to GSM network functions for example to initiate a call from a WML page. WTAI will not be discussed as there is no commonly used existing scheme on the Internet to compare to and it is not directly related to the thesis theme of Web accessibility. Besides many mobile user agents had no implementations for WTAI. On the objective for defined content formats, WAE defined phonebook, calendar formats as well as a new black and white bitmap graphics format used in WML.

The WAP Forum's specific approach to setting a standard for wireless information appliances, sets a precedent. Their organisational objectives demonstrate a departure from device independence and Internet standards.

Wireless Markup Language (WML) is a tag based document language applied to representing general information. WML is specified as an XML document type and is specifically designed to support the small narrow-band constraints of the mobile. WML is based on a subset of the Handheld Device Markup Language (HDML) 2.0 [Hyl97], which is a significant departure from the HTML standards utilised on the Internet's WWW. Presentation tags were introduced from "physical" markup elements in HTML, which contradicted W3C's policy of the time to separate content and style [LB96] and derided the importance of the logical, informational and semantic markup.

HDML as a markup language is targeted for handheld devices. It was not designed for devices such as printers, projectors, PCs and therefore can not be described as a device independent language. HTML was argued in the HDML proposal as being unsuitable mainly due to its hypertext or browsing navigation model. HDML instead proposed a transaction based navigation rather than the document-based paradigm of HTML. This is the fundamental difference between WML and HTML and it leads to several consequences that the content author needs to worry about. A content author now needs to generate another separate mobile version of their Web document and think about how a user will transact with their content in small use cases to fit the new navigation model. This forces authors to undertake the expensive costs of device dependent engineering.

HDML advocates argued that desktop browsers provide more context than mobiles could. Desktop context such as title bars, URL displays and history menus, to help users keep track of where they are. Title bars, URL displays and history functions are also implementable in WML micro browser user agents. Another argument was the assumption that machine distillation of data will lose data deemed as important by the author. Machine distillation or text summarisation at the time seemed inevitable if one wanted to use a mobile to access Web information. HDML forced authors to rethink the structure of their information in order to make it brief and accessible on a handheld device. However, in practise HDML did also suffer from verbose text problems of the Web. Despite the criticism for HDML, it did succeed to become the basis of WML.

WML inherits the deck of cards metaphor from HDML, however WML is not backwards compatible with HDML. The basic component of a deck is an atomic card, it can not be broken down to smaller parts without losing context. Cards amount to a deck, that in turn makes up an application. Cards become discrete units of data, you have types of cards for example for entering data and cards for an index listing. The reasoning is that multiple types of data on a single card would result in a loss of context. This segregation of data is said to motivate a navigation model that is multistep goal orientated (transaction) based, as opposed to the browsing navigational model of the Web. When the WAP compatible user agent on a mobile accesses a WML deck, it reads the whole deck and navigation between the cards in this deck is done without the need to load any more data. This should have made a deck application's cards quick to navigate between, although this was let down by often slow and unresponsive interfaces of mobile user agents in practise.

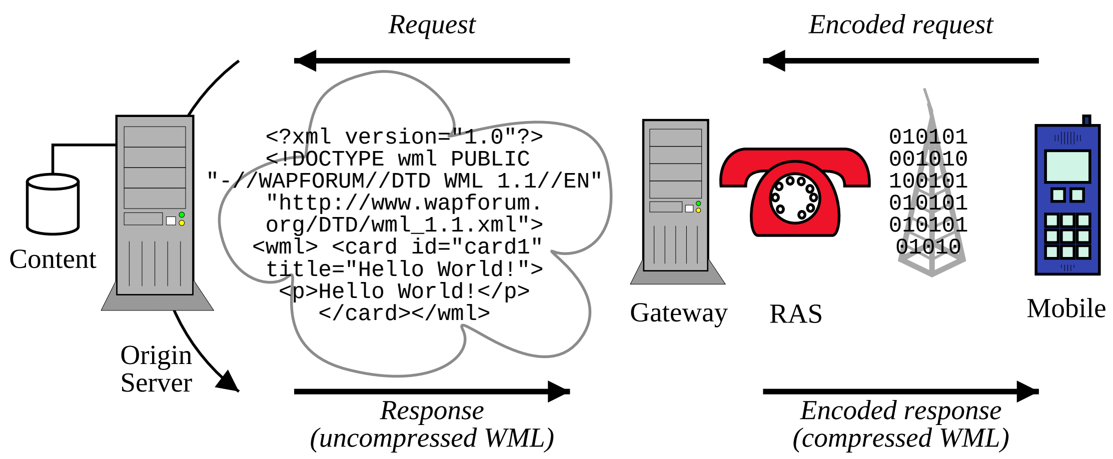
A significant problem was that the size limits of the card and the deck varied from mobile to mobile. With no concrete limits a WML application will work unpredictably and requiring authors to fine tune their application between devices. A closer look at the WML specification reveals that it was designed for Man Machine Interface (MMI) Independence [WAP98b]. Stress was put on user agents to decide how best to present a card. A card is allowed to be broken up across several units of displays. However a card is already a broken down element of a deck and now allowing further fragmentation arguably defeats an authors efforts of break down the information from a document into atomic cards. There is a underlying problem in the model and that is the size of the card cannot be declared otherwise you limit how the model scales. If screens are large then a large card size is desirable. However if using a display of the size of a wristwatch, an author would want the cards to be the size of a sentence. Hence the card and deck metaphor is device dependent and does not scale for the mobiles it was designed for.

Another consequence for meeting MMI independence was an abstract specification for layout and presentation, in order to let vendors choose how to implement the user interface. The reasoning was that user instead of using numeric inputs, could perhaps use a voice based interface. However voice based interface in practise has not been widely implemented. Authors had to now optimise their WAP services for different mobile phones, as they would never know exactly how a WAP mobile would render on different mobiles. This abstract specification resulted in implementations leaving out important navigational conventions such as the back button from mobile interfaces which harmed usability.

International support was touted by the WAP Forum, as enabling the presentation of most languages and dialects. In practise mobile phones for different markets (Europe, Asia, Americas) had different limited font sets built in. This meant that even though European WAP implementations may have understood Unicoded Chinese characters, they were unable to render them on the screen of mobiles. As a workaround authors could attempt to substitute image bitmaps representing the unavailable glyphs. However many mobile user agents could not make image glyphs hyperlinks to click on like text as they missed support for images anchors. Early mobile internationalisation which is required for universal access was poorly implemented.

Popular desktop browser user agents such as Internet Explorer [The04] have no native support for WML, therefore developing or viewing WML content from the Internet typically involved a cumbersome and inaccessible Java applet. WML was also difficult to access on a mobile device, as WML had to be strictly well formed XML otherwise the device could not process it. By comparison desktop browsers would show invalid HTML without complaint, because of error recovery routines of the “tag soup” parser. WML browsers did not have a “tag soup” parser to handle invalid markup, hence a reason why error recovery on micro-browsers are described as ungraceful. For authors programmatic errors in their content was inevitable while transferring information from the original HTML format to WML. Hence WML users would often be denied access to information. As a result, budgets for WML testing efforts by many Web design firms had to increase to accommodate this. A significant debugging problem with mobiles is that it is very difficult for a user to message the responsible party in case an error occurred. Many WML browsers were unable to show the address of where the user was, or the contact details of the Webmaster. If the mobile user knew the Webmaster’s email address, the user was unable to email and report the problem from the existing WAP implementations. On a desktop reporting a problem was possible as browser implementations were commonly integrated with an email messaging client. Not being able to recover and debug errors as well as the desktop domain does not improve WAP content.

Figure 3 shows a mobile is required to go through a gateway mechanism via a dial-in server or Remote Access Service (RAS) to access WML content from the origin server via the Internet. The gateway assembles packets and encodes ordinary text WML into tokenized and compressed WML (WMLC) for wireless transmission. This compiled compressed binary data accepted by the mobile client minimises the size of the data, as well as minimises the computational energy required by the client to process the data. Binary XML compression is a good choice for small files and low powered devices. However later developments have undermined the compiled XML binary strategy, with wide deployment and satisfactory general purpose compression with gzip [Lew03]. However poor gzip performance with small data and the local processing overhead may have been seen as too high for the environment at this time. The WAP gateway talks to the mobile using the WAP protocol stack and translates the requests it receives to normal HTTP. This assists content producers as they can use their existing HTTP Web servers, without having to implement and deploy the WAP protocols themselves. Besides optimising content for wireless transmission, the gateway acts as proxy to cache content and firewall to help protect the mobile network from intrusion. Some gateways were configured to deny access to WML pages with invalid markup, degrading the user experience by silently failing not allowing the browser or User Agent access to information. Every WAP mobile requires a gateway to access the Internet, however it is a common misconception that the gateway and other applications are controlled by the mobile operator. For example banks and other financial institutions had to put the WAP proxy within company premises due to a security vulnerability that prevented end-to-end security [Paa01]. The added third party of a WAP gateway demonstrates the complexities of putting middleware between the client and server.



```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC
"-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.
org/DTD/wml_1.1.xml">
<wml> <card id="card1"
title="Hello World!">
<p>Hello World!</p>
</card></wml>
```

Figure 3: Gateway binary compressing WML

Network operators would often employ a walled garden model, whereby their mobile customers could only access WML content they sanctioned. This blocked users from accessing a service which was not part of their provider's garden. Another usability problem required users to manually enter gateway settings or possibly change network providers all together if they wanted to switch gateways.

Mobile browser user agents, gateways and the standard WML itself all suffered from various implementation problems due to pressures to market these parts of an information services quickly. Browser's usability was neglected, gateways had security and scaling problems and the standard WML was quickly incremented to 1.2 and then revised to 1.2.1 to correct initial bugs. As a markup language WML was a novel approach to promote transaction based navigation, however it did suffer from the same accessibility problems such as verbose text as did its document based HTML competitor.

When evaluating WML it is difficult not to get entangled in the complex economic, technical and political issues of the wireless landscape. That shows that WML was so constrained, that arguably more important elements of this special markup language such as a study into how well its card and deck semantics or its binary XML features worked, were never reached. This indicates WML never matured into a competent markup language for information. WML as a specification of the WAP protocol stack did play its own part in its demise, as it isolated itself from the Web, content authors, desktop devices and users expectations.

2.3 SMS

The Short Message Service (SMS) application of mobiles deserves to be mentioned, as it proves that the mobile can be used as a platform for a text driven information application. SMS popularity can be attributed to several factors, including:

- Part of the initial GSM standard
- Simple interface
- Size limit of 160 "western" characters
- Pricing model

The GSM standard is digital and included a provision for relaying short data packets. The application for the transmission of small texts through the GSM standard is SMS. Practically every GSM compatible mobile phone is able to send and receive SMS text messages, therefore SMS had a distinct advantage over WAP in terms of deployment.

The simple effective interface in the bounds of the mobile, allowed a user to read and write messages easily and quickly. The SMS application is very well integrated with the device, as the interface provided direct keys to read and write the messages. Other usability enhancements included predictive text input, to speed up text input on a numeric keypad. SMS is hence a usable accessible mobile information application. The fixed size of 160 characters ensured messages were short and hence fast to read. The solid limit forced users to carefully compose their text messages in a efficient manner. The success of SMS affirms compactly written text as an essential property in high information transfer.

The SMS pricing model unlike WAP is a fixed charge per message sent. This charge was not time dependent and clear, whilst a pay per-use WAP subscription was typically based on time, which was difficult to determine by a user. If SMS used WAP, you would be charged for time it takes to pick up your messages, as well as the time it takes you to write and send them. This detracts accessibility, as some people take longer to type out messages than others [CCVW04]. Since SMS was unlike the Internet's email where it costs almost the same to send 1000 messages as it does to send one message, the service has been fortunately largely free of spam. However mobile operators have used SMS ability to push messages to the user's SMS application in order to advertise services, especially while a user is roaming under a foreign operator. The push application are considered important commercial applications for broadcasting users content. Pushed content could be messages, applications or services such as weather forecasts, news or stock alerts and reminders. As neither CHTML or early common WAP implementations have native push application support, specially formed SMS messages have been used to deliver profitable push services such as ringtones and logos. As mobiles can be carried with users wherever they go, the mobile is a profound direct target for marketeers and hence the scope for abuse is significant. If and how push messaging is implemented on mobiles in the future will be an important issue for the mobile as an information appliance and how it will distinguish itself from the W3C's Web where there are no plans to standardise such a mechanism.

SMS has not succeeded everywhere as in the USA market the size limit of 160 characters is not a standard. Pricing is not as simple, as users can get charged for receiving a text message. Additionally SMS transport is highly unreliable between competing providers. For these reasons SMS has not succeeded in the USA market, as it has done in the European markets.

SMS is so successful in some European markets that it is now the dominant application on users' mobiles. Although SMS is primarily a communicative application rather than an informative one, SMS can be implemented within the Web framework. Hugely popular Web services such as Microsoft's Hotmail offer email within the Web architecture. And similarly Docomo's I-mode and other Japanese services use email instead of SMS. Email will arguably replace SMS application in the future. However in the short term size limits could be a problem and the added overhead of the subject field. Incidentally the SMS size limit of 140 octets which translates to 160 7-bit characters will not survive the variable-length encodings of the international character set of Unicode. It will be a key milestone when SMS loses its profound dominant position on the mobiles platform to email. As email has been encapsulated by Webmail on desktops, SMS should also follow the trend by also converging to a messaging application within the Web architecture. Another key milestone for the Web applications that run on a mobile UA is when it competes with the native SMS application.

In conclusion SMS demonstrates that the mobile's physical dimensions are not preventing it from becoming an information appliance [Nie00]. This simple pervasive standard with an easy to understand pricing model by operators, integrated and polished SMS user agent by vendors are key factors of a text driven mobile application.

2.4 CHTML

> The greatest achievement of ”i-mode” service is its content development. In this context, our choice of adopting an HTML subset as the Markup Language, was the key to maximising the variation and quality of contents.
>
> Keiichi Enoki, Member of the Board and Managing Director, Gateway Business Department, NTT DoCoMo, Inc.

Compact HTML (CHTML) is the markup language used by the popular I-mode service in Japan by the NTT Docomo mobile telecommunications company. I-mode is not part of the WAP architecture and the implementation details are not available, except for a document submitted to the W3C detailing their compact markup language. The design goals of CHTML were [Kam98]:

- Completely based on (then) current HTML W3C recommendations
- Create a lightweight specification
- Can be viewed by a black and white display
- Can be easily operated by users

As a subset of the HTML markup language, CHTML content developers had a vast amount of existing HTML resources readily available in 1998 to draw upon. The specification was easy to understand, as it is almost half the size of WML's as it did not have the notorious deck of cards metaphor. Docomo's CHTML based service reduced the time to market as developers did not have to learn a new markup language.

Meeting CHTML's light specification goals entailed dropping elements that are now conveniently considered accessibility issues [CVJ01]. Most forms of embedded styling such as frames, image maps and tables. Therefore by correctly cutting down on the then heavy HTML specification, designers not only made it easier for the mobile device to render pages, but also made the pages more accessible for the end user. Practical implementations and experiments showed that Compact HTML was useful for small screen of 5-10 text lines and 10-20 characters wide. The specification included recommended limits on input buffer sizes that helped relieve authors from worrying about individual implementations of mobiles.

Notice that the Japanese language has far more complex language representations than European languages. That makes it technically more difficult to input characters on the mobile. The mobile platform in Japan has overcome these challenges to become a viable information appliance.

A compatibility benefit of CHTML, is that CHTML can be rendered by a typical desktop browser, making it easier to develop for and allows end users to swap onto a desktop machine and retrieve the same information. In practise desktop users could not view CHTML I-mode content as I-mode operated a wall garden model where it was free for content authors to add their content to I-mode's servers, with users requiring a subscription. Theoretically CHTML offered the possibility for multimodal access between the mobile and desktop. If you are reading a CHTML document on the desktop, but then you need to leave, you could finish reading the same document at the same address on the mobile device and vice versa. With WAP you would need two different versions of the document which bears a switching cost for information to transfer between and for users to navigate upon. CHTML as a subset of Web standards achieved device independent accessibility of information between devices.

Attributing the success of the hugely popular I-mode Internet information service to CHTML alone would be incorrect. The I-mode service charged per byte of a 9600bps packet switched data link and incorporated a convenient aggregation of small transactions of content providers' services into one monthly bill. These are two important factors they were missing from WAP services available at the same time that influenced user adoption of the service.

I-mode's service shows that an implementation of an accessible subset of the Internet standard HTML was a successful strategy. WAP's strategy against HTML's navigation model was defeated, as I-mode's success is now emulated by current WAP Forum standards which explained explained in detail in the section 3 about current trends.

2.5 W3C's Web

The Web boomed in about 1996 which derailed the Web from its W3C standards tracks. The standards of this time if implemented were plagued by proprietary vendor extensions. These presentational extensions and enhancements introduced by browser and plugin developers were aimed to meet the demands of content authors and to compete for a share of the new Web marketplace. As a result the W3C in an effort to reach consensus with implementations by the market leaders, Netscape and Microsoft, recommended HTML 3.2.

HTML 3.2 as a standard inadvertently introduced a trend in Web design that is still prevalent today, with poor use of presentational markup. The FONT and TABLE tags in particular allowed graphic designers to create visually appealing Web sites. Web pages were now based on presentational instead of structural principles for marking up information. For example many content authors used an elaborate physical font face with a certain size instead of logically using the first heading tag H1 for the title of the page. Now a machine is unable to determine if the text marked up in a FONT tag is of any semantic importance. The table tag for tabulated information was abused by authors to accomplish complex nested layouts such as multi-columned magazine pages. Tables are computationally expensive to render and they broke text flow by having to grow and shrink the table if the text size changed. Using font and table tags made browsers unable to properly resize text or the text flow when scaling, which is needed when displaying content on a small screen. Hence the then 3.2 markup standard of HTML featured elements that were difficult for mobiles to access.

The W3C standardised the frame element in HTML4 in 1998 that reduced scalability and broke Web navigation down from a single to a sequence of navigation actions [Nie96]. Frames were used to keep users locked within an application, making it difficult for them to leave. Bookmarking a page within a frameset often would not work as a user expects. Besides the usability problems, it made Web design more complex and comparable to WAP's WML card and deck metaphor. It was complicated for browsers to render as each frame within a frameset was a separate rendered page. HTML frames were unfortunately widely used as a means to help user navigation. This shows HTML lacked satisfactory means of overview navigation. This prompted WAP Forum's decision to adopt HDML for solving these navigation related problems. The W3C's bloated 3.2 specification made it a poor candidate for WAP's markup language. Future W3C specifications need to shed problematic elements, in order to be light weight enough to be implementable on a mobile.

Bitmap graphics were introduced to the Web in November 1993 with the Mosaic browser. Graphics are now used extensively in Web sites for not only showing pictures, but also abused commonly for controlling visual layout with contentless blank and spacer images. It is good Web practise to put alternative text to describe the images shown, except this practise is not widespread. Identifying presentational images from the semantics of the informational content they stylise is a problem. The main problem with all bitmap images is that they do not scale. A technology that fails to scale is very difficult to display on high resolution screens, as well as it is the low resolution screens characteristic to that of a mobile. There are workarounds to increasing image accessibility, but it demands computationally expensive techniques. Techniques such as image resizing, converting colours and cropping to an interesting feature have high processing and memory requirements. These techniques unfortunately still often lead to unsatisfactory results on the mobile with degraded images and slow rendering. The bitmap graphic formats of the Web GIF, JPEG and PNG all present problems for mobiles to access.

Besides W3C standards other proprietary Web technologies have established themselves on the Web, such as Adobe's Portable Document Format (PDF), Macromedia's Flash and Real's Realplayer. Many authors make extensive use of these client side technologies that made sites difficult to access from other platforms that do not have these proprietary clients for these formats. These clients exploited the abundant resources of the desktop and hence inappropriate for a limited mobile of this era. Other organisations require animation and interactivity and are unwilling to wait for W3C technologies to catch up with what is already commercially available. Multimedia like bitmap images all pose serious accessibility issues to the mobile, especially multimedia often needs further output devices such as a sound device or 3D graphics. The sound device on most mobiles is not easily audible from a hand-held position. Wide use of non-standards in Web content is extremely challenging as standards need to not only catch up with them, but also replace them. The W3C needs time to create them and authors need time to migrate to them, hence leaving mobiles inaccessible to disruptive, but exciting Web technologies.

The W3C sanctioned Standard Generalised Markup Language (SGML) based HTML of this era allowed malformed markup and as a result the Web is malformed too. SGML's grammar allowed malformed markup that would be incorrect in XML such as a missing end tag. SGML's flexibility was balanced with a large specification and rare implementations of it. SGML parsers were often incomplete and as a result browsers employed even more liberal, unpredictable and nonstandard "tag soup" parsers. At this time in order to stay compatible with leading browsers, competing browsers had to emulate the unpredictable way that the most popular browser would deal with poor markup. Therefore writing a browser for the limited first generation mobile and maintaining user expectations by emulating quirky behaviour of browsers of the desktop was very difficult. "Tag soup" is tolerant to authors' mistakes, but hard on the computer. At the time the mobile utilised the XML markup language that is harder on people, but easier on the computer. However a new markup language for the Web will not fix all the user errors of SGML's legacy on the Web. Browser parsers must be liberal as users expect pages to render. Denying access to a badly formed HTML is a strategy that would not work, as you would deny most of the Web from the user.

HTML aside, the way the Web worked was simple and fault tolerant at the same time due to its architecture represented in figure 4. Normal access did not require a proxy or gateway, therefore Web sites often could utilise simple IP based authentication to control access to information. Nor did it require the overhead, the likely bottleneck and possibly restrictive WAP gateway reprocessing the application layer content.

In order to combat presentational devices polluting content, the W3C recommended Cascading Style Sheets (CSS) [LB96]. Although recommended in 1996, it was not implemented before HTML 3.2 made its impact. As other widespread Web technologies were able to do impressive visuals, in comparison CSS when implemented years later in roughly the year 2000, it could only control fonts and borders in a style sheet. This is what the widely adopted font and table elements already HTML 3.2 did by this time and CSS implementations were often incomplete or incompatible in leading browsers making it difficult for authors to later switch over to. Through the style sheet, content and presentation could be separated which would greatly aid mobiles ability to access Web content. If the demands of the style were too high, the mobile could theoretically not use it and use the vanilla but accessible version. A style sheet could be associated with any number of pages, therefore bandwidth savings were made that would have greatly benefited the mobile. Considering when

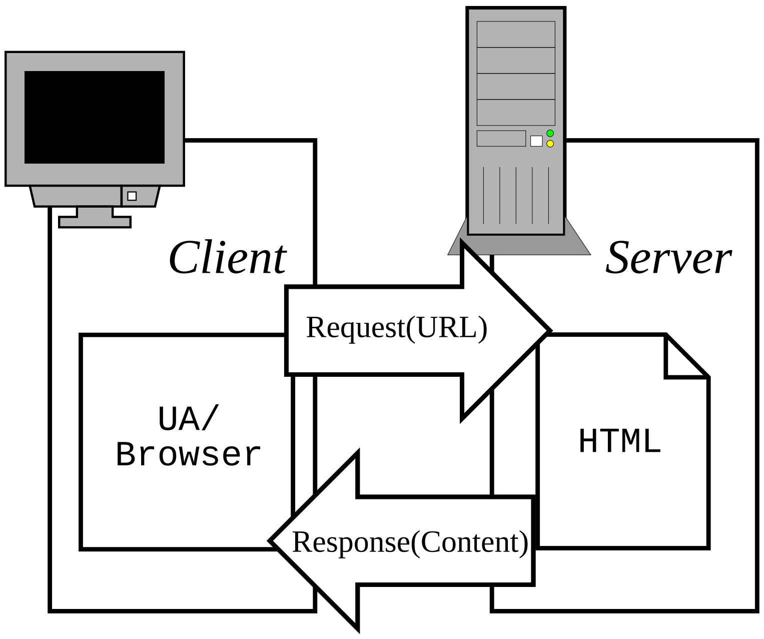


Figure 4: WWW architecture

it became a recommendation, it was unfortunate it was neither implemented in desktop or mobile devices until years later. Although rendering stylised text on a small mobile screen is still a problem, CSS presents an option not to have any complicated style in order to let the browser decide how to best display the character based content for the device. Separating content and style is crucial for accessibility required by that of mobiles and future use of such practises will indicate the Web's appropriateness as an universal information space.

The Web is an emerging media that still needs to reach its full potential. The radical growth it experienced in the past few years has created several accessibility problems. The popularity of troublesome presentational elements embedded in HTML is being separated with CSS, however non-text multimedia and common proprietary Web technologies have to be also addressed by the W3C. There is also the larger problem of the legacy of bad markup of the Web, it too should be accessible to mobiles. Despite the faltering steps of this immature medium, the Web has made an impact on our society and is here to stay as the Internet's dominant information space for the near future.

3 Second generation of the mobile Web

The IT bubble has burst, Web innovation has slowed and mobile devices have advanced. Knowing the background leading up to today, we can now evaluate how strategies, methodologies and technologies have improved to address the problems both the mobile and desktop platforms raised that prevented a shared and accessible information space.

A close look of how the W3C and the WAP Forum, now known as the Open Mobile Alliance, are converging their standardisation efforts and learning from their previous mistakes. The W3C mandated migration from SGML to XML and its family of Web technologies warrants a special focus. As the Web medium is an established part of computer science, there is now an opportunity to introduce research to cast another viewpoint to the problem of improving accessibility of information on the Web.

3.1 Environment

The World Wide Consortium Device Independence Group has specified a framework called Composite Capability/Preference Profiles [KRW$^+$04] to help describe User Agents (UA) for content adaption. In the past year this has resulted in the availability of quantitative data in the form of UA profiles from mobile manufacturers with device software and hardware characteristics. Unfortunately temporal elements such as the time the device was issued are not included in the schema. One could use the announcement date of the product by trawling manufacturer's websites, but that value is subjective. The device may never have been shipped according to that date and it might have had a very limited production run and market influence. Besides UA profiles are a relatively new W3C recommendation, hence the profiles are only approximately one year old and therefore too short a time for identifying mobile hardware trends.

Different manufacturers tend to use different schemas or vocabulary to describe their devices. If they do use the same schema, some might tell about how many characters can be rendered on the screen, whilst others tell only about the screen size in pixels. In addition the screen size stipulated in the schema is no indicator of the addressable maximum screen resolution of the browser user agent of the device. Therefore in practical implementations the screen size offered by the manufacturer's device profile has little value.

Quantitative analysis of the device matrix offered by market leaders such as Nokia has further problems due to market fragmentation. The Nokia range of mobile devices unnecessarily differentiate integral hardware and software features from one another. Hence it is difficult to identify hardware trends of in-built cameras, Bluetooth or Enhanced Data Rates for GSM Evolution (EDGE) functionality. Mobile models for example can differ between those in the Americas compared to Europe. Mobiles can differ on demographic grounds, for people who seek high fashion where devices are elegant, for business users, or sports enthusiasts who want a more robust model. Individual mobile life cycles differ as older Nokia communicators with screen sizes of 640x200 are still supported, which reflects poorly on the overall trend of screen sizes. Some models may be in abundance compared to others, however obtaining sales volume figures and determining their worldwide impact is again difficult. For these reasons statistical quantitative analysis of mobile hardware characteristics to identify trends can not be seriously conducted. Therefore we will have to continue with qualitative observations of the current mobile environment relevant to that of accessing the Web.

Mobile software can be still considered as not being upgradable. There is little incentive for mobile vendors to port drivers dependent on a specific mobile model to newer versions of a mobile operating system. Demanding user agents are dependent on operating system enhancements to implement more functionality. Porting drivers of closed specified hardware to the new operating system is an expensive in house operation, although it can greatly improve older mobiles functionality and lifetime of a mobile. There is little return on this investment for the mobile vendor, as the user periodically replacing the entire physical mobile for an upgrade is typically part of the vendor's business model. Hence mobile software still doesn't give users a software upgrade path comparable to that of the desktop computing environment.

Mobile platform software is typically particular to each mobile manufacturer. In Nokia's range, there is a development platform grouping between the series 40 and 60 software platform. Both platforms allow software applications to be installed in a Java environment. This environment although an improvement over the earlier generation of mobiles is not enough for a full featured user agent. Series 60 is the high end "smartphone" platform that provides an environment where 3rd party user agents such as the Opera browser [Ope04a] can now be installed and operated. The more common series 40 however only ships with Nokia's in house browser. Evaluating this user agent is difficult because the User Agent profile does not indicate versions of the browser. The user agent string of the browser shows the model of the phone, followed by the version of the development platform. Next in brackets is the version of the operating system. This version is the best indicator of the version of the user agent, as there is no link between the development platform and the user agent version. Changes in the version do not necessarily mean a change in the browser user agent however, as some other part of the system unrelated to browsing may have changed. Neither the user agent string or profile provide Web engineers a concrete indicator to the mobile's user agent. As the Web user agent is the most important part of the mobile platform, it is vital to correctly identify the user agent in order to track it Web standards compliance. Web developers otherwise face a difficult task of working around inevitable user agent software bugs, without knowing which user agent they affect.

```
Nokia3100/1.0 (04.01) Profile/MIDP-1.0 Configuration/CLDC-1.0
```

From October 2004 Nokia has corrected this UA string to a Web compliant User Agent header [Nok04].

The user agent string also indicates Java class functionality of the phone, however it is not directly relevant to the browser. Java classes are helpful for identifying the broad capabilities of the mobile with less complexity than user agent profiles. The classes are defined [GSM03] to propose that screen characteristics must be at least

120x120, minimum memory amounts of 256KB and the colour depth must be at least 8 bit.

Many mobile phones today implement at least 4096 (12 bit) colour screen displays. Colours can improve accessibility of information by helping people recognise fields that need attention [CCVW04]. However they can also detract accessibility if the colours are chosen poorly, for example when two colours do not have enough contrast. For people who suffer from colour blindness, certain colours might be indiscernible and any information which is dependent on colour will be less accessible. Therefore marked up information should be independent of colour and there should be an option provided by the user agent to turn off the colour styling. Another possible negative affect of colour screens is that they seem sluggish and unresponsive compared to older black and white screens which detracts the usability of the device.

Fonts and internationalisation problems have improved. Better storage capabilities on mobiles have resulted in more Unicode glyphs, therefore on today's mobile you can read Russian Web pages in Cyrillic. However mobile interfaces are still often limited to the languages available on the phone in the country it was purchased in. Therefore now you can probably read Web pages written in exotic languages such as Sanskrit or Arabic on a modern mobile, but the user agent has no functionality to allow the user to write it. There are typically three different text sizes, small, medium and large available on mobiles. This allows a little more logical structure with headers, however this is arguably of little use as a deeply hierarchical document on a mobile screen is not a typical use case. Small text sizes do allow a lot more information to be crammed into a small screen size. In the scenario of using the mobile whilst walking, text may be too small to read. Therefore since different text sizes have been introduced, the mobile should allow the user to resize the text to a larger one. This feature of either zooming or scaling the text is often unimplemented by mobile user agents, possibly causing further accessibility problems for the aged for example. Web developers should also be aware that if they logically markup information with text size alone, then it will lost meaning when a user wants the font size as large as possible. The distinguishing element of text formatting lies with emphasising the text by **bolding** (weighting) or *italics*. Unfortunately different levels of weighting and italics are often poorly implemented and hence colours with their aforementioned issues are commonly used instead.

Besides the input issues for internationalisation, input design and functionality forms a considerable contrast to the desktop's full size keyboard and mouse. There are no rigid standards for inputs, however there are industry core recommendations for a dedicated select/accept and back/erase key. This does not translate predictably to Web navigation paradigms of following a link and returning from one, as for example the back key terminates a network connection and returns the mobile to an idle state on Nokia phones. Enhanced user interfaces are recommended by the industry to have two softkeys which change their meaning upon different contexts. These keys are commonly used for assisting navigation via a menu. There are several other physical input considerations such as the size of the keys and if and how a joystick (4-way control) or scroller(2-way control) is implemented. Unfortunately keypad design is not settling as it has become a way of differentiating a mobile product from a competitors'.

However as browsers are chosen, deployed and tested by the mobile vendors themselves, it is their responsibility to ensure the user agent is usable with the physical interface they designed. The software drivers for the inputs are important, as well as text input software providing predictive and adaptive capabilities. These software enhancements can make text input faster and easier than without them and this functionality is used to great advantage in the SMS application. Incidentally predictive writing from a browser input field is a core requirement of the GSM Association [GSM03]. Another trend in inputs is that mobiles can be connected to external inputs such as a full size keyboard or a mouse with Bluetooth. Future improvised and wearable inputs could drastically change the notion of the typical mobile device. Today the mobile's inputs defines itself in the range of small computing devices, however this is set to change. Although limited and often inconsistent the inputs alone should not obstruct Web developers attempts to bring the Web to the mobile. Inputs are in the domain of the user agent and device manufacturer to cooperate on and not the Web developer.

Wireless network connectivity is still expensive and error prone. There may be be more bandwidth now, but it can easily fluctuate to previous low levels. Today the major leap forward for the mobile as information appliance is the GSM supplement General Packet Radio Service (GPRS). It solves two problems that plagued the earlier incarnations of the WAP information appliance. The mobile does not need to dial-in over a circuit switched network, as it always connected for a packet switched network. This results in faster connection times, making the device much more responsive and usable. Due to limited cell capacity and design, GPRS speed gains over the previous circuit switched system are not significant. The other problem it solves is the cost of accessing the Internet from a mobile. Since GPRS is always connected, operators can offer tariff plans such as charging for the amount of data transferred or to a simpler flat monthly fee. The user with GPRS now does not face the stress of per second billing whilst using the phone, or face the expense of redialling after a lost connection. However per byte billing can be just as stressful when an UA fails to provide how much bandwidth consumed and converting that into clear costs for a user. Although out of the scope of this thesis, network provision and billing play key roles in uniting the Web with the mobile.

The problem addressed in the earlier generation of information enabled mobiles was the walled garden approach of operators. The walled garden is increasingly referred to as a vertical market. The walled garden still exists largely because it is difficult to bill users for services. For application vendors it is easier to sell services to mobile operators for them to attract customers to services unique to their operation. This model works in reverse with operators restricting users to particular applications in order to generate revenue from the application vendors. For example although a user wants to use the search functionality of X company, the user is redirected by the mobile operator to Y company. Y company pays the mobile operator for the generated traffic. Often the application is not even redirected, resulting in a frustrating 404 error for customers. Therefore though mobile users have a HTML browser, there is no guarantee that they will be able to access the same sites as a desktop Web user. Operators still threaten to fragment information space, without the help of specifications.

The problem that plagued earlier generation of information enabled mobiles was the failure to stipulate a maximum file size has not gone away. Content developers still need to check the browser characteristics with difficulty from each mobile device to determine the maximum size of the content. With dynamically generated HTML, CSS, bitmap images and compression it is a difficult problem to accurately determine the total payload size. If the size is exceeded it is also difficult to form a policy of adjusting the content appropriately. At this stage mobile manufacturer user agents should have created an environment whereby authors can create and do not have to worry about these frustrating variable limits.

Error recovery and feedback on mobiles is still awful. There is no specification defining the user agent behaviour regarding display priority [GSM03]. On the Nokia 3100 mobile device a user's error notification is a simple notification message which is flashed for a couple of seconds. Unfortunately the actual message is often unhelpful for debugging. Errors whilst rendering a page in a browser often still silently fail or worse fail to show perhaps an important component of the document. This problem also affects desktop browsers that are not standards compliant. An important aspect of error handling is providing feedback to the responsible party. For example browsing with the Nokia series 40 XHTML browser on a secure HTTP site gives a "No gateway reply" error message. The user doesn't know if they should contact the mobile vendor, their network operator or the content author about resolving the problem. Once again, if they knew who was responsible there is no standard mechanism to message or call in the problem report from a mobile UA. This lack of core interactivity and responsibility on the mobile platform misrepresents the large user base behind mobiles and it must be addressed to ensure the mobile as an interactive information appliance.

Plotting CPU power and memory capacity trends of mobiles today is difficult as their specifications are not well publicised and difficult to compare. They have improved roughly in accordance with Moore's law, hence both CPU and memory are becoming non-issues. Moore's law does not apply to radio wireless or battery power which is still the weak link, especially as CPU intensive applications drain more battery power. Software power management has become more efficient whilst the Lithium cell technology powering today's generation of second generation mobiles has not significantly changed from the earlier generation. The GSM association interestingly mandates a one day minimum lifetime from device manufacturers with three days strongly recommended [GSM03].

The desktop environment on the other side of the spectrum has also evolved. Power supply demands of desktop incidentally increased in the past years. High powered high frequency CPUs also need better cooling in the form of high powered fans, hence desktop machines of late have been known as noisy power consumers. There are now even larger screens, more memory and increased CPU speeds. User agent software with desktops is dominated by Microsoft's Internet Explorer browser [The04]. Web compliance updates to this browser are completely dependent on the Microsoft corporation. The consequences of Internet Explorer's impact on the Web are dealt with in section 3.3.

Comparing the environment of the information appliance in earlier and current generations of mobiles shows us that the mobile device is rapidly evolving. Numerous capabilities have broadly improved such as connectivity, memory size and CPU. This provides a more capable platform to service the high demands of a Web User Agent implementation. The increase in computing power has allowed mobiles to replace the address book and calendar functionality of small computing devices such as PDAs. Mobiles have also now incorporated cameras, possibly replacing the snapshot camera device. Mobiles can provide a variety of tasks from telling time to changing TV channels. This small device consolidation indicates a trend towards a more generic and versatile computing device.

The two persistent elements that are said to hinder a mobile's progress as a Web appliance platform is its typical poor keypad input and screen size. Input methods are already enough to allow text messaging to become the leading application on a mobile phone. The screen size limitation to an extent is also a fallacy. Recent research shows that a human's eye span only stretches a few characters wide [BDDG03]. Therefore if the user agent can scroll text as smoothly as reading a printed magazine column then this problem could be solved. However in this generation screens are typically slow scrolling and a far lower DPI than printed media and therefore a hindrance.

For Web user agent developers the mobile platform needs to become more generic, more powerful and easier to deploy updates. For Web engineers, it is the user agent compliance to standards that matter. For Web users its the cost of connectivity and the experience of the user agent. The software user agent is the key element in this environment.

3.2 OMA

As early as 1998 the WAP Forum approached the W3 Consortium [HMK98] to cooperate in addressing the problem area of mobile access of Web content. In 2001 a proposal for WAP2.0 by the WAP Forum was announced which confirms WAP Forum's change of direction in adopting W3C standards of markup and discontinuing WML 1.x.

Shortly after releasing the white paper on WAP2.0, the WAP Forum was consolidated into the Open Mobile Alliance (OMA). WAP Forum was joined by the SyncML initiative, Location Interoperability Forum (LIF), MMS Interoperability Group (MMS-IOP), Wireless Village, Mobile Gaming Interoperability Forum (MGIF), and Mobile Wireless Internet Forum (MFIW). OMA's attention has now expanded to interoperability of calender, contact, instant messaging, multimedia, gaming and locational information. The OMA has set upon itself setting up several new interfaces of information, although notice that the Web architecture could be applied to all these application areas mentioned.

The WAP Forum or OMA decided to replace WML with W3C's Extensible HyperText Markup Language(XHTML) which is the reformulation of HTML4 as an application of XML. XHTML is modularised into a collection of abstract modules that provide specific types of functionality [ABD$^+$01]. Modularised XHTML elements can be combined and extended which allows the language to adapt to different contexts and contents. Using figure 7 as a guide, we can see how modularised WML 1.x combined with XHTML Basic to create XHTML Mobile Profile (XHTML MP). Unfortunately parts of WML1.x could not be expressed in an XHTML module. A markup language called WML2.0 which features XHTML MP and modularised WML extensions via a XML namespace was drafted to ensure backwards compatibility as seen in figure 5. For pragmatic mobile content developers it was important to have WML functionality in order to assist navigation and ensure usability. In practise the WML namespace became optional, after WAP Forum decided that the WML extensions were just for backwards compatibility and were not going to be extended. There are two separate markup languages in WML2.0, XHTML MP and WML 1.x. as seen in figure 6.

The Nokia user agent implemented WML2.0 so that they seamlessly supported XHTML and WML by switching modes when it encountered the content type identified markup. Therefore there was no option to embed WML in XHTML which might have avoided the problem of mobile specific markup in HTML, which could have made WML2.0 unavailable by desktop browsers. Nokia has isolated WML and began implementing a Web user agent for their mobiles.

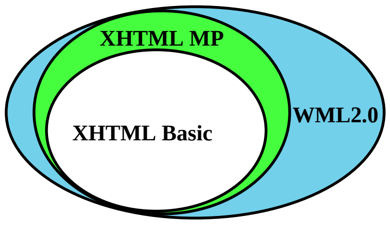


Figure 5: XHTML Browser with XML namespace

Moving from WML 1.x to XHTML MP is more painful than just an XSLT transform on the XML [Ope01]. For developers it meant losing the functionality of their expensively engineered deck application consisting of individual task cards. Leaving transaction based navigation put stress on the mobile browser to ensure good browsing navigation. XHTML MP user agent implementations did not compensate WML for the deck loss. For example the user agent should show the URL of the Web page. Without this basic navigation aid it is difficult to know where you are in the Web. The new user agents could have paginated long XHTML documents

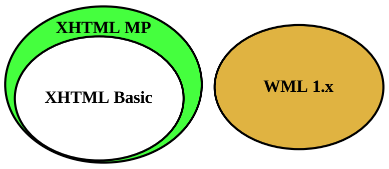


Figure 6: XHTML Browser with separate WML browser

to make to more usable like a deck, but failed to. This means that the server often has to paginate the page and therefore each page is a long request reply round trip. Using XHTML instead of WML would mean a loss of usability that assisted earlier in navigation and caching of pages.

XHTML MP does frustratingly mix and match W3C specifications, by introducing some elements from XHTML 1.1. Despite XHTML MP's confusing design and not being a W3C endorsed Web standard, by and large XHTML MP can be accessed by a typical desktop Web browser. This is accomplished by the author who simply adjusts the MIME type of the served document from the mobile specific application/vnd.wap.xhtml+xml to the universal text/html type. This marks a convergence milestone between the the mobile and the Web. Unfortunately as dual browsers silently process XHTML and WML as if they were no different, users do not know they are using the Web API's HTML interface as there is no indication by the user agent. A user does not know they have the profound possibility to access the same information from a desktop PC. Convergence has also been marred by poor marketing as many users still think they are using WAP, which they have now accepted as not being the Web. Authors face a similar dilemma as it not publicised that a mobile UA handles XHTML MP no differently than the ordinary HTML of the Web.

Comparing XHTML MP with Docomo's CHTML from section 2.4 is worthwhile. OMA has followed a similar strategy and now both parties converge too as seen in figure 7. This unites Japanese and European mobile information markets, with a compatible Web markup specification. The difference lies in the user agent's implementation quality of the HTML specification. European mobile manufacturers' software such as Nokia's user agent has to play catch up to Docomo's I-mode lead in user agent technology and experience.

WAP2.0 features wireless profile CSS (WCSS) [WAP01], which is an important part of Web architecture and engineering to separate style and content [Jac03]. This second generation of mobiles have grown their graphical capability and CSS is the appropriate Web standard complement to allow authors to style their content. WCSS is based on a subset of W3C's CSS2 [LBLJ98], however with additional elements such as:

Marquee For the infamous scrolling text effect (popular in Chinese markets)

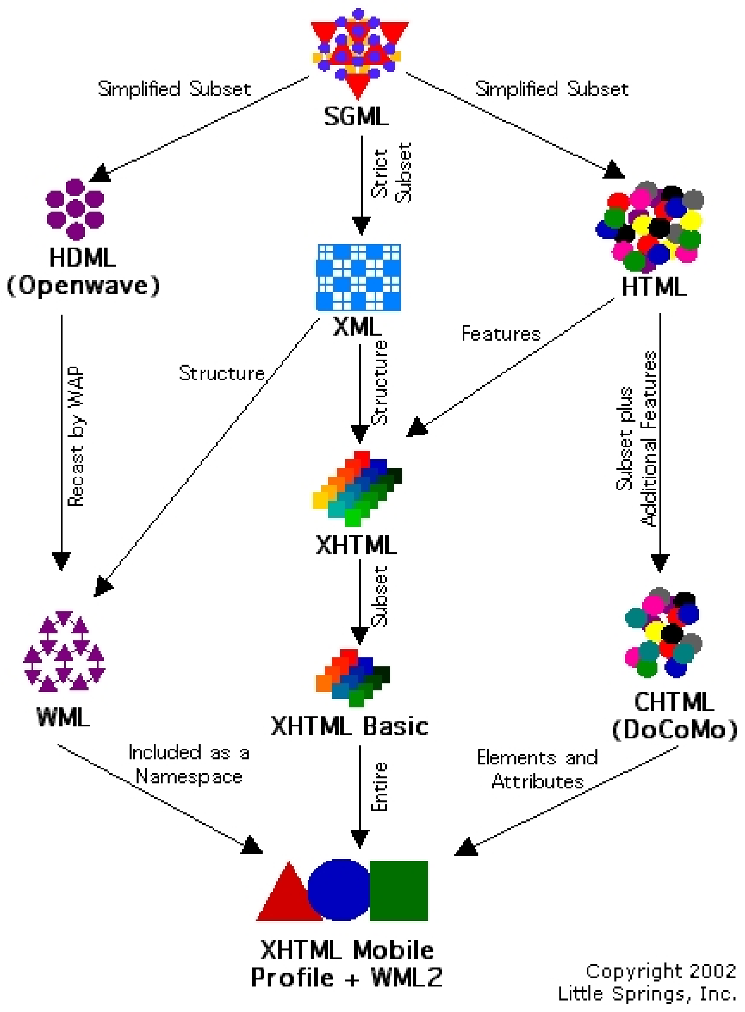


Figure 7: Markup History [Lit02]

Input Specifies additional input controls

Accesskey Which is already in XHTML Basic's Hypertext module

Marquee usage and incidentally the blink effect although implemented by desktop browsers too are not encouraged largely due to their accessibility issues. These styles are time dependent and hence someone who reads faster or slower will be inconvenienced by such an effect. These animations are eye catching and popular in use for entertainment purposes, hence their inclusion into this specification. Input controls are sorely missing from HTML specifications. To some degree this void has been filled with ECMA script, but the mobile equivalent of WMLScript is difficult to work with, due to substandard conformance and mixed implementations. These input controls are especially important to reduce the costly server requests and client replies, hence the implementation in WCSS. For the desktop paradigm of

Web access there are mature uniform deployments of ECMA script. Therefore a desktop user agent would conceivably ignore this WCSS input extension, whilst a mobile user agent would ignore ECMA script. For now this inconsistency does not affect viewing of information, but for Web application interaction it does. A Web application developer now needs to add redundant code to ensure it works on mobiles and desktops which either increases the payload for mobiles or forces the developer into device dependent engineering.

The accesskey was proposed as an another optional method to activate an element with CSS. The WAP CSS extension accesskey allows a key or key combination to be bound to a particular document element in order to activate that element. WCSS explicitly names keys such as the mail access key of a mobile that can be rebound by the user agent's rendered document. This feature is open to abuse allowing an application to bind every key and lock the user to the page. Furthermore the list of supported keys or characters used by accesskey is device dependent, therefore inappropriate for device independent engineering. Accesskey implementations on the desktop browser can also conflict with existing shortcuts, however the user agent's own bindings takes priority, which is different to what the OMA specified for mobiles. For mobile users it is not intuitive to have a key such as the answer call button bound to an element by the author on the Web page. The user agent should be able to automatically give appropriate shortcuts to interesting elements or components in a document. This is already accomplished to a degree with softkeys. Giving authors this ability to override shortcut conventions degrades usability and is best avoided. For users it is difficult to tell which accesskeys are bound. Both Marquee effects and the non-Web standard input controls results in mobile specific code, however user agents can be easily programmed to ignore them. Although OMA enhanced the specification for the mobile arena, WCSS can be used on any device user agent complying to the CSS standard [LB96] without any major interoperability issues.

The underlying WAP2.0 stack can inconsistently alternate between WAP1.0's stack and WAP2.0's IP based stack much in the same way as the dual mode browser. It is not apparent what stack you are using as they can be both implemented. This might be a concern to setup a direct connection between the user agent and server for security and integrity reasons of a Web service application. Connection quality is also dependent on if and how an operator has established a gateway. OMA's decision to adopt Internet standards with the legacy of WAP attached it leaves Web developers with the same difficult problems as before. Since binary compression may not be used in the IP based stack, one can argue that the problem is now worse since you can not rely on any of the benefits of WAP delivery optimisations.

Relating back to section 2.3 OMA's messaging efforts are still separate from both Web browsing and email. Long or concatenated SMSes have enhanced SMSes as well as Multimedia Messaging Service (MMS) has come of age. OMA has interestingly mapped the 3G messaging solution MMS into the WAP specification, taking advantage of browsing technologies UA Profiles and WAP Push to create the service. UA profiles are addressed in detail in section 3.3. Conceivably MMS is a Web application, however operators are keen to make each MMS discrete in order to charge for them. The underlying push technology specifications are still SMS, with a URL to the multimedia payload. The legacy of SMS based services are easier for mobile network operators to control and charge for compared to email and hence email implementations have been ignored.

SMS is a vertical market push technology monopolised by the mobile operators that might be regulated. There are no clean horizontal interfaces between operators as they are protective of their own markets and investments. Interestingly push messages may be regulated under particular country rules due to their associated privacy concerns and risk of abuse. For example where marketeers are deemed to inevitably abuse the channel to deliver a targeted message, which is commonly known as spam.

Push messages are not part of the Web's request and reply architecture on the desktop platform. Although email is often used in conjunction with many Web applications, especially for authentication. Email can be encapsulated inside a Web application, however with the mobile characteristic of limited screen space for advertisements this could prove problematic for current Web mail business models. Access to an mail Web application will sidestep the operator's control of messaging. Therefore questions like 'Will operators open up messaging?' or 'Will mobiles implement email standards?' are not as important as an Web application that functions on the mobile UA. The major stumbling block for email and MMS carried within Web architecture is the quality and usability of the user agent. The MMS application on mobiles are unfortunately separate to the browser application, although they share the same underlying technology. Push technology within a Web application on the mobile platform remains a milestone to reach.

OMA's focus on Digital Rights Management (DRM) is another effort to ensure vertical markets which plays no part on the liberal Web. DRM is used to prevent consumers from sharing content, allowing operators to carefully control and charge for content. These DRM locks are not part of the Web architecture. The Web has its own ways of protecting content already. DRM does not threaten the nature of the Web, however the Web does threaten the use of DRM. Disallowing users to make use of a freely available ring tones, images or other Web resources from a mobile UA would be a move that would only frustrate users. When today's image and ring tone businesses disappear, it will be another curious milestone for universal access to information. DRM is largely avoided with the current nature of the Web.

OMA has changed its direction towards W3C Web standards, with WML effectively discontinued and replaced by their deceptive selection of W3C standards. Device independence would be better assured if OMA did not mandate a non Web standard superset of W3C's XHTML Basic and CSS technologies, but in practise the damage is minimal. Politics and marketing are a different matter as this important transition to Web standards has occurred under the unfortunate legacy of WAP. Dual mode browsers, the dual stack and restrictive operator policies will confuse and frustrate developers and users. Developers will find it difficult to be develop for substandard and premature implementations of Web user agents that are currently being shipped in mobile phones. The future is hampered by little or no opportunity to choose another or upgrade their mobile's user agent without replacing their entire mobile. However there exists exploitable shared support of the Web's basic markup and style between the mobile and desktop devices.

3.3 W3C

> All these technologies are supposed to be device-independent, so if they aren't suitable for mobile devices, the problem is with the technology itself, not with the lack of a profile.
>
> Ian Hickson

In this section we continue analyse and comment on the W3C efforts to standardise the Web for universal access.

HTML was put in maintenance in 1999 [RHJ99] and since then the W3C has thrown its weight behind XML and XHTML technologies. Now there is a growing family of XML specifications by the W3C such as Xforms for forms processing, XSLT for transformations, Xpath for queries, SVG for vector graphics, SMIL for multimedia, MathML for mathematic symbols, Xlink for describing components links and many more.

As described in the previous section, XHTML is simply the reformulation of HTML 4 as an application of XML. The earlier section 2.5 before raised the problems of SGML and its complementing computing intensive and quirky "tag soup" parsers. Information marked up in XHTML [BIM+00] makes information much easier to parse and manipulate for the user agent and hence very attractive for UAs with limited resources. Thus the UA requires less implementation at the expense of the author. The author now has to ensure the information is strictly in well formed XHTML. With XHTML UAs can easily catch well-formedness errors and allow XHTML to be mixed and matched with the other extensive and extensible XML related technologies mentioned. For these reasons the OMA chose to base its markup standard on W3C's XHTML.

Unfortunately there is a substantial difference between W3C's endorsement of XML and the Web's adoption it. The dominant desktop browser Internet Explorer 6 (IE6) does not support XHTML. Because XHTML is largely a stricter form of HTML, if the XHTML page is sent as the content type text/html IE will render it. As it is impossible to reliably automatically detect XHTML sent as text/html [Hic02], the UA must assume the content is HTML even though it might be XHTML. Hence the benefits of using XHTML are lost at the cost of ensuring well formedness. If you pretend that IE6 does not exist, authors need to change their development practises and learn new tools which as we have already seen in section 2.2, is costly. XHTML must be well formed to be advantageous, therefore authors can not simply test their content with leading UAs as their accustomed to. They have to use a validator. Most authors will find it difficult to comply to the strict rules governing XHTML

1.0 which has the features as the less demanding HTML4. Disadvantages of serving the XHTML content type include the loss of incremental (progressive) page loads by the UA and the popular search engines not indexing XHTML. The reality is that there is a insignificant amount of valid XHTML on the Web today in 2004. XHTML is not the markup of the Web, HTML still is.

Ignoring the state of the Web and concentrating on building a better and somewhat Web compatible world with XHTML is an option worth exploring. Most of the Web is poorly written markup and will not stand a chance of running on a second generation mobile in any case, one could argue. You could conceivably leverage the XHTML browser in mobiles and reap all the benefits of a clean lightweight UA implementation with extensibility. However after testing the Nokia UA in the popular series 40 line of mobiles to determine whether or not the parser is XML compliant or not, reveals interesting results. Despite the limited resources of a mobile, the XHTML browser does not implement a compliant XML parser and treats malformed content without complaint. Therefore there is absolutely no benefit of using XHTML in current mobile UA implementations either. Doing so may cause more harm than good as the author may think their producing valid XHTML, when they are not. The popular Nokia device and its preinstalled own Web UA follows the "tag soup" trend with desktop browsers, so one can safely predict that XHTML will not be the de facto standard in the mobile domain too. In conclusion the mobile UA is currently supporting HTML, not XHTML.

CSS is the key technology behind achieving separation of style and content [Jac03]. CSS needs to fulfil the high demands of Web designers, at the expense of user agent control. There is an inverse relationship between developer and user agent about the fine control of the information display output. The user agent is in the best position to choose the appropriate font or colour scheme especially on the limited mobile platform, however the designer may want to override these safer defaults. Ultimately the user agent should have the last say in rendering the style sheet or not by presenting the option to the user, as the style sheet may be inappropriate for the device. CSS offers a mechanism for mobile UAs to work around heavy embedded styling, ensuring access to content on any device. CSS saves on bandwidth by allowing a UA to cache the style sheet which is typically applied across a Web application. CSS allows the possibility of more fluid and scalable designs, that are more likely to work on a mobile platform with its limited screen. CSS works with either HTML, XHTML and other markup languages. CSS is vital to ensure information is device independent and accessible on mobiles.

Despite the importance of CSS, the W3C threatens to undermine CSS's universal goals with device dependent profiles. CSS Mobile profile [WDSR02] is a candidate recommendation specification marking out elements of the CSS specification [LBLJ98] that an UA does not need to implement. This additional specification creates confusion and encourages device or profile dependent UA implementations. The way CSS mobile profile relates to WAP Forum's WCSS [WAP01] mobile dependent CSS subset also threatens to confuse and frustrate. Web developers are needlessly given the task of mapping the differences between three specifications and judging what or what not to implement. W3C has decided to make a profile to declare the minimum CSS to what a mobile device should implement, which is arguably out of their domain. User agent developers should declare what they have implemented or what they have not of the specification. UA developers, the W3C implementors are experts in this field and not the W3C. Making profiles segregates UA developers and Web authors by allowing effectively device dependent implementations of W3C standards which are different again from mainstream desktop implementations.

W3C's unpractical standardisation efforts regarding Standard Vector Graphics (SVG) threatens the Web on the mobile [Ann04]. SVG also suffers from several confusing separate device dependent profiles, which are both subsets of the complex SVG specification. One named SVG tiny and the other SVG basic [Cap04]. SVG is incompatible with CSS and it does not have the flexibility of switching to alternate style sheets. It breaks the fundamental Web architecture of separating content and style, by making it very difficult to distinguish between the two. SVG can be compared to Flash multimedia, a self-contained format featuring content, graphics and animation. It is seen as a format to offer slick entertaining clips of information to mobile users. SVG does not scale like a text resize, instead it can be zoomed upon. That leaves users with the uncomfortable action of magnifying and panning a presentation if the text size is too small, which will be a problem on varying small sizes of mobile devices. SVG is unable to degrade unlike CSS. If currently widely deployed UAs which do not support SVG attempt to view SVG content, that content will not be rendered at all. On desktop devices SVG implementations have been heavy, very limited and not widespread. SVG and its mobile profiles is a Web incompatible specification.

W3C CSS media queries [LG02] at first inspection may be interpreted as encouraging device dependent engineering. With media queries a developer can make different style sheets for different device groups. The groups include screen for desktop devices, handheld for mobile devices, print for printers, tv for television and so forth. The groups are a little vague, especially with the environment of mobile devices quickly becoming as capable as the "screen" media type. Another argument against media types is that it devalues the notion of the Web medium. It does not treat the Web as a media of its own and only considers it in the view port of other mediums be it paper or television. CSS media queries serves the demands of designers who are not concerned about information for each device, while still allowing an UA to disable them. To its credit, the core requirements of device independent engineering are assured. Media queries helps authors to create just one document that can then be used on all devices, instead of authors needing to create one document per device, which is far worse.

The W3C working group most concerned with the mobile device, is the device independence work group [Wor01]. Previously this group originated from W3C's User Interface domain. The UI domain split into Document Formats and Interaction domain in July 2001. Document Formats was later was split again to Architecture and some parts moved in the Interaction domain. The interaction domain first featured the Mobile Access Activity which later was merged into the Device Independence

Activity. They have notably produced Composite Capability/Preference Profiles (CC/PP) [KRW+04] and Authoring Techniques for Device Independence [HM04].

UA Profiles (CC/PP) was criticised earlier in section 3.1 for being difficult to parse, having no temporal value and featuring alternating vocabulary to describe the device. However UA Profiles have been deployed successfully in commercial vertical markets for identifying device capabilities as used for the MMS service which allows mobile users to send and receive pictures. UA profiles outside the domain of these mobile vertical markets is very hard to find.

If a device updates or installs another UA it is unclear how the UA profile is supposed to reflect this. UA profiles only describe the UA deployed by the device's vendor. This limits consumer choice to the UA chosen by the vendor. If an alternate UA is used, the externally referenced UA profile would have to be altered too, with great difficulty. UA Profiles allow device vendors to monopolise UA developer's domain.

The content types the UA profiles details are exactly the same information contained in the HTTP headers given by the UA, except it is more difficult and expensive to parse the RDF. The reasons for the duplication is probably for non-web UAs for services such as MMS which do not fully implement HTTP like a Web UA should. MMS should be a web application, but instead via UA profiles and a crippled UA, mobile operators have made it an exclusive and lucrative service. Therefore UA profiles does very little help to bring the Web to the mobile. Mobile operators leverage these vertical market technologies from the W3C to create exclusive vertical services, purposely separated from the Web and the desktop environment. UA profiles does not promote device independence activity which contradicts this W3C working group's mission statement: "Access to a Unified Web from Any Device in Any Context by Anyone".

4 Engineering approaches

Four different engineering trends from device dependence to device independence are evaluated.

4.1 Device dependent engineering

There are several forms of device dependent Web engineering, however they can be all characterised by the way they output device dependent content. Unfortunately device dependent engineering can be introduced in many guises in other publications such as multi-channel content, reusable user interfaces, device abstraction or modelling [MPS03]. Some methods to add to the confusion may claim to be device independent because of the intermediate language used. This abstraction layer facilitates and encourages authors to engineer and adapt their content for particular user agents. These device dependent approaches will be argued against for their costly complexity and maintenance demands.

Device dependent engineering is typified by the assumption that HTML is a device dependent language. This is the unfortunate legacy of HTML without style sheets, without content separated from style. HTML with embedded style with the abuse of presentational markup is the typical reason why HTML can be found to work inconsistently across UAs. Immature UAs also share some of the responsibility for this misconception. Many UA implementations are flawed producing different results to the same HTML without special attention to the particular UA by the Web developer. Despite poor UA implementations, HTML is an UA independent language.

User agents are heterogeneous across vendors, devices and version numbers. The crucial problem is that these user agents fail to comply to Web standards in numerous serious ways. Those that do not comply, often also fail to degrade gracefully. Some other user agents might implement a non-standard proprietary extension which brings significant usability gains for users when the author takes advantage of the functionality. Therefore authors are often feel forced to address every user agent to squeeze the most usability out of their immature Web UA implementations.

The most common methodology for device dependent Web pages is to embed content into static HTML templates and publish it for a particular device or rather user agent. This is a simple way for Web developers to fine tune their content for particular devices. Web developers usually lose some of this control when opting for more abstraction, which may be needed especially for dynamic content. Modelling is an high-level abstraction where an intermediate language shapes content dynamically for particular devices in advance or immediately after UA capability sniffing. Various models can be used, for example the task, domain, device, dialogue, presentation and user model are abstracted in the intermediate language RDIXML. Some abstraction is always needed as HTML itself can be argued to be an abstract interface, however too much abstraction can make even very simple applications, very complicated. HTML's popularity and success lies within its relative simplicity.

Figure 8 describes how device dependent Web engineering works. There is typically no third party involved in the content transformation. The bulk of the work happens within the Web application using any one of the different markup, modelling or templating languages to generate device dependent HTML for every UA. UA type is determined either by various UA sniffing techniques or unique URLs.

Another reason for abstraction is the amount of heterogeneous devices and the different ways they are deemed to be used as information appliances. The browsing navigation model versus the task or transaction orientated model argumentation of the earlier WAP reappears. Instead of creating an information appliance for a mobile, a desktop and a printer for example, an author can describe these devices for generating content for. There are several available engines or languages perform this functionality, such as:

- RDIXML
- MyXML

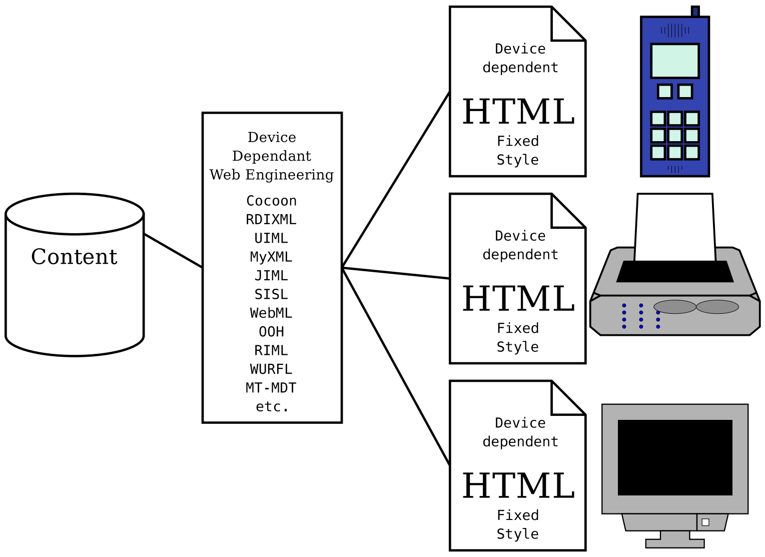

Content
Device Dependant Web Engineering
Cocoon
RDIXML
UIML
MyXML
JIML
SISL
WebML
OOH
RIML
WURFL
MT-MDT
etc.
Device dependent HTML Fixed Style
Device dependent HTML Fixed Style
Device dependent HTML Fixed Style

Figure 8: Device dependent engineering

- UIML
- WURFL
- Rendering Independent Markup Language

RDIXML is a task-based user interface language that breaks a Web application into tasks. Each task is arbitrarily mapped on particular device groups, such as PDA or WWW for desktop. The thesis describing RDIXML admits that its complex structure hampers the intelligibility of the UI constructions and directly affects the usability of the language. Additionally editing the XML is said to be error-prone and time consuming [Mar02]. Other elements that may prove problematic is how some task options are included or excluded on different devices, which would severely limit multi-modal access.

MyXML is an implementation from the Device-Independent Web Engineering (DIWE) [Kir02] framework, which is another heavy solution for outputting device dependent content. MyXML combined with device dependent XSL, a compiler and four basic run-time processors that require configuration in order to run as Web services that provide “device independence support”. The novel element with MyXML compared to other techniques is that it has provision for limited memory workarounds such as page splitting and process partitioning for the generated Web applications. Although splitting and process partitions requires an extra design effort. For every device MyXML supports a XSL needs to be made, a huge maintenance cost. XSL is a style sheet unlike CSS, is error prone and tedious to edit. The MyXML study admits the complexity demands of the approach required more than twice the amount of time as a normal Web application.

The User Interface Markup Language (UIML) is another XML based language for creating device-independent user interfaces. There are three stages in the UIML approach [Ree02] :

1. Device independent user interface elements
2. Device dependent style definitions for user interface classes
3. Content database for specific user interface elements for internationalisation and specific rendering environments

UIML is designed to be device independent, however it paradoxically performs this in peers with a separate vocabulary for each device. The next problem with UIML are how the user interface elements are based on Java user interface classes, which departs in several ways from the simple HTML form controls. For example the request and reply Web architecture does not implement UI synchronisation push messages that UIML uses. The UIML style definitions unlike W3C's CSS do not degrade if the device is not explicitly defined or allow the Web user an option to turn off the style sheet. Often the way HTML is generated is controlled by UIML definition vocabularies which makes the content mapping from UIML to HTML inflexible. HTML unlike generic user interfaces has hypertext features such as linking which UIML clumsily provides [Kir02]. Other problems associated with UIML is that content is inevitably intermixed with the user interface definition. UIML also does not attempt to cover design and maintenance stages of Web applications. UIML to its merit does has compilers for WML which older mobiles implement, VoiceXML and Java. Java applications could be deployed and updated on the very popular series 40 Nokia mobiles unlike their immature Web UAs, however they need to comply to a reduced mobile edition of the Java API. UIML transcoders and services are complex and suffer from performance issues [Ran03] which limits usability and functionality of Web applications.

The project named Wireless Universal Resource File (WURFL) [Pas04] exists to maintain a public database of devices in order for Web application developers to abstract away device differences. WURFL aims to overcome several of UAProf shortcomings. The WURFL project collects UA profiles from vendors and then corrects the data known to be wrong or missing, though WURFL database does not guarantee information to be correct and it is reliant on user contributions. WURFL's database can be installed locally instead of downloading UA profiles off vendor's websites, helping with performance. WURFL tracks over 400 devices with up to 100 capabilities and unlike some of the other device dependent engines has several real world applications. WURFL has bindings for several programming languages and accepts simple namespace prefixed syntax in HTML such as:

```
<wurfl:if capability="wap_push_support">
  <a href="subscribepush.jsp">Push Services</a>
</wurfl:if>
```

WURFL relies on UA sniffing which may not work on proxied connections. Dynamic adaption via the WURFL clauses may suffer from performance scaling problems. WURFL may tempt developers to squeeze usability by using non-standard device standard capabilities without providing an adequate fall back. WURFL Web applications can quickly become complex with the different esoteric device dependent options. When embedding WURFL into HTML, it does not cater for older mobiles with WML UAs. A developer would have to create another WAP application for the countless legacy WAP enabled mobiles. A Web application could be using WURFL as a kludge database to avoid UA bugs. There is a risk that certain mobile UA developers may not fix problems that WURFL workaround, which encourages more inelegant solutions. WURFL embedded in HTML may not address the device dependent need of splitting content due to common size limits. Deploying WURFL may prove too costly for simple static Web pages. WURFL highlights the immature state of mobile UAs, which often merits device dependent HTML generated by WURFL to achieve even basic access to information.

The Consensus project [Con04] is a European funded consortium of companies which produces the Rendering Independent Markup Language (RIML) specification and prototype implementation. The project aim was to enable access to Web content through run-time adaptation primarily for mobile devices. To clarify, RIML is not adaption as it does not adapt Web content, it compiles its own RIML language. Their approach like other device dependent techniques was to compile a list of devices and their immature UA features in order to output device dependent content to.

RIML is based on W3C technologies XHTML 2.0, XForms [DKMR03], SMIL and extensions. All these technologies are unlikely to be adopted by the Web. XHTML 2.0 for example is backwards incompatible with HTML or XHTML 1.0. XForms is a very large specification which has been criticised for so many dependencies such as Xpath and reuses so few of the technologies already implemented such as JavaScript or HTML. RIML extensions include pagination, layout and navigation in the language, utilising UA Profiles. The XML based language uses XSLT to transform the RIML markup broken up into device classes to generate particular content such as WML or XHTML.

The Consensus project claims to provide an implementation of a reverse proxy prototype that supported the use of RIML with no installation required on the client side. The prototype has a large amount of dependencies of resource hungry tools, that this service would be warranted in order for widespread RIML adoption by Web developers. Although it is unclear who take responsibility for such a service and how would fit into the architecture of the Web.

Despite RIML's strong support and funding, it is a another example of a heavyweight solution for generating device dependent code that is so complex that it is impractical. The project has been inactive for several months, despite the code being open sourced to attract developers. RIML is a sad testament to the unworkable W3C family of XML technologies in the last 5 years and their ignorance of the language of the Web, HTML.

These methodologies are all trying to compensate for a limited UA, but instead of the UA domain addressing the rendering of the Web application the Web developer is. A UA is a complex implementation that is largely in the domain of the device vendor to ensure compliance to Web standards and their device functionality. The UA should have some responsibility of making sure a Web application is usable and looks its best. For a content author to assume this role entirely that is in the also domain of the UA is unbalanced. A content author should focus on content, not on rendering issues of an immature UA.

Arguments for abstracting an application interface include voice, context and different paradigms of device usage. For reasons stated earlier in section 2.1, voice abstraction often proves unwarranted. Web applications in mobiles are often thought to require different interfaces in different contexts, hence the suitability for abstraction. An application that behaves differently under different contexts risks breaking multimodal access. There is no standard way of conveying contextual information yet and the need for such information in popular Web applications today is often unwarranted. There are also unresolved privacy concerns with most contextual information such as user location. Mobiles are also thought to utilise a task-based usage paradigm rather than a browsing one. With the same arguments in section 2.2 regarding task based navigation, the need for abstraction can be rebutted.

In the beginning of this thesis in section 2 we explored the problem of defining and describing a mobile. There are numerous obstacles to adequately define devices. You simply can not rely on the vendor's information or UA profiles for reasons iterated earlier. The market is quickly moving and different UAs are introduced to market every couple of weeks therefore device grouping could never be a one off solution. The descriptions would constantly need updating and fine tuning. The risks of not being vigilant is that a new device might not function with the device dependent application. These device dependent outputs are likely to be incompatible between user agents and break in unpredictable ways. Maintenance and testing costs tend to increase with the sheer amount of user agents and their different versions on the market. Often authors are inclined to only support popular devices to the exclusion of less popular ones. It is impossible to gauge the demand of a Web application with a device that has an unsupported UA that is unable to access it. It is difficult to record the inevitable lost business, as mobile users do not have as much opportunity to complain as email enabled desktop users. The usability problems of device dependent engineering are hardly avoided by this abstraction, instead they are arguably encouraged.

Different information on different devices will disrupt multimodal access. The formats that these techniques generates are not just typically incompatible between devices, the information can be also different. For example authors put more information on a desktop device and more likely to strip off menus and verbose footers for a mobile version. Another example is a URL that points to weather information about Finland on a desktop version may display the weather of several Finnish cities. On a mobile version the same URL address may only give the weather report of Helsinki. Device dependent engineering facilitates unpredictable information loss and inequality between devices.

Device dependent engineers typically generate different URLs for different devices. The root could be for the desktop, /mobile for mobile devices and /tv for TV devices. This pollutes URL space as in this case there could be three URLs pointing to the same or different information. The URL scheme or structure devised by the host is often difficult to design and predict, therefore users will find it difficult to find the URL most appropriate for their esoteric device. Templating can encourage poor URL addressing.

Generating device dependent content for many different UAs is expensive. Complexity and logic can directly downgrade server performance and therefore many methods generate pages periodically. This may mean some versions of the information may be out of date compared to others. Another weakness is that there is likely to be input the solution does not cater for.

As these pages are generated there is an emphasis on output. Information input, dynamic and responsive information now becomes harder to facilitate by the author. Static pages and static applications are much more likely with these techniques.

These methodologies do share several harmful common traits which affect device independence goals that should be acknowledged and best avoided. Device dependent techniques do cater for the millions of legacy UAs in existence, therefore not utilising such a method would limit Web access for many millions until they upgrade their UA. Due to the mobile device's characteristic of being difficult to upgrade the software, UA updates may be accomplished only when the consumer buys a new device. As many people perceive browsing the Web as the failure of WAP, then the attraction of a improved UA may not be enough. Some people however may be forced to upgrade due to degradation of the mobile's battery. This upgrade cycle could be as long as a decade or more. In the long term the harmful implications of adopting device dependent techniques arguably outweigh the short term benefits.

4.2 Adaption

The adaption methodology seen in figure 9 is a technique proposed by the W3C which is more suitable to vertical markets than the Web. Adaption is proposed to work by downloading the UA profile manufacturer's site from the address given in the header of the UA and parsed. Then a system not limited to the mobile device can be employed to adapt Web content dynamically. The adaption system would take in Web content and output content better suited for the limited UA of a mobile. The adaption system is deployed as a proxy server in the domain of neither Web developer or user agent. Using this technique could allow existing UAs to browse the legacy Web of poor markup. As user agents can not be upgraded easily, adaption is argued as a worthwhile technique to increase access. Conceivably the author does not have to worry about ensuring the content is well formed or well designed, as intermediate adaption will transcode the site to best fit the UA. Unfortunately this middleware "silver bullet" fix amounts to a complex expert system, that is not

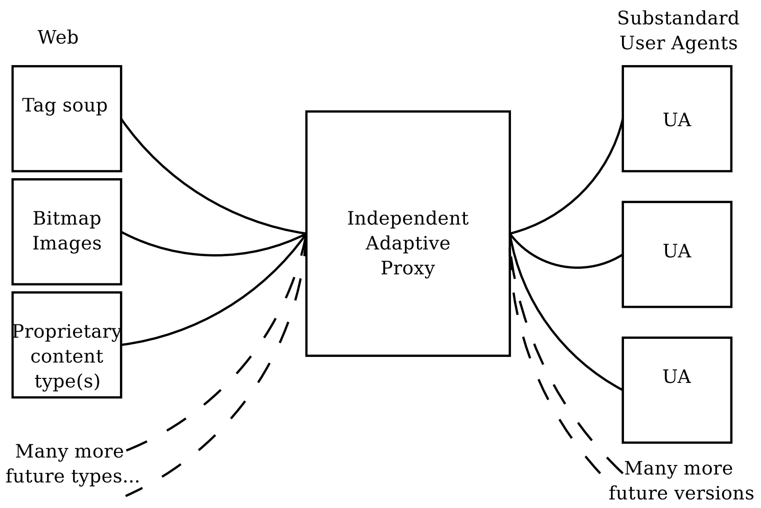

Web
Tag soup
Bitmap Images
Proprietary content type(s)
Many more future types...
Independent Adaptive Proxy
Substandard User Agents
UA
UA
UA
Many more future versions

Figure 9: Adaption [HM04]

suitable for the heterogeneous inputs or outputs.

Generating and outputting device dependent content shares all the problems of device dependent engineering discussed in section 4.1. Except with adaption we do not know the author's intent.

Information marked up in HTML can be distributed in a variety of incorrect ways as authors are often not careful about semantics or separating style from content. Old Web applications did not have the possibility of separating style from content as CSS either was not around or unimplemented by leading UAs at the time. This makes the task of parsing and tokenizing older content a considerable challenge.

Text/html may not be the only input to an adaption system. Style sheets, images and proprietary formats such as Flash may also be input. The adaption system should be able to accept this input, without losing any meaningful information.

One approach to identify elements in HTML is the Function-Based Object Model (FOM) [CZS+01]. It tries to group elements of a Webpage into semantic units by using general and specific rules and combining a myriad of techniques such as image analysis, decision trees, semantic analysis, clustering and link analysis. These various methods can be used effectively once the input structure has been analysed. Research acknowledges if the input changes unexpectedly the effectiveness of these methods decreases and they are expensive to update. This is why adaption is more suitable for vertical markets as the adaption system can expect a certain controlled quality of Web content as input. Web content with different designs and markups are introduced constantly, therefore input rules would need to be maintained. Input that isn't stable and not well structured can not be transcoded easily.

The adaptive proxy may need to discard inappropriate redundant information like needless options or advertisements in order to generate effective output for a small screen. Determining the quality and type of information in an Web content input stream is a near impossible problem. Some information may be left out with adapted content for devices like mobiles, that could be important information such as the authors details. Adaption needs to either intelligently guess the authors intent or the content owner needs to get involved in order to produce reasonable results. Adaption does not scale and is simply ineffective with the Web as its input.

It will not scale to all mobile devices unless there is a commitment to developing and maintaining the system. This technique may work in a closed vertical market where you understand the content and understand the UA, but on the Web this technique is wholly inappropriate.

From the efforts of the W3C we have learnt how their work billed for Device Independence such as XHTML and UA Profiles largely fail to address the real issues. The authoring technique of adaption proposed by the device independence work group was also shown to be flawed. However adaptive proxies do have the potential to extend the functionality of an UA by outsourcing expensive operations.

4.3 UA support

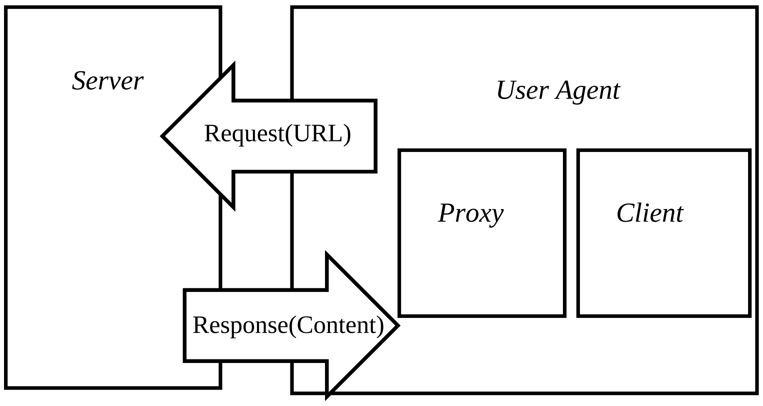


Figure 10: Proxy and client providing an UA

As UA choice is especially limited for mobile users, UA functionality could be exported to a proxy as seen in figure 10. As UA Profiles are not adequate, if an UA had expensive operations outsourced to a another more open platform then mobile UA could be enhanced.

Opera software have a proxy service dependent on their specific mobile Opera UA which works together to reduce traffic costs [Ope04b]. Opera's proxy claims to compress Web pages and eliminates unnecessary content before it is downloaded to the mobile phones. Opera estimates a significant 65% reduction in traffic costs.

The Power Browser is another example of an UA and an adaptive proxy working mutually [BKGM+02]. The proxy and UA work together to provide the user five levels of summarised information. The UA prompts and displays special icons for each mode while the proxy performs indexing, summarisation and produce keywords for serving to the client at request. The proxy performs UA dependent features, therefore they are reliant on one another and come together to provide a full featured UA. To distinguish from adaption, the proxy is in the domain of the UA, not the author or the network operator.

These architecture does have some scalability problems like that of adaption. Conceivably the UA proxy could serve several UAs although this was not addressed in the research. A proxy as an extension of the UA is a worthwhile approach to overcoming device constraints to provide a better browsing experience on a mobile.

4.4 Device independence

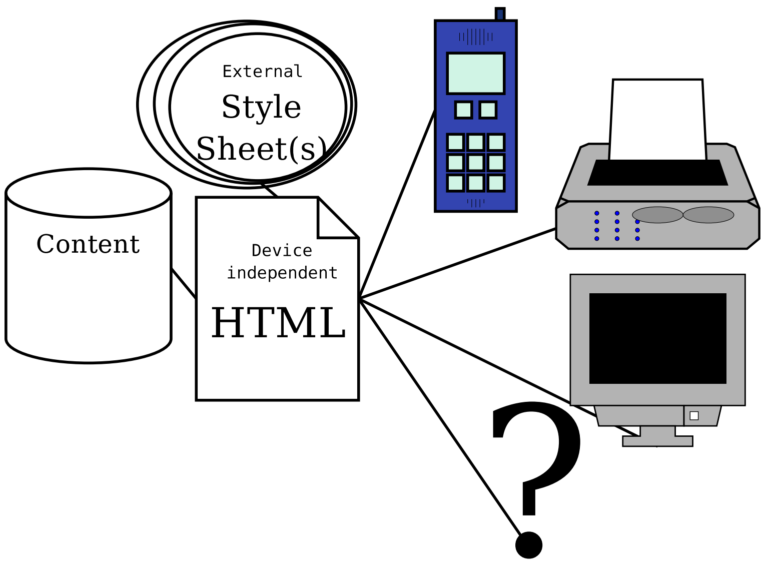


Figure 11: Style and content are separated with an emphasis on the UA

Device independence is a discipline where a single Web interface works satisfactorily on all devices without modification. That assumes that every device has an UA that implements a minimum specification of de facto Web standards. That minimum specification today is a dynamic subset of W3C's recommended standards. The specifications revolve around changing de facto standards that have invariably appeared due to various consequences of poor W3C leadership and UA implementations.

Harsh realities which feature quirky and incompliant W3C implementations, with UAs such as the market leading Internet Explorer UA of the desktop device and the Nokia UA of the mobile device. Hence Web developers can not rely on standards, their work instead is awkwardly dependent on numerous esoteric UA implementations of varying quality.

Defining exactly what is the minimum common UA implementation required for a Web application is relative and changing over time. The traits of today's reasonable Web UAs is one that implements HTTP, is able to accept invalid HTML and render it, all of which are difficult to implement. Web applications often require more from an UA, such as form processing for interaction, image handling and several conventions such as navigation aids presented to the user. Competing UAs such as Mozilla and Opera implement many more W3C standards than the minimum with every limited release. The lowest common denominator between UA market leaders is the moving target for Web application developers.

Validating Web applications to selected standards increases UA compatibility and hence device independence. The common denominator between UAs today is typically HTML4. However every UA implements HTML4 support in different ways, thanks to the ambiguity involved when liberally parsing HTML. Testing Web applications against targeted UAs is still very important, but validation services offered by the W3C ensures other UAs will have a good chance of working too. In the past UAs displayed specific behaviour independent of the device in order for Web applications to depend on a particular vendor of the UA. If Web applications are dependent on the eccentricities of a particular UA, then the vendor of that UA has a monopoly. This is harmful for device independence due to close ties between a device platform and the UA. For example if a Web application is dependent on a particular UA that was not ported to all device platforms, the Web application device independence is limited. UA independence via Web application validation by a more independent third party is fundamental to device independence. Validation increases the chances for a Web application to function on all existing and future user agents regardless of device.

Good Web applications are designed around short ambiguous URL structure, where a single interface ensures consistency. When adapting or modelling for particular UAs or device platforms different forms of Web content can come from a URL such as `http://example.com`. In order to prevent confusion and ambiguity, many Web applications are tempted to employ a unique and inconsistent URL schema for effectively the same resource for different device or UA groupings. Examples of this bad practise include `http://mobile.example.com`, `http://example.com/pda` or even the proposed top level domain `http://example.mobi` [BL04]. With a single common HTML interface the integrity of addressing of the Web is kept intact. Multimodal access from different devices is supported with portable, mnemonic and consistent URLs provided by URL design discipline.

With one Web interface generic proxying and caching becomes more effective. If a Web resource offers differently tailored content to particular devices, then generic caching may not work. For example if you are reading content from a Web application from a desktop device which has been cached by a proxy that was previously requested from a mobile device. The mobile dependent Web content on a desktop UA may look inconsistent or lack information that the user expects. A single interface generated by the device independent Web application means that generic Web proxying can be employed to save network costs and time.

Having a single interface allows Web application developers to focus on their user interface. With a myriad of generated HTML interfaces in the device dependent methodologies it is easy to lose sight of how the user interface behaves. One interface allows polishing that simply could not be catered for cheaply in modelling techniques. Mobile UAs might catch errors of the common interface that affect desktop users and vice versa. As there is less abstraction, a more robust and consistent application will more likely result. Less resources can be spent on testing different interfaces on different devices.

The practise of separating style from content into style sheets is the key to device independent engineering. For new Web applications this is easier to achieve than existing applications which do not already exploit CSS. Some common Web layouts featuring columns are quite difficult to achieve in CSS, therefore some large changes may need to be done to migrate across to CSS. CSS saves on bandwidth by caching the style sheet that is typically used across a Web application. However CSS can be poorly applied by being embedded in HTML and not in an external file. Hence the bandwidth saving will be lost and make UAs less likely to able to control their use. Having a style sheet separate from the HTML allows the UA to use it, switch it for another or turn it off altogether. This allows content always to be accessible, by removing a restrictive device dependent style that for example has too small text. CSS facilitates the Web on the mobile platform by economising bandwidth and allowing the possibility of controlling restrictive device dependent styles.

CSS support on popular mobile UAs on the Nokia series 40 is almost a regression, but there are practises for Web developers to workaround these problems. Some mobile UAs can not turn off CSS. There may be a good reason to turn off CSS, as some early implementations are very slow or worse, simply incorrect. Non-existent media types can be employed to turn off CSS styles before they come into affect. Assuming that the UA in question does not query CSS media types and processes CSS in order it is loaded, both of which applies to the Nokia UA, a workaround is availably to avoid immature UA bugs.

The device independent methodology more likely requires the use of relative elastic designs by Web application developers. As CSS media types can not possibly cover all devices in existence, one can not statically style every screen size or device. Style sheets need to be elastic enough to scale to every device. Unfortunately elastic designs for developers are more difficult than static ones. Static designs are often much easier to think about and implement for designers used to working with fixed dimensions. A static design is comparable to the print medium, where page size is known and exploited. Static designs are difficult for many user agents to render satisfactory. With style sheets however, these restrictive static styles can be turned off and hence avoided. Some advanced user agents such as Opera's Small Screen rendering can transform static designs into elastic ones to increase the pages of the Web application's scalability. However Web developers for the most part should give up some restrictive control and aim for a relative scalable elastic design for their style sheets. Generally Web developers should be designing for the device independent Web medium, not the restrictive static screen or print mediums that tend to be device dependent.

Whether screen sizes are large or small, the word count per HTML page should be be kept to a minimum especially with this technique. Many mobile user agents have small limits to what can be rendered at any one time. Pages that are too large are usually not even partially rendered, they are not rendered at all. As it is difficult to ascertain if this limit will be exceeded, the best practise of a minimum length should be employed. Web developers could remove all whitespace on their content output to save on precious bytes for example.

On larger screens too much text overwhelms a user, therefore this concise practise of limiting length can be applied to good affect in the desktop domain too. Copyright text and contact details typically found in footer texts of Web pages can often be placed in meta tags in the head element of HTML to economise on screen real estate and allow other programs such as the user agent to find this information more efficiently. Form inputs and controls should also be kept to minimum by splitting them across steps of the Web application. Filling in a long form is tiresome, however filling it in stages can be more rewarding and help find problems in the input quicker. The practise applies just as well on mobile user agents where large forms are known to be problematic. Splitting input forms assists multi-modal access. A use case whereby one begins filling entering input on a desktop and then finishes off the form on a mobile is much more likely to succeed with a form in stages. Web application data storage and security is the responsibility of the Web application, not the user agent with this technique.

The acronym and abbreviation tool (AAT) is an example of a Web application engineered to be device independent. As seen in the figures 12, 13, 14 and 15 in the appendix of the thesis AAT works on any UA tested with it. Through editing the URL or the single input and submit button an acronym can be looked up. The query matches any string of an acronym or its expansion. The results list both the acronym and its expansion. This Web application could be made more generic in order to query any database table's values. Although simple, this device independent Web application demonstrates how information can be queried from any device with a basic Web UA today.

Images as discussed do not scale and hence their use especially inline HTML should be avoided. Some mobile user agents and their users are unable to turn off image downloads, therefore a Web developer should take care not to use images alongside HTML text. Direct hyperlinks to images are better practise, as it gives users and user agents finer control on how to display that image. Allowing users to select an image from a Web page saves gives the user more control on a typically expensive mobile network connection. Direct hyperlinks allow user agents to display image across the whole screen, which is a better practise on mobile devices with limited screen size. Inline images in IMG tags within HTML on earlier mobile user agents typically are inaccessible or assume a particular device dependent screen size. Due to mobile characteristics such as limited memory sizes and high network costs inline images should be avoided, with best practise is to link images instead.

Whilst external style sheets offer a migration path for existing Web applications, new content types do not. The majority of existing user agents do not support non-HTML content types. If a typical mobile UA opens SVG content, that content will not be able to be displayed at all. If a typical UA opens HTML content with advanced unsupported style sheets, the user will still have access to content. Therefore this technique is limited to HTML as the base for delivering information, unless the vast majority Web content and user agents do not use HTML, which is very unlikely. Existing mobile platforms unlike desktop platforms usually are unable to install plugins to support new content types. Device independence technique requires new Web technology to be introduced via the ubiquitous Web text/html content type.

5 Outlook

Electronic devices, networks and information technologies should be generic commodities. This aim is emphasised throughout this thesis and its acceptance and support with each stakeholder playing their own role are needed to realise the Information society.

Discriminating between devices, such as arguing the requirements of a device are fundamentally different from another breeds inconsistency and inequality. This thesis has shown how difficult it is to define a mobile and how over a generation the technical gap between a desktop and mobile has narrowed considerably.

Setbacks occur when assuming one device requires a fundamental different form of navigation, markup or use than another device. The thesis dispelled further myths such as mobile inputs being inadequate or that small screen outputs were insufficient to read text.

This thesis affirms that the way we use a mobile device to access the Web, should be no different to the way we use any other electronic device. When usage paradigms are consistent across devices then we can achieve device independence where people can switch between devices at no cost. The sheer convenience and usability of multimodal access demands device interfaces converging on a generic UA Web interface.

Discrimination between networks has limited mobiles. The thesis reaffirms the Internet as the generic network for all devices to operate in. Attempts to separate networks in order to monopolise a market fragments the distribution of information.

This thesis has shown how exclusive mobile vertical markets have harmed progress. Mobile operators should become Internet service providers and compete in an open equal market. Mobile operators need to bill for open network access in a generic way, or risk discriminating users or fragmenting network access between fixed and wireless.

Not viewing the Web as the single information interface and to shun HTML's profound achievements harms the growth of the Information society. The thesis has shown how choosing anything else except the de facto language of the Web HTML has limited, divided and slowed access to information. The Web is an evolving medium in its own right. Viewing the Web through restrictive mediums such as paper and television limits it. Static styles of a desktop screen restricts the Web to the screen medium. The Web is a medium that must scale to any output, making it profoundly different to mediums we are familiar with such as newspapers and cinema projection screens. The Web is the superset medium in which other mediums should be derived and it should not mimic other mediums as shown in device dependent engineering.

Device manufacturers should refrain from needless differentiation of their devices, such as Nokia's keypad adjustments. A good generic responsive smooth scrolling screen would be more competitive commodity strategy for example.

The operating system should become more generic and open in order to allow UA software to update and upgrade. Proprietary platforms limit the entire Web by forcing Web developers to cater for non-evolving User Agents which are dependent on esoteric operating systems. UA developers need this generic open environment in order to improve their UA. Mobile UAs have largely been left behind desktop UAs because of the restrictive mobile platform. A generic operating system between devices would help UAs evolve in the same conditions, promoting compatibility by levelling the playing field for UA developers' products.

Web developers should build applications for the Web medium or risk fragmenting their work and focus. Up until adaption we saw Web developers taking the brunt of the weight of making the Web work on different devices, as they have been let down by poor user agents. Focusing on the Web medium requires switching to device independent practises which is difficult for developers accustomed to fixed dimensions of previous mediums.

Web users may also need to accept how information will be accessed through a generic UA interface. Although premature, the Web user agent is likely to replace the desktop rich client paradigm of access for most IT users. This may mean users will have to accept often poor but simple usability of today's Web form controls instead of their rich clients of the past. Generic applications calls for generic and stable usability.

The future for the device independent Web lies with HTML. The W3C Workshop on Web Applications and Compound Documents in June 2004 puts mobile device access again on a crossroads between impractical profiled XML based technologies and de facto Web standards such as HTML tag soup.

In the first generation, WML was chosen instead of HTML. In the second generation XHTML mobile profiles was chosen instead of HTML. Improving and extending the various complex error prone XML technologies that favours narrow vertical markets with expensive device dependent solutions has been shown to be the wrong approach. HTML tag soup is an undesirable erratic result of inevitable malformed Web content, that is not going away. An effort willing to accept these realities and provide direction is needed.

A collaboration between Opera Software developers of the Opera UA and the Mozilla foundation developers of the Mozilla UA, has created the Web Hypertext Application Technology work group (WHAT WG) [Web04b]. Their aim is to improve the neglected HTML specification in order to realise the Web's potential as an device independent application platform. They propose that Web application technologies should be based on familiar technologies such as HTML, CSS and JavaScript. This allows a clear migration path for improvement as well as backwards compatibility via degradation. These design principles including problems like how to define standardise error handling need to be addressed.

Already Opera has an UA port that ships on series 60 Nokia mobiles. Mozilla also has an UA port for mobiles named Minimo. With Internet Explorer's inactive development and decreasing market share on desktop devices, this may allow Opera and Mozilla to commodify the UA market. There is an exciting opportunity to push both workable standards and UA development forward making the Web medium easier to develop for and more usable regardless of device, unleashing the ubiquitous information application.

Suggested further research should delve into practical development and testing for Web UAs on mobile devices, together with distributed UA support from proxies. One of the major open problems discussed with the Web API and the mobile device is the need for a scalable bitmap image format. These topics as well as a commitment to existing Web technologies such as HTML is required to develop the Web API for greater access.

Despite these problems a new exciting era of ubiquitous computing via the popular mobile device unified by the Web API can be realised now with a device independent engineering approach.

References

ABD$^+$01 Altheim, M., Boumphrey, F., Dooley, S., McCarron, S., Schnitzenbaumer, S. and Wugofski, T., Modularization of XHTML. World Wide Web Consortium, 2001. `http://www.w3.org/TR/xhtml-modularization/`

Ann04 Anne van Kesteren, Unpractical specifications. 2004. `http://annevankesteren.nl/archives/2004/08/specifications`

BBC04 BBC News, Mobile phone sales dial up record. 2004. `http://news.bbc.co.uk/2/hi/business/3454811.stm`

BDDG03 Baudisch, P., DeCarlo, D., Duchowski, A. T. and Geisler, W. S., Focusing on the essential: considering attention in display design. *Commun. ACM*, 46,3(2003), pages 60–66.

BIM+00 Baker, M., Ishikawa, M., Matsui, S., Stark, P., Wugofski, T. and Yamakami, T., XHTML Basic. World Wide Web Consortium, 2000. `http://www.w3.org/TR/xhtml-basic/`

BKGM+02 Buyukkokten, O., Kaljuvee, O., Garcia-Molina, H., Paepcke, A. and Winograd, T., Efficient web browsing on handheld devices using page and form summarization. *ACM Trans. Inf. Syst.*, 20,1(2002), pages 82–115.

BL04 Berners-Lee, T., New Top Level Domains Considered Harmful. World Wide Web Consortium, 2004. `http://www.w3.org/DesignIssues/TLD`

Cap04 Capin, T., Mobile SVG Profiles: SVG Tiny and SVG Basic. 2004. `http://www.w3.org/TR/SVGMobile/`

CCVW04 Caldwell, B., Chisholm, W., Vanderheiden, G. and White, J., Web Content Accessibility Guidelines 2.0 (Working Draft). World Wide Web Consortium, 2004. `http://www.w3.org/TR/WCAG20/`

Con04 Consensus Project, Renderer-independent ML. 2004. `http://www.consensus-online.org/`

CVJ01 Chisholm, W., Vanderheiden, G. and Jacobs, I., Web content accessibility guidelines 1.0. *interactions*, 8,4(2001), pages 35–54.

CZS+01 Chen, J., Zhou, B., Shi, J., Zhang, H. and Fengwu, Q., Function-based object model towards website adaptation. *Proceedings of the tenth international conference on World Wide Web*. ACM Press, 2001, pages 587–596.

DKMR03 Dubinko, M., Klotz, L. L., Merrick, R. and Raman, T. V., XForms 1.0. World Wide Web Consortium, 2003. `http://www.w3.org/TR/xforms/`

EMC04 EMC Database and GSM Association, GSM on target to connect billionth customer in Q1. 2004. `http://www.gsmworld.com/news/press_2004/press04_06.shtml`

Eur04 European Commission, Information Society Policies. 2004. `http://europa.eu.int/information_society/policy/`

Gar04 Gartner, Mobile Communications Worldwide: Methodology and Definitions. 2004. `http://www4.gartner.com/resources/119700/119794/119794.pdf`

GSM03 GSM Association, M-Services Phase II Evolution. 2003. http://www.gsmworld.com/technology/services/

Hic02 Hickson, I., Sending XHTML as text/html Considered Harmful. http://hixie.ch/advocacy/xhtml

HM04 Hanrahan, R. and Merrick, R., Authoring Techniques for Device Independence. World Wide Web Consortium, 2004. http://www.w3.org/TR/di-atdi/

HMK98 Hjelm, J., Martin, B. and King, P., WAP Forum - W3C Cooperation White Paper. 1998. http://www.w3.org/TR/NOTE-WAP

Hyl97 Hyland, T., Proposal for a Handheld Device Markup Language. World Wide Web Consortium, 1997. http://www.w3.org/TR/NOTE-Submission-HDML.html

Jac03 Jacobs, I., Architecture of the World Wide Web. World Wide Web Consortium, 2003. http://www.w3.org/TR/webarch/

KAIM99 Kamada, T., Asada, T., Ishikawa, M. and Matsui, S., HTML 4.0 Guidelines for Mobile Access. World Wide Web Consortium, 1999. http://www.w3.org/TR/1999/NOTE-html40-mobile-19990315/

Kam98 Kamada, T., Compact HTML for Small Information Appliances. World Wide Web Consortium, 1998. http://www.w3.org/TR/1998/NOTE-compactHTML-19980209/

Kir02 Kirda, E., Engineering Device-Independant Web Services. Master's thesis, Technical University of Vienna, 2002. http://www.infosys.tuwien.ac.at/staff/ek/phd.pdf

KRW$^+$04 Klyne, G., Reynolds, F., Woodrow, C., Ohto, H., Hjelm, J., Butler, M. and Tran, L., Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. World Wide Web Consortium Device Independence Group, 2004. http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/

LB96 Lie, H. W. and Bos, B., Cascading Style Sheets, level 1. World Wide Web Consortium, 1996. http://www.w3.org/TR/REC-CSS1

LBLJ98 Lie, H. W., Bos, B., Lilley, C. and Jacobs, I., Cascading Style Sheets, level 2. World Wide Web Consortium, 1998. http://www.w3.org/TR/REC-CSS2

Lew03 Lewontin, S., Nokia Position Paper: W3C Workshop on Binary Interchange of XML Information Item Sets. Nokia, 2003. http://www.w3.org/2003/08/binary-interchange-workshop/02-Nokia-Position%-Paper_02.htm

LG02 Lie, H. W., C. T. and Glazman, D., Media Queries. World Wide Web Consortium, 2002. http://www.w3.org/TR/css3-mediaqueries/

Lit02 Little Springs, Inc., An Overview of Mobile Versions of XHTML. 2002. http://www.littlespringsdesign.com/design/xhtmlinfo.html

Mar02 Martikainen, A., An XML-based framework for Developing Usable and Reusable User Interfaces for Multi-Channel Environments. Master's thesis, Deparment of Computer Science, University of Helsinki, Helsinki, Finland, 2002. http://www.soberit.hut.fi/T-121/suomi/Antti_Martikainen-gradu.pdf

MPS03 Mori, G., Paterné, F. and Santoro, C., Tool support for designing nomadic applications. *Proceedings of the 8th international conference on Intelligent user interfaces.* ACM Press, 2003, pages 141–148.

Nie96 Nielsen, J., Why Frames Suck (Most of the Time). 1996. http://www.useit.com/alertbox/9612.html

Nie00 Nielsen, J., Readers' Comments on WAP Backlash. Nielsen Norman Group, 2000. http://www.useit.com/alertbox/20000709_comments.html

Nie04 Nielsen//NetRatings, Three Out of Four Americans Have Access to the Internet. 2004. http://www.nielsen-netratings.com/pr/pr_040318.pdf

Nok04 Nokia, Web compliant User Agent Header. 2004. http://ncsp.forum.nokia.com/download/?asset_id=11972

Ope01 Open Mobile Alliance, XSLT transformation sheets associated with WAP 2.0. 2001. http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html#xsl%t

Ope04a Opera Software, Opera for Smartphone/PDA. 2004. http://www.opera.com/products/smartphone/

Ope04b Opera Software, Opera mobile accelerator. 2004. http://www.opera.com/products/smartphone/accelerator/

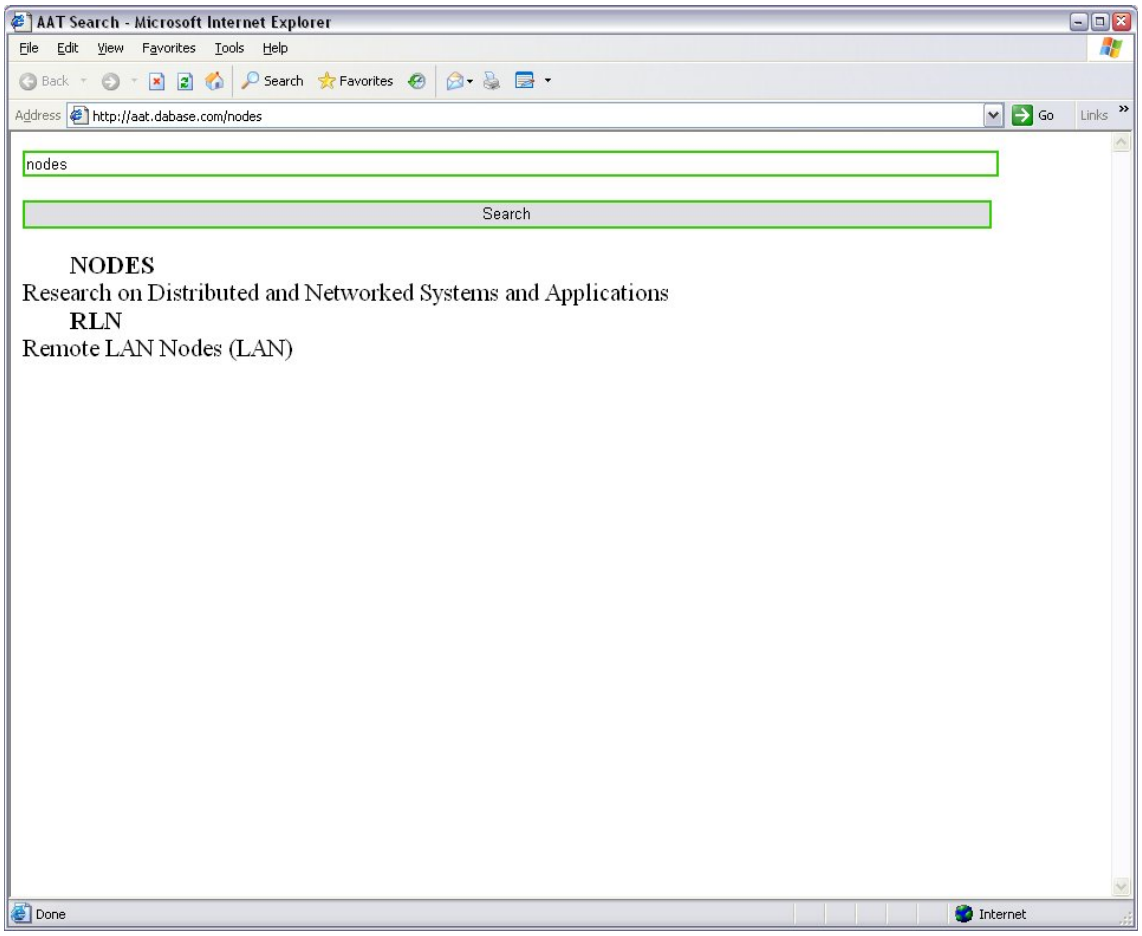
Paa01 Paavilainen, J., *Mobile Business Strategies.* Wireless Press, 2001.

Pas04 Passani, L., Wireless Universal Resource File. 2004. http://wurfl.sourceforge.net/

Ran03 Rantakokko, T., User Interface Adaption Based on Context. Master's thesis, University of Oulu, 2003. http://www.vtt.fi/virtual/adamos/material/rantakokko_t_diploma_thesis.p%df

Ree02	Rees, M. J., Evolving the browser towards a standard user interface architecture. *Third Australasian conference on User interfaces.* Australian Computer Society, Inc., 2002, pages 1–7.
RHJ99	Raggett, D., Hors, A. L. and Jacobs, I., HTML 4.01 Specification. World Wide Web Consortium, 1999. `http://www.w3.org/TR/html401/`
Rya04	Ryan, J., European Internet statistics. 2004. `http://www.altevie.net/mediagraphix/europeaninternetstats/`
The04	The Counter, Global browser statistics. 2004. `http://www.thecounter.com/stats/`
Val00	Valdes, R., Waiting on WAP. 2000. `http://www.newarchitectmag.com/documents/s=4737/new1013637144/`
WAP98a	WAP Forum, WAP Architecture. 1998. `http://tinyurl.com/ysxlh`
WAP98b	WAP Forum, WAP WAE. 1998. `http://tinyurl.com/ysxlh`
WAP01	WAP Forum, WAP CSS Specification. 2001. `http://www.wapforum.org/tech/documents/WAP-239-WCSS-20011026-a.pdf`
WDSR02	Wugofski, T., Dominiak, D., Stark, P. and Roy, T., CSS Mobile Profile 1.0. World Wide Web Consortium, 2002. `http://www.w3.org/TR/css-mobile`
Web04a	Web Corp, Latest Mobile, GSM, Global, Handset, Base Station, and Regional Cellular Statistics. 2004. `http://www.cellular.co.za/stats/stats-main.htm`
Web04b	Web Hypertext Application Technology work group, Position Paper for the W3C Workshop on Web Applications and Compound Documents. 2004. `http://www.w3.org/2004/04/webapps-cdf-ws/papers/opera.html`
Wor01	World Wide Web Consortium, Device Independence Working Group Homepage. 2001. `http://www.w3.org/2001/di/`

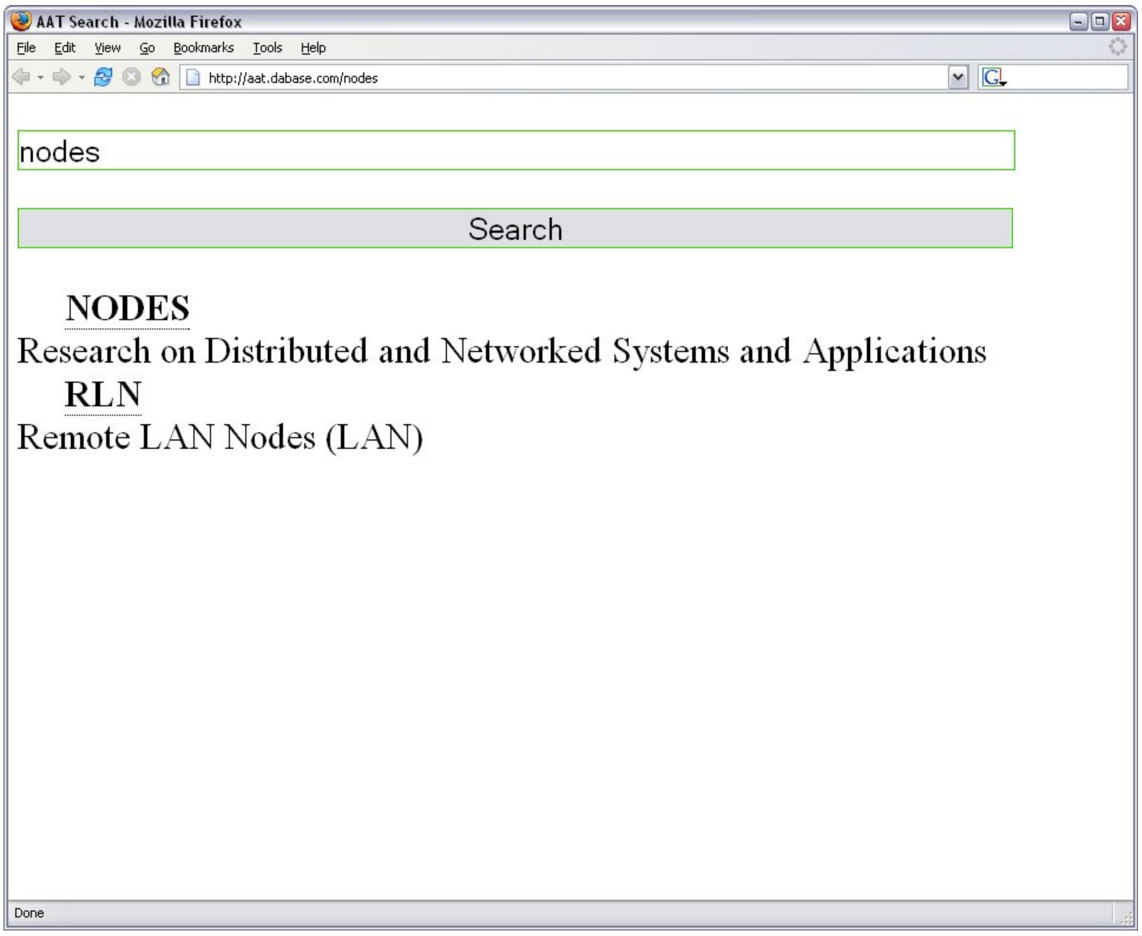
Appendix Acronym and Abbreviation Tool (AAT) screenshots


AAT Search - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
http://aat.dabase.com/nodes
Go
Links
Search
Done
Internet

nodes

NODES
Research on Distributed and Networked Systems and Applications
RLN
Remote LAN Nodes (LAN)

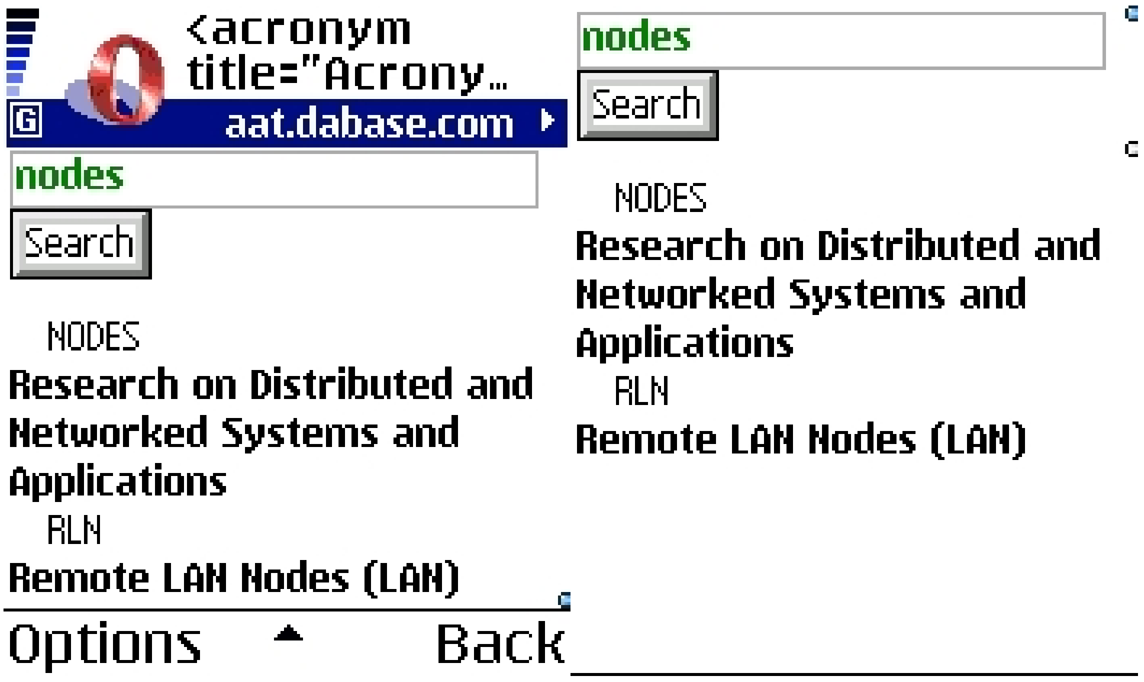
Figure 12: Microsoft's Internet Explorer user agent rendering AAT from a Desktop PC


AAT Search - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://aat.dabase.com/nodes
Search
Done

nodes

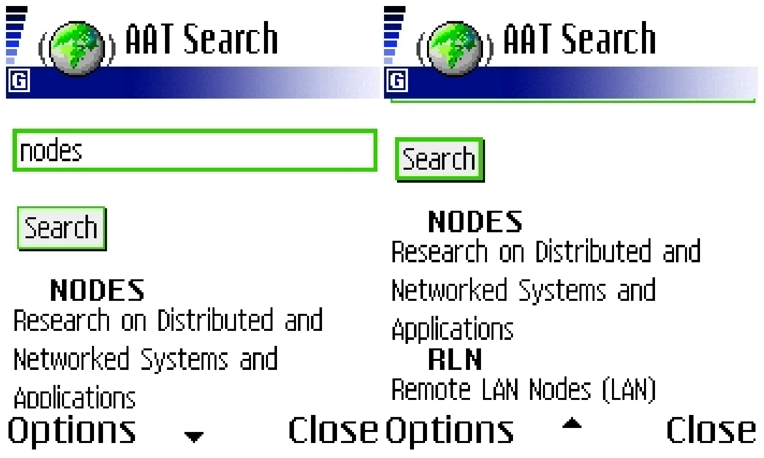
NODES
Research on Distributed and Networked Systems and Applications
RLN
Remote LAN Nodes (LAN)

Figure 13: The Mozilla Firefox user agent rendering AAT from a Desktop PC


<acronym title="Acrony...
aat.dabase.com
nodes
Search
NODES
Research on Distributed and Networked Systems and Applications
RLN
Remote LAN Nodes (LAN)
Options Back

nodes
Search
NODES
Research on Distributed and Networked Systems and Applications
RLN
Remote LAN Nodes (LAN)

Figure 14: The Opera user agent rendering AAT from a Nokia 6600


AAT Search
nodes
Search
NODES
Research on Distributed and Networked Systems and Applications
Options Close

AAT Search
Search
NODES
Research on Distributed and Networked Systems and Applications
RLN
Remote LAN Nodes (LAN)
Options Close

Figure 15: Nokia's in built user agent rendering AAT from a Nokia 6600