Proposing Usability Patterns for Mobile Multimodal Applications

Werner Kurschl, Wolfgang Gottesheim, Stefan Mitsch, Rene Prokop, Johannes Schönböck
Research Center Hagenberg
Upper Austria University of Applied Sciences
Softwarepark 11
A-4232 Hagenberg, AUSTRIA
{*wkurschl, wgottesh, smitsch, rprokop, jschoenb*}*@fh-hagenberg.at*

Abstract

Speech recognition has matured to the point that companies can seriously consider its use. We developed a distributed framework that enables multimodal user interfaces with speech recognition (dictation and command/control) on any type of mobile device. But do users already accept speech as additional input modality, and if so, which patterns help to cope with usability challenges in multimodal applications?

We found that existing guidelines for designing multimodal interfaces concentrate on high-level issues, but fail to provide applicable patterns and implementation guidelines. Usability patterns are an approved way to pass on expert knowledge on solving specific problems and enable developers to meet challenges in an efficient way. This paper discusses patterns to help with successfully developing multimodal interfaces for mobile devices supporting speech as additional input modality.

1 Introduction

Speech is a natural communication means for human users. However, natural communication does not only include speech, but also comprises gestures, facial expressions, and other non-verbal signs. In many situations these modalities complement each other; many tasks can be performed better with one modality than with another (e.g., speech is appropriate for entering text, whereas a PDA's stylus is the better choice for pointing at something). Thus, a combination of different modalities promises to provide a superior user experience over each of the single modalities. Multimodal applications allow the user to combine two or more input modalities to operate the application. But to design these interfaces expert knowledge that many developers lack is required.

How can this knowledge be passed on? Guidelines are a well-known way to pass on design knowledge and help designers, but applying them is prone to a number of problems [3, 8]; e.g., they are often too simplistic or too abstract, or they are difficult to interpret and apply, or the sheer number of guidelines available makes it difficult to choose the correct one. Patterns are more powerful than guidelines: They explicitly state when, how and why a solution is to be applied. They are extremely popular in software engineering (e.g., [5]), but while they gained a lot of interest in the area of user interface design (e.g., [1, 2, 12]) a proper set of patterns is still not available.

We implemented an e-mail and contact management application providing a multimodal interface that can be used on a mobile device (a smartphone or a PDA) and can be operated by speech, a keyboard and a pointing device (e.g., a stylus). During the implementation, we found that existing guidelines for designing multimodal interfaces concentrate on high-level issues, but fail to provide applicable patterns for mobile devices. In this paper we present a collection of patterns we identified after conducting a usability study.

2 Related Work

Graphical user interfaces have been well researched in the past, and many findings naturally apply to multimodal interfaces as well. For example, Nielsen [9] presents ten usability heuristics that are broad enough to apply to many user interfaces. Tidwell [10] and van Welie [11, 12] motivate and describe patterns for interface design, but these are targeted to graphical user interfaces and can therefore only provide starting points for multimodal interaction patterns.

As multimodal interfaces are often seen as a part in accessibility initiatives, various organizations try to help designing them. For example, the ETSI publishes "Multimodal interaction, communication and navigation guidelines" [4], but they employ a high-level perspective and thus mainly cover general accessibility issues. They are therefore only partially applicable to the scenario discussed in this paper as they lack concrete and easily applicable implications on multimodal interfaces on mobile devices.

The W3C Multimodal Interaction Activity [15] tries to "extend the Web" and publishes the W3C Multimodal Interaction Framework [16], proposing a general framework for multimodal interaction considering various markup languages. Furthermore, the Multimodal Interaction Requirements [14] cover a wide range of issues like input and output modalities or privacy considerations, but do not aim to help with designing and creating a multimodal system. While they take speech recognition into account, they do not provide help with specific speech-related issues.

Zajicek [17, 18] suggests eight patterns for speech systems for older adults. These patterns are focused on non-multimodal (voice only) systems and are therefore not applicable to the multimodal application domain.

3 Usability Patterns

Building a high-quality user interface requires expert knowledge on human-computer-interaction. Several sources provide know-how for building graphical user interfaces, such as tutorials and guidelines. Usability patterns are an approved way to pass on expert knowledge on solving specific problems. Most of those resources are not available for the development of speech-based and multimodal user interfaces. Moreover, hardly any developer is experienced in multimodal or verbal user interfaces especially in combination with mobile devices.

Therefore we defined basic usability patterns for human-computer-interaction (HCI) using multimodal interfaces supporting speech input on mobile devices. Our findings presented in this paper are based on a usability study we conducted using a fully functional prototype of a multimodal e-mail and contact application designed for mobile devices. Our primary goal was to find out main barriers that distract people using an application by speech and provide ways to avoid these potential pitfalls.

Our test subjects were familiar with computers, some of them were also using PDAs for mobile communications, but all were novice users to the application tested. No one was familiar with using a speech recognition system before.

The application provides a multimodal user interface and can be completely controlled by voice, a keyboard, a pointing device (e.g., a stylus), or a combination of those. Its graphical user interface is similar to Pocket Outlook (see figure 1) to minimize the effort of learning the application's features our usability

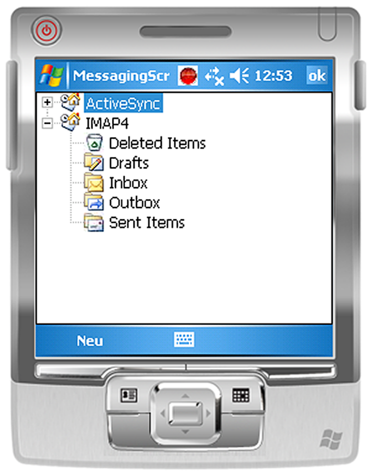

Figure 1: Example screen

subjects need. We built this application using the Gulliver framework [7] to achieve support for command-and-control and dictation functionality. Thus it is possible to create new mails and contacts by dictation instead of using a keyboard emulation or alike.

The patterns discussed here all apply to applications providing a multimodal interface on a mobile device, supporting speech as additional input modality. We recognize the value of a structured pattern language, but due to space constraints decided to choose an informal way as follows:

Feedback on each input modality handler's state We identified three relevant states users have to know about: whether the handler (e.g., a speech recognition engine) is (i) active (e.g., waiting for speech input) or (ii) it is disabled (e.g., because the microphone is turned off), or (iii) the handler speech engine was unable to recognize and process the last input. These states can be visualized by a simple icon which changes its color and shape depending on the status (e.g., green spot, red spot, or white cross on red background). An icon seems to suit the requirements best because it does not demand too much of the limited space on mobile devices and does not annoy the user as pop-up dialogs would do. Pop-up messages should only be used if additional information is shown and the user gets the opportunity to react on the messages (e.g., choosing from several possible results).

Feedback on the processing progress Since processing input implies a significant overhead, users may experience additional delays when using the application (e.g., by voice). Although these delays depend largely on network bandwidth and the processing power of the devices involved and thus can be minimized, they cannot be completely eliminated. In our tests, we encouraged the testers to state how they experienced the application's speed.

Our tests have shown that speed perception differs from case to case: Most users accepted some seconds of processing time before text input was recognized, while on the other hand delays in command recognition or in navigating between screens were noticed as intolerable. Immediate feedback (e.g., showing a wait cursor) remedies the problem somewhat.

Users demand immediate results from their actions as they know it from graphical-only applications: If they press a button they not only expect some action to happen, but they insist on immediate and visible results from their actions. The technical necessities of input processing (e.g., transmitting speech data over a network to a more powerful device) sometimes conflict with these requirements as the recognition process involves additional time. Therefore, user acceptance for command-and-control proved to be low if it imposes waiting times on the user, even if its usage might be more intuitive than other modalities.

Say-what-you-see Users cannot know all supported commands, especially when using the speech-enabled application for the first time. The most intuitive concept is to provide grammars that reflect the graphical user interface, allowing for a very simple and step-by-step way to operate the application. This approach is the only way to enable a fluent workflow for novice and expert users. Additionally, this method works well on mobile devices that do not offer enough space to show the commands explicitly on the screen.

This approach has side effects on the graphical user interface. To ensure acceptable recognition results, the displayed texts must be optimized for verbal input (i.e., abbreviations and similar labels should be avoided). Interface elements that are not clearly and unambiguously identifiable by their appearance, like items in a list, should offer a surrogate label (e.g., a number).

Provide verbal shortcuts More experienced users might feel hindered by the simple grammars imposed by the say-what-you-see concept. Besides commands derived from visible interface elements, more complex commands can act as ”verbal shortcuts” and simplify tasks. For example, adding a recipient to a new message might be performed by opening the address book and selecting the desired person, requiring to switch screens back and forth, or by simply saying ”add recipient <name>”.

Nevertheless, grammars cannot be indefinitely complex to be accepted: grammars which would encapsulate a whole process in a single command (i.e. ”send e-mail to <name> with subject <subject> and content <content>”) are technically challenging and too long to be used as they impose a high cognitive workload. This might result from the long lasting training with graphical user interfaces, which demand a stepwise process (set focus before entering data).

Unambiguous Format For entering structured data or data with special syntax the input controls should be tailored to the task at hand by limiting the input capabilities. This increases recognition accuracy and allows automating data formatting, which would be very cumbersome to do using speech. This pattern is well-known in graphical user interfaces and described in various pattern collections like [12]. Typically it is used to enter date and time values or currency values. In connection with speech input the pattern becomes even more important. For example when entering a phone number by speech the number must be represented in digits separated by special characters although the spoken command does not differ from entering digits that should be written as words.

Dealing with inaccurate user input Speech recognition is less exact than other input modalities are; hence misrecognized text might occur when using dictation, or unwanted or even harmful actions might be executed if a command is misrecognized.

Validating user input is by no means a new concept, but due to these additional factors applications that rely on speech input have to pay even more attention than other systems. Accidentally selected functions which might lead to irreversible (side) effects should be secured by confirmation dialogs (cf. pattern ”Shield” in [12]). Therefore, functions that cannot be undone or require a lot of system resources to undo should be secured by confirmation dialogs. But as confirmation dialogs pop up in each case–no matter if an action was invoked deliberately or by error–they are annoying and lower the user’s productivity if they occur too often.

For protecting small steps of user interaction, like entering single words in a textbox, an Undo/Redo mechanism is more suitable than confirmation dialogs. All actions and changes are performed immediately– even those which result from misrecognition of speech. The obvious advantage of this mechanism is that it does not conflict with the human computer interaction: the user can use the system fluently without fearing misrecognitions, because every unwanted action can be undone with minimal effort.

Misrecognized dictated text is seen as an annoyance but causes no irreversible effects. The application has to provide means to easily correct text, for example by allowing the user to select alternative recognition results, or by an undo mechanism.

4 Conclusion and Further Work

Most of the usability criteria available for graphical applications unsurprisingly also apply to speech-enabled or multimodal applications, but they need to be intensified. This is mainly due to the transience and ambiguity of speech, and to current speech recognition systems' recognition performance. We provide a collection of patterns that aim to deal with the challenges arising and help software developers to efficiently and effectively develop mobile multimodal applications.

Typical mobile applications (like our e-mail and contact application) can be used efficiently with a stylus for pointing and speech for entering text. But any multimodal application should strive to be usable with either of its modalities exclusively, as there can be situations where using a particular modality is less comfortable or not possible (e.g., the stylus while driving, speech in public). As multimodal systems demand a sophisticated software infrastructure, it is also necessary to improve software libraries, frameworks, and tools to some extent. The Gulliver framework [6,7] is a step towards meeting these development challenges.

The most important patterns are: (i) provide feedback about the application's state, let users (ii) say what they see, (iii) gracefully handle inaccurate input and allow for easy correction of mistakes (e.g., by providing undo functionality), and (iv) ask for confirmation. Our patterns are only applicable to multimodal applications supporting speech as additional input modality. Future work on these patterns might include improvements by making them applicable to a broader scope, e.g. by including the usage of speech as an output modality. We also aim to provide a structured representation according to a design pattern language (e.g., the language suggested in [13]).

References

[1] E. Bayle. Putting it All Together: Towards a Pattern Language for Interaction Design. *SIGCHI Bulletin*, 30(1):17–23, 1998.

[2] J. O. Borchers. A Pattern Approach to Interaction Design. In *Symposium on Designing Interactive Systems*, pages 369–378, 2000.

[3] A. Dix, G. Abowd, R. Beale, and J. Finlay. *Human-Computer Interaction*. Prentice Hall, 1998.

[4] ETSI. Human Factors (HF); Multimodal interaction, communication and navigation guidelines. http://docbox.etsi.org/EC_Files/EC_Files/eg_202191v010101p.pdf.

[5] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Boston, 1995.

[6] W. Kurschl, S. Mitsch, R. Prokop, and J. Schönböck. Development issues for speech-enabled mobile applications. In *Proceedings of SE 07 – The Conference on Software Engineering*, 2007.

[7] W. Kurschl, S. Mitsch, R. Prokop, and J. Schönböck. Gulliver—A Framework for Building Smart Speech-Based Applications. In *Proceedings of the 40th Hawaii International Conference on System Sciences (HICSS-40)*, Hawaii, USA, 2007. IEEE.

[8] M. Mahemoff and L. Johnston. Pattern languages for usability: An investigation of alternative approaches. In *APCHI 98 Proceedings*, pages 25–31, Los Alamitos, CA, USA, 1998. IEEE Computer Society.

[9] J. Nielsen. *Usability Engineering*. Morgan Kaufmann, San Francisco, USA, 1993.

[10] J. Tidwell. *Designing Interfaces: Patterns for Effective Interaction Design*. O'Reilly, 2005.

[11] M. van Welie. Interaction Design Patterns. http://www.welie.com/patterns/, 2007.

[12] M. van Welie and H. Trætteberg. Interaction patterns in user interfaces. In *Proceedings of 7th Pattern Languages of Programs Conference*, 2000.

[13] M. van Welie, G. van der Veer, and A. Eliens. Patterns as Tools for User Interface Design. In *Workshop on Tools for Working With Guidelines*, Biarritz, France, 2000.

[14] W3C. Multimodal Interaction Requirements. http://www.w3.org/TR/mmi-reqs/.

[15] W3C. W3C Multimodal Interaction Activity. http://www.w3.org/2002/mmi/.

[16] W3C. W3C Multimodal Interaction Framework. http://www.w3.org/TR/mmi-framework/.

[17] M. Zajicek. Patterns for speech dialogues for older adults. http://cms.brookes.ac.uk/computing/speech/index.php?id=61, 2003.

[18] M. Zajicek and W. Morrisey. Speech output for older visually impaired adults. In *Proceedings of IHM-HCI*, pages 503–513, 2001.