WAP enabling existing HTML applications
Marcin Metter, Dr Robert Colomb
Department of Computer Science and Electrical Engineering
The University of Queensland, QLD 4072, Australia
metter@csee.uq.edu.au

Abstract
Already existing HTML applications can be converted into WML applications for use on WAP-enabled devices, yet this process is not as simple as the alteration of the markup tags. This paper investigates the problems associated with the conversion process by examining the conversion of a functional information system that would be of benefit to its users when access from remote locations is required. This paper details a number of problems discovered with the conversion of complex HTML documents into simplified WML documents. The major problems of displaying tabulated data, hyperlinks, navigational aids, and user input are discussed, with possible solutions presented. The paper concludes by raising the issue of 'intelligent' automated HTML to WML conversion, a topic for future research.

1. Introduction

Until now the adaptation of access to Internet based information services via wireless mobile devices has been very limited due to the fragmentation of the potential market with no standard protocol encompassing all mobile service providers [1]. In 1997 the mobile device manufacturer, Ericsson, took notice of this problem, realising that greater adoption would only be possible if all manufacturers adopted a generic protocol. A consortium, consisting of Ericsson, Motorola, Nokia, and Unwired Planet (now Phone.com), was set up to define the new common protocol.

In April 1998 the first version of the new protocol, called the 'Wireless Application Protocol' (WAP), was released. It comprises of a five-layer model, consisting of transport, security, transaction, session, and application layers, as shown by figure 1, giving greater flexibility to the already existing Internet model [10]. The model was developed to support functionality on a variety of wireless networks and to allow further development of each layer without affecting the layers above or below it [10, 1].

![Figure 1: The WAP protocol stack and corresponding Internet hierarchy [1]](image)

Of the five layers, application design is concentrated on the first or the Wireless Application Environment (WAE) layer. This layer contains the definition for a micro-browser and the Wireless Markup Language (WML), both of which are optimised for the limited memory and processing power of current wireless mobile devices [9]. Also provided is a lightweight scripting language, WMLScript, and an interface for accessing telephony services and programming interfaces, Wireless Telephony Application (WTA or WTAI) [9].

The definition of WML has only 37 markup tags, as opposed to the HyperText Markup Languages (HTMLs) 90+, due to the limited input and output capabilities of current WAP-enabled devices [11]. With HTML currently being the language of choice on the World Wide Web (WWW) many of the important features of HTML are incorporated into WML, although new WAP specific features have also been added [3, 11]. In particular WML adapts a new document metaphor in the form of 'cards' and 'decks', as opposed to 'pages' as used by HTML (Figure 2) [11].

![Figure 2: Web document to Card/Deck relation](image)
The conversion of HTML documents into WML therefore requires the modification of not only the markup tags but also the document structure [11]. The WAP forum and a number of external parties have proposed the following solutions to the conversion process:

1) Using eXtensible Markup Language (XML) documents, with transformation into WML or HTML via an eXtensible Style Language (XSL) processor [10].
2) "What You See is What You Get" (WYSWYG) software that outputs documents in both HTML and WML formats.
3) Automated HTML to WML conversion software.
4) Wrapper software to convert HTML documents into WML documents as a site is accessed [4].

Ideally, the final two methods could be used to provide users with access to already existent HTML documents, removing the need for document re-authoring. Yet the complexity of some documents prevents these methods from being highly successful. In order to show the complexities and possible ineffectiveness of the rule-based automated processes, the manual conversion of a functional HTML application into WML was conducted with an examination of the whole process and of the final result for conversion problems.

2. The HTML application

The HTML application converted was a functional prototype of the application described and documented by Guerini, in "Interfacing Essential Drug Informatics", that provides precise drug information to practising veterinarians [7]. The motivation to convert this application originates from the fact that access to the drug information system may be required whilst away from the office, and a desktop computer.

Ideally access could be provided via a device that is already carried by them, such as a mobile phone, but until recently this has not been possible. The introduction of the web-enabled mobile phones promised access to the Internet, but was limited to selected e-mail, weather, stock, and news information services. WAP is set to remove this limitation, allowing the development of applications that are accessible via both desktop computers and wireless devices.

3. Conversion from HTML to WML

When designing for the WWW, document content layout is important, but due to the large amount of display space even badly designed documents are still readable. This is not true for WAP applications, with the document layout playing an integral part of application design, as badly designed documents can be rendered useless. The conversion of HTML documents into WML must therefore include an examination of document layout and how it is to be optimised for WAP applications.

Two software packages have been used in the conversion process:

- An HTML tag editor (CuteHTML 1.2, from GlobeScape)
- The Nokia WAP Software Development Kit

The first package was used to examine the original HTML documents, and for basic editing of the WML documents. The second was used to test and debug the WML system, with an emulation of the Nokia 6110 mobile telephone being selected as the wireless mobile device.

The HTML application converted was composed of a large number of HTML documents, interlinked by hyperlinks in a hierarchal structure. Each document falls under one of the following five formatting and layout categories:

1) Simple text and hyperlink
2) Complex option lists
3) Long simple option lists
4) Complex text, multiple/single row and column table
5) Complex, long text

3.1. Simple text and hyperlink (About.htm)

An examination of this document identified three data types: images, informative text, and hyperlinks. The images were the least significant in providing information and were used to 'beautify' the document. The informative text provides the user with details about the application, via the title, and a prompted question, via the body text. The hyperlinks make up the rest of the body text, by providing the user with a set of answers that are selected by clicking on them.

In order to convey the same information over a limited WAP device, the WML document must contain the details that allow the user to make a selection and proceed to the next question. In this document the important data types are the informative text and the hyperlinks, without the images as they would only take up valuable display space.

Figure 3: (a) HTML and (b) WML opening screen(s)
Figure 3, above, shows the opening screen(s) as viewed by a desktop computer (HTML) and a mobile phone (WML) based browser. The original document introduces the application and provides the first set of options on a single screen. Yet this would crowd the limited display area of the mobile phone, forcing the list of options to be displayed off screen and thus requiring the user to scroll.

By separating the introduction and the first question/answer set into two screens, with an automatic transition between the two, the user is no longer required to scroll down the screen in order to examine the options, with all of them in full view.

As the limited display width is capable of showing only a limited number of characters per line, depending on the device being used, text past the right-hand edge either wraps onto the next line or is truncated. If the option text is too wide to fit onto a single line it will either be split into two or more lines, or will be truncated, thus reducing either the readability or useability or both.

By reducing the display font size, if available, or using alternative representation, such a Just for Justabout, More for Moreabout, and Latest for Latest about, the possibility of the problem occurring can be reduced or removed.

In the HTML version a mouse pointer is used to select the correct option, allowing the positioning of the hyperlinks to be random (Figure 4a). The WML version requires a mobile phone or similar keypad to be used (Figure 4b). With the test mobile phone keypad only having a vertical, up/down scroll key, optimum user feedback should have the highlight moving in the same direction as the button is being pressed.

For this reason the document layout of the WML version is very important, with optimum user feedback provided by laying out the options as a vertical list.

It must be noted that although future mobile phones, such as the Ericsson R380S, may include sideways scrolling, it is much better to present users with a vertical list of options. This is because data present past the side edges of the screen may not always be obvious or accessible on all devices [6].

Figure 4: (a) Mouse pointer and (b) Mobile-phone (NOKIA 6110) keypad

3.2. Complex option lists (Justabout.htm)

In this document the layout of the information text and hyperlinks is highly important, with a table being used to associate the choice made with not only a species but also a drug family at the same time (Figure 5a).

Examination of this document highlights four data types: a header, the information depth selected, a question, and a set of answers. The header of the HTML document takes up a lot of screen space yet does not provide any new information, it is therefore not required in the WML version. The data types that are important, and must be displayed, are the information depth selected, the question being asked, and the available answers.

As the user moves around the drug information system they must be helped maintain their orientation. This can be achieved by presenting a document title, via the title bar, that highlights where within the information system they are currently located. To utilise the screen space effectively, this title must be informative, in this case the depth of the information selected or 'Just About' was used.

To prompt the user for further input the instructions, via a question, are presented. Ideally the title bar would contain this question, as it is visible to the user even after scrolling down the screen. Yet it is already used to provide the orientation information and also the question does not fully fit into the title bar. Since the title bar does not support word wrapping, it would be truncated, thus rendering the question ineffective, as it cannot be fully read.

In order to solve this problem, the question has been moved so that it is presented above the option table (Figure 5b). Allowing the user to see the question and understand what is required of them before they are presented with the options. This is because the top of the document is presented first and the user usually examines the output in a left to right, top to bottom direction.

The conversion of the system should be functional as both an HTML and a WML based application. It is therefore advantageous to maintain a relatively high level of consistency. For this document the table structure used in the HTML document is important and must be used in the WML version (Figures 5-1 and 5b).

Figure 5: (a) Original and (b) converted 'Justabout.htm'

With the emphasis on consistency a navigational problem with the WML version is introduced, since in order to scroll through the various drug families a user must scroll though each Dog/Cat combination entry before the next drug family is reached. Thus the navigation of the document is in a zig-zag manner, which requires a long period of time to scan through the
available entries and can cause some confusion (Figure 5a). Optimum user feedback is also disrupted, as pressing the down/up scroll key results in the highlight moving sideways every second press.

Allowing a user to scroll through the drug families for each species before the selected species changes (Figure 5b), allows optimum user feedback to be applied. Yet the selection process still requires a long period of time, due to the need to scroll through all of the drug families for one species before the next is selected.

Using the card/deck document metaphor to separate the drug family and species into two separate questions, provides a much faster interface, but results in the inconsistency with the HTML version being lost.

When devices that are capable of sideways scrolling and selection are developed this navigation problem may no longer exist. However, until sideways scrolling is available in all devices, during the application conversion process it must be decided which of the above two methods to use, depending upon the amount of data presented by option lists such as this, and the amount of HTML-WML consistency required.

![Image](image1.png)

**Figure 6:** (a) Zig-zag and (b) Vertical navigation of 'Justabout.htm' menu

### 3.3. Long simple option lists (Djantibi.htm)

A long listing of options can be achieved by simply using a list of hyperlinks, with one hyperlink per line. The example document provides such a list, with the drug names being listed as a single column of hyperlinks (Figure 7a). Although the WML specification includes an option list function, it was not used as the device used determines the display of option lists, whereas full control of the option list was required so as to keep HTML-WML consistency.

With the list requiring a small amount of screen width, the HTML document uses a table with a coloured background, to focus the user on the list itself (Figure 7b). The WML version does not require a table with a background image to be used, as the display screen is already sufficiently small enough to focus the user on the options list. However, the WML standard does not provide support for background images as the image format defined for WAP uses black and white images, making the reading of the black text very difficult.

In order to aid user orientation, the title bar presents the drug family being accessed. A further title is also included to provide title consistency with the HTML document but it is placed above the option list at the top of the document body using a different text format to that of the drug list.

![Image](image2.png)

**Figure 7:** (a) Original and (b) Converted 'Djantibi.htm'

### 3.4. Complex text, multiple/single row and column table (Djamoxic.htm)

Documents with complex structures such as tables with multiple rows and columns are usually used to present a large amount of information in a compact form (Figure 8). Yet direct conversion from HTML or WML is not always possible, due to the screen width of most WAP-enabled devices.

The document examined contains a horizontally aligned table which when directly converted into WML positions data past the edge of the visible display area. For devices without sideways scrolling capabilities the information past the edge is lost, due to truncation (Figure 9a) [5]. However, by allowing text within columns to wrap to the next line, the table will automatically re-size itself so that it fits within the available screen width.

This solution is limited to simple tables, such as that of figure 8. For tables with either large numbers of columns, or large amounts of text within each cell, or both, the end result does not always produce a readable table, as illustrated by figure 9-2.

![Image](image3.png)

**Figure 8:** Original 'Djamoxic.htm'

An alternative solution is to convert the horizontal tables so that they have a vertical layout by changing columns into rows, and vice versa (Figure 9c). The success of this solution depends on the number of rows a table has, with table readability suffering as the number of rows is increased.
Yet another alternative is to convert the data from a table-based layout into a formatted text-based structure, thus removing the problems associated with tables as detailed above.

By assuming that the first row or column of a table structure contains the headers, and the other cells the content, the conversion process separates the header and content data and interlaces them by matching the content with the appropriate header (Figure 10).

In order to differentiate between the headers and the content different formatting can be used for each. For the example document the header is displayed using a medium sized, bold, and underlined font, whilst the content can use a small sized font, with no formatting features. To stop truncation, the text is set to wrap-around as the right hand edge of the display screen.

This new display format vastly improves the readability of the document, with the text wrapping allowing document text to adapt itself to use the full width of the screen, much like in desktop-based web browsers.

When more than one row of information is present, as it is in the example document, with each row being related to a drug with a different 'Trade name' or 'Active ingredient(s)', each row's information must be indicated. This can be achieved by using a single column, multiple row table (Figure 12), with a row boundary indicating where information about a new drug begins.

The 'more about' equivalent of the example document incorporates a number of hyperlinks to other documents that provide the 'extra' information, requested. Depending upon the device, the way that hyperlink text is indicated varies, but it is usually highlighted in the same way, the inversion of the text and background. If a large section of the text is related to a hyperlink, as it is in the example document, the text can become very hard to read, as shown by figure 13.

When this occurs it is much better to provide a list of hyperlinks at the end of the document, with a new indicator, within the text, being used for the association of the hyperlinks with the appropriate content in the body text. The use of a referencing type of system is a possibility.

3.5. Complex, long text ('Clamixic.htm')

The example document again uses a table with a different background to focus the user on the information provided. This time there is a very large amount of information provided within the table, but it still serves the same purpose, to focus the user (Figure 14a). Although the different background cannot be used, for reasons detailed earlier, the table itself can still play a functional part, as a method of separating the title and the drug information (Figure 14b). This arrangement is necessary as on the device used for testing the WML version, the underlining and bolding of small text, used for displaying the title, was not permitted and would cause the drug information and title to blend together.

The HTML version of the document contains a hyperlink to the MEDLINE medical database. The major
problem with this implementation is that it requires the user to enter a search string, after being forwarded to the database system. Text entry via a WAP-enabled device is tedious, and so the system should query the MEDLINE database upon access to the document, and provide the user with a list of articles found that matched the topic.

As the information returned by the site is not in WML format, an error would occur on a WAP device. A possible solution would be to execute a specified query, on the MEDLINE database system and convert the data returned into a WAP-compatible format.

![Image](image.png)

**Figure 14: (a) Original and (b) Converted ‘Clamixic.htm’**

4. Observations

The conversion of a HTML-based application into WML format is not achieved by simple replacement HTML markup tags with their WML equivalents. It is a process that is more complicated, with the designer being required to consider a number of issues if a functional WAP application is to be made. These considerations can be examined under the following three categories:

1. Document encoding,
2. Text formatting and layout, and
3. User input and navigation.

Although each of these can be examined separately, they must be combined in order to produce an effective model for the system.

4.1. Document encoding

The Wireless Markup Language (WML) specification, version 1.1, consists of 37 markup tags, as opposed to HTML’s 90+ [12, 8]. These markup tags range from event and navigation handling to document structuring and layout, with the tags being optimised for the capabilities of wireless mobile devices, whilst providing the application author with enough freedom to display the information requested in a user-friendly format.

The WML specification has been developed using XML and therefore inherits a number of the rules and features that are specific to XML. These rules are [9]:

- All elements, attributes, and enumerated attribute values must be entered in lowercase, due to case sensitivity.
- Each file must begin with a document type declaration (DTD). The following is used for WML.

```xml
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-/WAPFORUM/DTD/wml 1.1/EN* "http://www.wapforum.org/DTD/wml_1.1.xml">
```
- All attributes must be surrounded by single (`) or double (`") quotation marks.
- Empty tags must end with a forward slash (`/`). (i.e. `<br/>`)
- Multiple white spaces are treated as single white spaces.
- The ISO/IEC-10646 Universal Character Set (or Unicode 2.0), including any of its subsets, is the page encoding language used[5].

Although these rules are strict, they promote the design of documents that are highly structured and simple to internationalise. However, this does mean that badly formed HTML documents are harder to convert into a WML format as document restructuring may be required, to make them well-formed. A process that can be somewhat automated.

4.2. Text formatting and layout

The information that is transferred across to the end-user is typically formatted text, with the addition of images and other multimedia components to 'enhance' it. The current WML specifications, combined with the transfer speed and display limitations, do not permit extremely complex images or other multimedia to be transferred. This leaves the textual information as the major source of information.

WML limits text formatting to three font sizes: big, normal, and small, and three formatting features: bold, italics, and underline, effectively leaving only one font type available. Although this is a major limitation for the presentation of information, it does have its advantages. Firstly, the information is presented using a font that the manufacturer has selected to provide the highest readability on the device's screen. Secondly, the need for the author to know if a font is available on a targeted device is removed. Thirdly, extra memory to store and processing power to render 'temporary' or rarely used fonts is not required.

Highlighting, separating, and identifying different types of information must therefore be achieved using older typographical standards combined with the available layout features.

The layout of information within a document is very important as data can be placed in areas not visible to the end-user and text wrapping can make it unreadable.
Figure 15: Header/content separation and reformatting approach

As illustrated by figure 9, the design of tabled data is very important with the table structure being dependent not only on the data contained but also the amount of data within each cell and its formatting. Tables that are originally designed using a horizontal layout structure must be reformatted before they are presented on a WAP device, due to display width restrictions. A number of possible approaches, using the available WML markup tags, to solving this problem are available.

**Horizontal to vertical table conversion approach**

By far the simplest method is to convert horizontal tables into vertical tables. The success of this approach is highly dependent upon cell content and/or the number of rows that a table has. Headers and cells with large amounts of text can be wrapped around so that both the header and the content fit into the available cell width. This may cause problems with readability with the text being presented over two lines, and possibly wrapping in a location that causes confusion, as illustrated earlier by figure 9-3. Due to this abbreviation the header text may be required to provide the content text with more cell space.

The number of data rows that a table contains also affects a table display, as each column added reduces the amount of display space for other columns, causing more text wrap-around, more frequently. Thus, this approach is not suitable for the conversion of tables with more than one row and possibly complex data.

**Header/content separation and reformatting approach**

By dissecting the header and content components and re-assembling them into a vertical table, using a header-content interlacing approach, the entire width of the display screen can be used to present the information (Figure 15). This method assumes that the header text is either the first row or column of a table, with all the other cells containing the content.

This approach allows tables with multiple rows and columns to be displayed, since the resulting table that has a single column and multiple rows. This table easily fits into the screen width. Even though all of the original tables data is now accessible, readability is reduced, as the separation of the data into rows of related information is lost.

Adding extra information to the table, such as, a blank cell between each set of row data or a header detailing to which row the data belongs two can solve this problem. The user can also be forced to scroll a very long distance to reach data closer to the end of the table.

**Cards and deck approach**

The 'Header/content separation and reformatting approach' can form very long tables, causing a user to scroll a long way to reach entries that are further down the table. The card/deck document metaphor induces splitting the rows into a set of smaller tables, stored as cards, but stored together, in a single document (Figure 16). This removes the need to scroll long distances between each row's data, instead requiring the user to swap between cards.

With the information being presented over a number of screens, the user must know the number of cards present, easily achieved by indication that the card is a part of a set through a header, such as, "card 1 of 2". Navigation between the cards can be either through a single 'next card' action key, or by assigning each of the numbered keys on the keypad to a specific card. This is highly dependent on the device used.

Figure 16: Cards and decks approach

**Single page re-format approach**

The 'header/content separation and reformatting' and 'cards/decks' approaches can be enhanced further by converting the table into a list of information instead of a vertical table. The information is then
presented using text formatting to differentiate the headings from the content.

By using a simple text based system the display space is used more efficiently as the area taken up by the table borders is reclaimed, therefore allowing more text to be displayed on the screen at the same time. Also the use of tables requires more WML code to be present in the document file.

Complex tables

All of the above methods have been developed for the display of simple horizontally tabled data that is used to display specific information about related topics. More complex applications of tables, such as the use as a comparison tool requires a more complex approach, which is easily derived from the detailed methods.

For these types of applications a horizontal table may be necessary. The table must therefore either be designed with the assumption that the user’s device is capable of scrolling sideways or the table can be converted in a different way, with each ‘card’ containing a column of the original table, with the deck making the complete table. This method can also be used for tables that compare data where rows contain the common data, by initially converting each of the rows into column and then performing the conversion process.

4.3. User input and navigation

User interaction in an application is very important if the system is to be accessed consistently. WAP-enabled devices input is significantly different from that of a desktop based web browser that permits keyboard, mouse or touch screen character entry. WAP-enabled devices have a wide variety of input methods, although the WAP specification requires that they have at least a scroll and a two-button assignment capability [13].

The entry of text can therefore be a very complex task, and this requires that applications use as little textual data input as possible, instead using menu-based, hierarchal, or multiple-choice types of interfaces.

As textual entry may be required for some applications the user should be provided with as much help as possible. This can be achieved through the use of questions that split the input into sets of ‘cards’ or a smart “predictive text input” system, such as that designed by Nokia [6]. Speech recognition is another possibility, especially since mobile phones already have voice capability built-in.

The HTML navigational model, using hyperlinks to link pages together, is also used for WML applications, with the ability to insert hyperlinks anywhere within the document text, resulting in their positioning in various places around the display.

On PC-style devices this does not cause any navigational problems, since the user can see large portions of a document simultaneously, using little or no scrolling. On a small screen, however, the amount of text presented simultaneously is greatly reduced. Thus the need to scroll through the document to find and examine the hyperlinks increases [2]. By providing a ‘quick link’ or ‘menu bar’ that lists all of the available hyperlinks within a document navigation can be improved. With the user no longer being required to scroll through and read the entire document to find a hyperlink, an HTML like navigation experience is possible.

The ‘quick link’ or ‘menu bar’ is easily implemented by either:
1) Listing the hyperlinks somewhere within the document, such as at the end.
2) Using ‘cards’ with a listing of hyperlinks.
3) Assigning ‘pop-up’ list to a function key.
4) Assigning the keypad buttons to specific hyperlinks.

The method that is chosen is dependent on the amount of consistency between the HTML and WML versions that is provided. Methods one and two are best suited as they can be adopted for both versions. Methods three and four are more WAP specific, although it is possible to emulate them in HTML.

In order to aid user orientation, the user must be provided with navigation prompts and location identifiers. The method used by an application can be varied, with the title bar or the first few lines of a document being used to inform a user of their location within the system. Although whole words can be used in the title bar, word length is limited to a device’s display width and text wrapping not permitted [12]. It is therefore much better to either use a coded method for the title bar, for example a number or letter representing the location, or by presenting the orientation information on the first one or two lines of the display, above the information content.

5. Conclusions

In conclusion the task of developing an application for both the WWW and the WAP domain requires knowledge of not only the limitations that are imposed with the WAP version of the application, but also methods of designing effective user-interfaces. This paper has documented the conversion of a simple HTML application, in order to highlight the complexity of this task. Although completed manually, to better understand the actions that have to
be taken, a large portion of the activities can be automated to some degree.

At the time of writing a number of 'automated' HTML to WML document conversion applications have been developed. Yet these applications focus on the almost direct transformation of the HTML code into WML code without regard to the layout and formatting, with the applications relying on the original documents being properly layed out and well formatted. Although not investigated, the current set of automated conversion tools may have trouble in representing information that is presented in tables, requiring templates to be produced by users.

Ideally a conversion tool would 'understand' a document and re-format it appropriately, but until this is possible semi-automated processes have to be used. It is believed that the solutions outline in this paper to some of the problems identified can be used to help in the drafting of a solution for the 'intelligent' automation of an HTML to WML conversion. At least they could be used as a set of guidelines for the conversion of tabulated data from the HTML to the WAP application base.

References


