WAP - Designing for Small User Interfaces

Abstract

Current and upcoming WAP-capable mobile phones introduce new user interfaces for which standard application design methods often fail. Automated translation of HTML to WML produces screen layouts and input mechanisms that are often unusable on mobile phones. In this paper we suggest a structured approach for designing WAP applications and in particular the development of the user interface. Firstly, we give a brief summary of the distinguishing features of phone interfaces and the mechanisms used to build applications for WAP devices. We then provide guidelines on how to port from Web to WAP and how to develop WAP applications. Finally, we introduce the SAP BW WAP application to highlight the design considerations discussed.

WAP-Devices

An increasing number of WAP devices are available today. In contrast to desktop computers these devices do not form a homogeneous platform in terms of UI and usage. This section gives a brief overview of our findings concerning different UIs and the support for WAP/WML.

User Interfaces

WAP devices have a variety of interfaces; output can be presented as text, formatted text, or graphics. Input is provided by touch screens and/or buttons. Displays are in general much smaller than those of PCs, however the size varies significantly. The Ericsson MC218 for example has a 640 x 240 pixel display whereas the Nokia 7110 has only 96 x 65 pixel; a ratio of about 24:1. In contrast, workstation screens differ by a ratio of only 6:1 (1600x1200 compared to 640x480). The large disparity between screen sizes makes it almost impossible to design an application that it fits all WAP devices [1].

The input mechanisms provided by WAP devices are also inconsistent. PDA-like devices offer point and click or a touch screen while most phones rely on a set of buttons. However, link navigation with a button interface is more difficult than with point and click. Scrolling up and down is supported very well on most devices (e.g. Navi-Roller on Nokia 7110) whereas the display width is often fixed and no horizontal scrolling is possible. Text input is difficult and time consuming on most phone-sized devices. They are better suited to, and users are more accustomed to inputting numbers. PDA-like devices on the other hand typically come with alpha-numerickeyboards.

To demonstrate these observations we undertook an experiment in which 19 arbitrarily chosen participants entered a block consisting of 100 digits or letters on a mobile phone (Nokia 7110). For each participant the time and error rate was recorded. The overall results are listed in Table 1.

	Numeric	**Textual**
Number of keystrokes	100	217
Average time (sec)	147.9091	376
Average time / key (sec)	1,479091	1,691244
Average Error rate	0,727273	3,5
Average Error rate / key	0, 727273 %	1,612903%

Table 1: Experimentalresults

The experiment shows that textual input is approximately twice as slow as numerical input (147,9/376) and that the error rate per key for textual input is about twice as high. The reason is not only that textual input requires roughly double the number of clicks required for numerical input. Even a single click takes longer when entering letters. This is surprising because once the correct button is found it is unlikely that the following click is executed more slowly. This means the process of searching for the right button is much more time consuming for textual input. The experiment indicates that numerical input should be the preferred input method on phone-sized devices.

Applications, WAP, WML and WMLS

PDA-like devices as well as phones offer a number of applications beyond phoning, such as messaging, calculator, address book, and games. For using these applications, and to some extent even for phoning, devices from different manufactures (or even of different models from one manufacturer) do not follow the same rules. With the introduction of the WAP standard, a manufacturer-independent platform for the development and distribution of applications for WAP phones and other WAP devices was achieved. Applications are written in WML and/or in WMLS and consist of one or more decks comprising one or more cards. The resource is provided on a server and can be downloaded by the user.

In our experiments we observed that most devices support only a subset of the specified markup tags and that these tags are also interpreted in different ways. This makes it impossible to predict the appearance of an application on the user's screen; the difference observed is significantly larger than the differences observed with applications in different standard Web browsers on desktop computers.

Designing and developing WAP-applications

Drawing on our experience of developing WAP applications (one of which is described in the next section), we devised a set of guidelines for the development of WAP applications. In the next paragraphs these guidelines are outlined as an informal procedure.

General Issues

First and foremost, it is important to identify the benefits of bringing an application to the WAP platform. Realizing these benefits must be the prime goal of application development. A potential benefit, for example, is universal access to information that is vital while on the move (e.g. contact addresses, travel information). The navigation space and user interaction with the application should be reflected in the structure of decks and cards in the implementation. Wherever possible (if cards do not rely on user input) a maximal navigation space should be provided within one deck to avoid reloading during navigation. If information is already provided on a Web site and should also be available on WAP the following approach can be used for porting: Define a typical path or more generally a graph and build a navigational structure for this path. This path should also include nodes where a query to the backend-system is necessary. Every path between two nodes is then represented by a deck consisting of multiple cards. Use the titles and headlines of Web pages along the path to create a content framework. Reduce text to a minimum and design information chunks that fit on the smallest devices you are designing for.

Input Design

Most current phones are best suited to inputting numbers rather than text. A direct point and click facility, familiar from PDAs and PCs, is typically not provided. Based on this the following recommendations for input design should be considered:

- use *numbers for input* wheneverpossible
- use *common abbreviations* like country codes (e.g. "D" for Germany)
- if letters are used, keep the input mechanisms in mind, e.g. *preferably the first letter on a button*
- *offer choices* (e.g. numbers, listboxes, radio-buttons, link-lists) or *default values* when applicable; even long link-lists prove viable because of scrolling
- *provide labels to hardware buttons* were possible
- *use standard conventions* on buttons (e.g. back)

The basic principle is to minimize the need for user input. Here customization of applications for the user provides one solution, e.g. a sorted listbox with the user's most frequently used items. Links between desktop interfaces and WAP devices can also be beneficial.

Output Design

Displays on mobile phones are in general rather small and provide only low resolution and color depth. Thus, reading large texts is rather difficult whereas displaying small chunks of information is much better. The use of graphics rarely provides additional value because of the low resolution and poor rendering. The following list draws attention to the most important issues:

- assess the *screen size and quality* of the target devices for text and graphics
- *reduce the output* by customizing to the users' needs [2]
- design *information chunks that are seen as a whole* on the screen, larger text blocks (more than 20 words!) within one card should be structured
- *vertical scrolling* is easy on most devices, horizontal is not!
- use *multiple cards in one deck* instead of very large cards or multiple decks

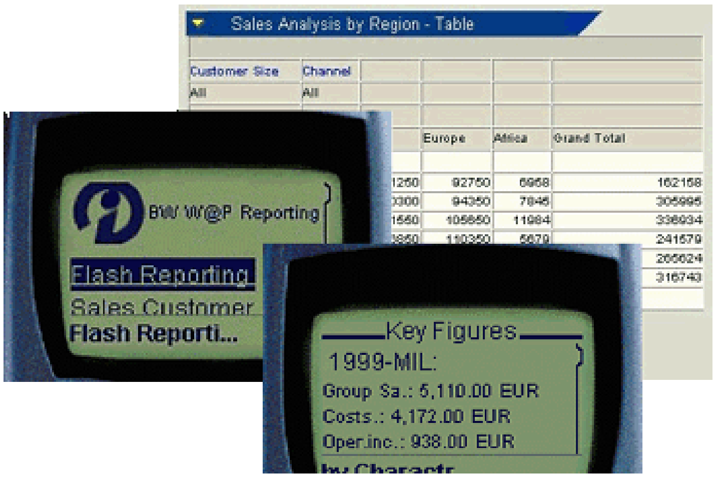
Given the wide range of devices and their capabilities, it is advisable to individually design input and output for certain device classes.

SAP BW WAP Reporting - A case study

SAP Business Information Warehouse (SAP BW) is the business intelligence hub of the mySAP.com offering. The personalized character of SAP BW's predefined Business Content is ideal for the individual user roles that mySAP.com supports. SAP BW supplies information in context, and in accordance with the user's tasks and decision-making needs. This includes multiple key performance indicators (KPIs), alerts, news and notes, as well as workgroup comments.

The main hurdle to presenting information to the user was the limited space available on a WAP device. Normally, the information is displayed using HTML based tables with full feature rendering capabilities offered by state-of-the-art web browsers. These includes color schemes, different font sizes and interactive tools like scrolling.

Current WAP devices do not offer most of these options and must therefore be supported differently. The amount of information to be displayed on a WAP device has to be much smaller than the amount displayed on a powerful desktop computer. Taking this into account, the application focuses on high level information that is displayed using the available features of WAP devices.



1999-MIL:
Group Sa.: 5,110.00 EUR
Costs.: 4,172.00 EUR
Oper.inc.: 938.00 EUR

Figure 1: SAP BW WAP Reporting

Conclusion

WAP applications require a new technique for user interface design. The primary design guideline is to minimize user interaction during input and output. Suggested techniques for input reduction include service pre-selection, provision of choices, input by navigation, and focusing on numerical input. Suggested techniques for output reduction include WAP specific presentation as well as user and context driven customization.

by Oliver Frick
SAP AG, Corporate Research
Pervasive Computing Group
oliver.frick@sap.com

Albrecht Schmidt
Telecooperation Office
University of Karlsruhe
albrecht@teco.edu

References

[1] Brewster, S., Cryer, P.: Maximising Screen-Space on Mobile Computing Devices, in Proc. CHI'99, 1999.

[2] Barret, R., Maglio, P., Kellem, C.: How To Personalize the Web, in Proc. CHI'97, 1997.