Evaluation of a Mobile Multimodal Application Design − Major Usability Criteria and Usability Test Results

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

Abstract

Mobile broadband internet access and powerful mobile devices make interesting and novel communication applications possible (e.g., recently emerging VoIP applications). Additionally, 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 usability challenges arise when developing multimodal applications?

This paper presents the results from usability tests that we conducted with a mobile multimodal e-mail and contact application. Based on the results we point out major usability criteria that need to be met in developing mobile multimodal applications.

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.

According to [1] multimodal applications are still rare in everyday use, though they would offer promising opportunities. This is primarily due to the wide adoption of well-known input devices like keyboard and mouse, which are even emulated by computing devices lacking those input capabilities (e.g., PDAs provide an on-screen keyboard).

With mobile broadband internet access and powerful mobile devices available, mobile business applications offer an added value. Especially field workers, who spend most of their time out of office and traveling between several sites, need access to up-to-date data. Although the devices are powerful enough to present high quality user interfaces, the handling of applications is still not satisfying. This is partly due to the emulated keyboard on PDAs which is hard to use and thus annoying. Therefore a lot of device manufacturers provide small keyboards. However, they are either too small for convenient typing or the device gets too large. Thus, the benefit of mobile applications is often lessened because of missing adequate input modalities. The means to this end is speech recognition, which has advanced to the point that companies can seriously consider its use.

To assess the feasibility of multimodal user interaction, we implemented an e-mail and contact application and conducted usability tests with persons that were familiar with e-mail and contact applications in general, but used our application for the first time. In the test we concentrated on spoken interaction to find main barriers that distract people from using applications by speech.

2. Related Work

In [2] and [3] Jokinen and Hurtig discuss evaluation results of a multimodal route navigation system. They compare user expectations from before the evaluation with the actual user experience during the tests. Their results show that users prefer multimodal systems over unimodal ones. However, the users' expectations vary with their perception of the system: they tend to expect fluent spoken communication from a primarily speech-

controlled system; but if speech is only expected to be a secondary modality, it provides additional value.

Nielsen in [4] presents ten usability heuristics that are broad enough to apply to many user interfaces. Some of these heuristics are also found in more specific works. Häkkilä (see [5]) derives design guidelines for context-aware mobile systems from extensive user studies. Some of the guidelines are general enough to also apply to multimodal applications, as we observed in our tests.

In [6] Turunen et al. describe evaluation results of multimodal applications built with their Jaspis architecture. They discovered that in human-to-computer communication shorter sentences, different words and sentence structures, and a terser style than in human-to-human communication were used. In our usability tests users showed similar behaviour: they did not have problems with short commands, but were unlikely to use complete sentences.

During the evaluation of the MATIS system (a form-filling application for querying train timetables, see [7]) the efficiency and effectiveness of interacting with the same system via different modalities (graphical only, speech only, and combined), as well as the user satisfaction were measured. Their results indicate that multimodal applications can improve interaction with speech-based applications. However, their users clearly preferred the graphical user interface over the other two because it was fastest and least error-prone. Thus, Sturm et al. suggest providing a backup input device (e.g., a keyboard) to let users handle speech recognition errors. Our test results showed that for some form-filling tasks, e.g., entering text when only a small-sized or emulated keyboard is present, multimodal interaction is more efficient and satisfying for users.

3. The E-Mail and Contact Application

PDAs are often used for managing contacts and reading e-mails, and hence such applications (like Pocket Outlook) are familiar to many users. Thus, we implemented an e-mail and contact application that provides a multimodal user interface (speech, keyboard, and stylus). Its graphical user interface is similar to Pocket Outlook to minimize the effort of learning the application's features our usability subjects need. However, it can be completely controlled by voice.

Figure 1 shows the inbox of the e-mail application. This screen can be completely controlled by voice, with a stylus, or with a combination of both. An e-mail can be selected by either saying the subject or the number of the e-mail, a new e-mail can be created by saying "new", and it is possible to sort the e-mails by saying for example "sort by date".

Figure 1: E-mail inbox

As described in [8] speech recognition can roughly be divided into constrained vocabulary (command-and-control) or unconstrained vocabulary (dictation) speech recognition. The e-mail and contact application builds upon the Gulliver framework (briefly introduced in the next section) and thus supports both dictation and command-and-control.

3.1. The Gulliver Framework

The Gulliver framework supports various types of mobile devices, from conventional cell phones to PDAs or even Laptops or Tablet PCs. Although these devices offer different capabilities, all of them demand for a flexible framework that hides speech recognition from the user interface code; with current speech engines' APIs they are tight-knit. As shown in Figure 2 the framework distributes speech recognition to several components to support both constrained and unconstrained speech recognition. While constrained speech recognition can be performed by a speech recognition engine hosted on the device itself PDAs are not powerful enough to recognize unconstrained speech locally. Therefore we have to transmit compressed voice data via VoIP to a remote server which does unconstrained speech recognition. For detailed information see [9].

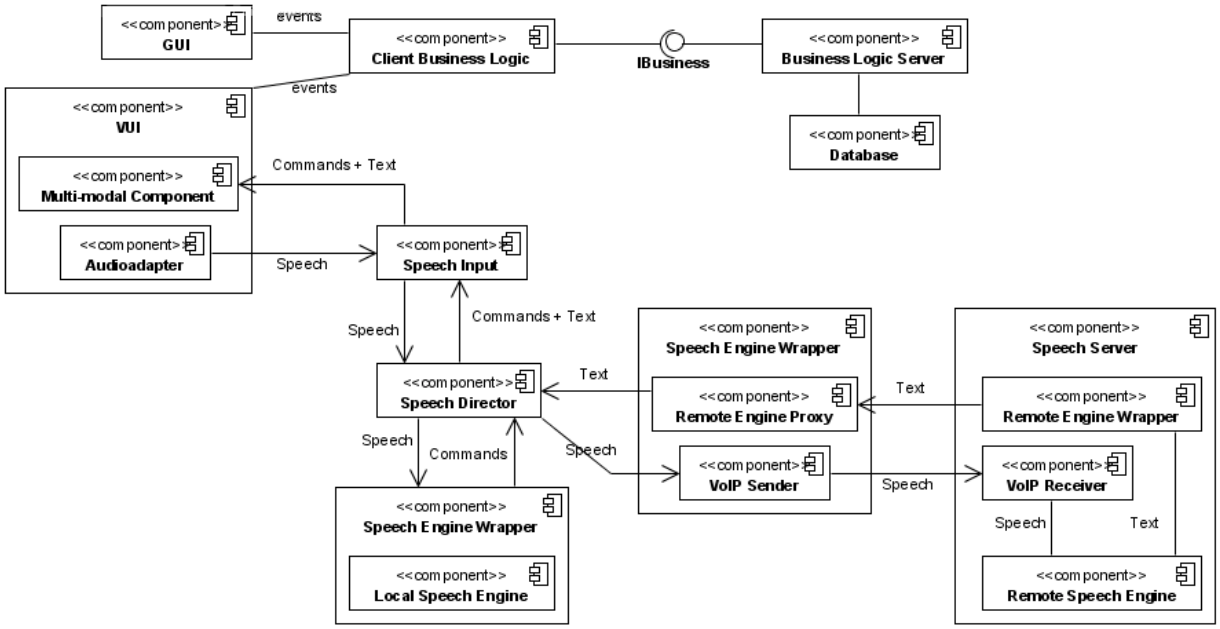

Figure 2: Framework architecture

3.2. User Interface Controls

Application developers should not need to deal with the internals of speech recognition; they want to efficiently develop speech-enabled applications. Thus, user interface components, which are capable of handling voice input, are needed. These multimodal components translate speech recognition results into events known from ordinary UI controls. The Gulliver framework contains speech-enabled controls like Button, ListBox or ComboBox. Each component defines its syntax (i.e. valid input values) in a so called grammar and provides a default grammar, which handles the basic functionality. A Button's default grammar for example is derived from the label of the button. Speaking the label then issues a button-clicked event, which can be handled by the application developer. For more complex tasks customized grammars can be assigned to a control.

The multimodal components can be integrated into the development environment and, thus, be used very conveniently. Currently, the controls are implemented for the .NET Compact Framework but the concepts are not restricted to a specific programming language.

4. Usability Study

The main goal of the test was to assess the application's usability when it was used for the first time (see 4.2, all test persons were novice users). We wanted to find out the main barriers that distract people from using an application by speech, and if these barriers differ from barriers in applications without speech recognition.

A minor goal of the test was to find alternative or more natural ways of solving tasks by speech. Therefore, the test persons were not given a detailed introduction how to use the application; they only received a basic introduction into the application's features. Visual clues are the basis for the voice input users try. We presume that the interpretation of these clues differ and therefore aim to provide flexible grammars.

4.1. Test Procedure

We chose to conduct our study according to Nielsen's approach of "discount usability engineering" described in [4]. A working prototype of the final application was used for the tests.

The tests were conducted in isolation; each test run was performed with a single test person and observed by a member of the development team. The test person was encouraged to express thoughts (both positive and negative impressions) in any situation aloud; these suggestions were logged. Test persons were advised to use speech, but they were allowed to use a stylus when they either became tired of using speech or when they did not find the solution to a task by speech.

A test run comprised the following steps:

1. Test goals were explained to the test person.

2. The application was introduced to the test person (features, online help, push-to-talk-button, etc.).
3. The speech recognition environment was set up, i.e., a user profile was created and the speech recognition system was trained.
4. The tasks were explained to the test person and executed one by one.
5. A guided follow-up interview was conducted.

The test itself consisted of the following tasks that a test person had to perform:

1. Create a new contact in the address book: provide at least the contact's first and last name.
2. Edit an existing contact: change at least the first name and add a note to the contact.
3. Delete an existing contact from the address book.
4. Send an e-mail message to a contact from the address book.
5. List all messages in the inbox and read one message in detail.
6. Reply to a message in the inbox.

4.2. Test Persons and Environment

All tests were conducted indoors in an office working environment (i.e., besides the test person and the observer, other people worked in the room, and persons entered and left the room). We did omit outdoor field tests, as [10] indicates that field testing does not necessarily lead to better test results.

We worked with a total of 12 test persons, all male except for one female, and all aged between 20 and 40. All of them use computers regularly (e.g. at work), and have or are in progress of obtaining an academic degree in the field of computer science. Some had prior experience with using a PDA for mobile communications, but all were novice users to the specific application tested. No one has used a desktop speech recognition system before, experience in this field was largely based on experiments with speech recognition integrated in mobile devices (e.g., voice dialing on a mobile phone).

4.3. Guided Follow-Up Interview

The follow-up interview consisted of 85 questions organized in six groups: Demographic questions, question concerning the overall application performance, speech recognition related questions, questions related to the contact management part of the application, questions related to the messaging part, and general closing questions. The answers are either free text or spread on a four-point scale.

4.4. Research Hypothesis

Based on previous research (see, e.g., [4]) and our own experience with the tested application we expected the following statements from the usability study.

1. Users are expected to have little or no problems interacting with interface elements that are "labeled" and can thus be addressed by saying this label.
2. Although users tend to limit their speech interaction to commands that can be derived from visible interface elements, alternatives that are more flexible and comfortable but not immediately visible on the screen are expected to be difficult to discover, but expected to be used after an initial "learning phase" as they are easier to use.
3. When using a multimodal application, we presume that users need additional information in order to successfully use the application (e.g., about the state of the recognition engine or about recognition results).
4. Despite the added benefit of multimodal operations users don't want to experience additional delays when using the application by voice.

5. Study Results

Since we decided to go with the "discount usability engineering" approach, statistical data analysis is not appropriate and the findings we report here are qualitative in nature.

Interaction with labeled interface components

We observed during the tests that the users' first try was to say the labels of the graphical controls interact with them. Our tests have therefore proven that "labeled" interface components are an intuitive way to select items or change the focus on a screen by voice. None of our test users had problems selecting buttons and text boxes as they usually only have a single label and therefore no ambiguous input is possible. On the other hand, selecting a certain item in a list like a contact or an e-mail imposes more difficulties as

ambiguities may arise. We chose to enumerate the items in a list and to provide appropriate grammars to select and handle list items by this index. This solution was easily adopted by the test users.

Consistency is crucial here: All elements with a label have to be accessible by voice commands; otherwise the users become frustrated quickly. Once users had adopted this behavior they were unwilling to use more natural alternatives to the often curt labels.

Usage of more complex grammars

Besides commands derived from visible interface elements, some screens in the application prototype have grammars that provide additional functionality (e.g. adding a new recipient to an e-mail) but do not indicate them by visible clues. Although no one discovered them on their own, some of the users continued to use them after discovering them in the help. Several stated that they have thought of trying something similar, but had refused to do so because they didn't think it would work.

Nevertheless, grammars can't be indefinitely complex to be accepted: grammars which would encapsulate a whole process in a single command (i.e. "send e-mail to … with subject … and content …"), haven't been expected at all. Users deemed them to be hard to memorize and difficult to use. This might result from the long lasting training with graphical user interfaces, which demand a stepwise process (set focus before entering data).

Feedback

The application prototype uses different techniques to inform the user about the state of the voice recognition component: an icon shows whether the recognition engine is currently listening to voice input, whereas a notification window pops up when the input could not be interpreted. This solution was considered suboptimal by most users. They judged the pop-up windows as too annoying, since they tend to appear quite often and hide a significant portion of the screen. Furthermore, the pop-up doesn't provide additional information about possible and legitimate input. Overall, the implemented approach proved to be frustrating as the information given was not helpful and actually hindered the user. Most of the test users considered an icon as a sufficient, unobtrusive and therefore better way of informing about the application's state.

Speed considerations

Since voice processing implies a significant overhead, users may experience additional delays when using the application by voice. Although these delays depend largely on network bandwidth and the processing power of the devices used 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 dictation was recognized, while on the other hand delays in grammar recognition or in navigating between screens by voice 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 voice processing 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.

6. Usability Patterns

Building a high-quality user interface requires expert knowledge on human-computer-interaction. There are several sources of know-how for graphical user interfaces: a significant part results from the fact that every developer has long lasting experience with graphical user interfaces at least from the user's point of view. Moreover, various resources offer tutorials and guidelines. An approved way to pass on expert knowledge on specific problems to inexperienced developers is the usage of usability patterns.

For the development of speech-based and multimodal user interfaces most of those resources are not available. Moreover, hardly any developer is experienced in multimodal or verbal user interfaces especially in combination with mobile devices. The low number of existing speech-based applications leads to little opportunity to get in touch with this kind of user interface even from the user's point of view.

Therefore we defined some basic usability patterns for human-computer-interaction (HCI) using multimodal interfaces on mobile devices. The most important ones are described in an informal way as follows:

Feedback on the speech recognition progress

The user needs to know the speech recognition engine's progress. When a user operates a graphical

user interface component an immediate action occurs. The same is expected for voice input but this is by far more complex. Thus the user has to be informed if the recognition engine is listening to user input and if it did recognize the input.

We identified three states, which are interesting for the user: (i) speech recognition engine is listening, (ii) microphone is turned off, and (iii) speech engine was unable to recognize the last utterance. These states are visualized by a simple task bar icon which changes its color and shape depending on the status (e.g., green spot, red spot, or white cross on red background). The 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 (like choosing an alternative recognition result).

Say-what-you-see

Due to the fact that the user can't know all supported commands, especially when using the speech-enabled application for the first time, the most intuitive approach is to provide grammars that reflect the graphical user interface. Additionally, this approach works well on mobile devices that do not offer enough space to show the commands 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).

The say-what-you-see grammars provide a very simple and step-by-step way to operate the user interface. Similar to graphical user interfaces they demand to set the focus before data can be entered. By that the way of interacting with the user interface is familiar immediately.

Although users might switch to use more complex but more convenient grammars as they become more experienced, the say-what-you-see approach is the only way to enable a fluent workflow for novice users.

Unambiguous Format

For entering structured data or data with special syntax the input controls should narrow down the task by limiting the input capabilities. The limited speech recognition space increases the recognition accuracy and automates data formatting, which would be very cumbersome to do using speech. This pattern is well-known from graphical user interfaces and described in various pattern collections like [11]. 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 doesn't differ from entering digits that should be written as words.

Confirmation Dialog

Accidentally selected functions, which might lead to irreversible (side) effects, should be secured by confirmation dialogs (cf. pattern "Shield" [11]). Based on the fact that speech recognition is less exact than other input modalities wrong actions might be executed if a command is misrecognized. Therefore, functions like for example sending or deleting an e-mail should be secured. This pattern should be used in connection with the Undo/Redo pattern, which suites better for small processing steps. But functions that can not be undone or which would require a lot of system resources to undo should be secured by confirmation dialogs.

Undo/Redo Mechanism

Due to the fact that 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 would be more suitable. All actions and changes are performed immediately – even those which result from misrecognition of speech. The obvious advantage of this mechanism is that the mechanism does not conflict with the human computer interaction. The user can use the computer fluently while not being afraid of misrecognitions, because every unwanted action can be undone with minimal effort.

7. Conclusion and Further Work

Most of the usability criteria 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 (recall that we have to compress speech for transmission to the speech

recognition server). Thus, speech-enabled applications need to be implemented in a conversational style. The most important patterns are: (i) provide feedback about the application's state, let users (ii) say what they see and (iii) easily correct mistakes (undo), and (iv) ask for confirmation.

The usability tests have shown that users are willing to speak to the application and that they can solve all tasks by speech. This is important for applications that need to be exclusively manageable by speech because the hand and eye distraction is dangerous or disadvantageous (e.g., applications in cars).

Typical mobile applications (like our e-mail and contact application) are used best with the 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).

In our further work we want to investigate additional applications (database search, media booking) and alternative interaction components.

Acknowledgment

This research was supported by the Austrian Research Promotion Agency under the FHplus program, the Austrian Broadcasting Corporation (ORF), and Microsoft. Any opinions, findings, and conclusions or recommendations in this paper are those of the authors and do not necessarily represent the views of the research sponsors.

References

[1] Turunen, M., "Jaspis – A Spoken Dialogue Architecture and its Applications", Academic Dissertation (Tampere, Finland, 2004).

[2] Jokinen, K. & Hurtig, T., "User Expectations and Real Experience on a Multimodal Interactive System", in Proceedings of Interspeech 2006 (Pittsburgh, USA, September 2006).

[3] Hurtig, T., "A Mobile Multimodal Dialogue System for Public Transportation Navigation Evaluated", in Proceedings of 8th ACM Conf. on Human-Computer Interaction with Mobile Devices and Services, ACM Press, New York, USA, 2006, pp. 251-254.

[4] Nielsen, J., "Usability Engineering", Morgan Kaufmann, San Francisco, USA, 1993.

[5] Häkkilä, J., "Usability with Context-Aware Mobile Applications – Case Studies and Design Guidelines", Academic Dissertation (Oulu, Finland, 2006).

[6] Turunen, M., Hakulinen, J., Rähä, K.-J., Salonen, E.-P., Kainulainen, A., Prusi, P., "An Architecture and Applications for Speech-Based Accessibility Systems", in *IBM Systems Journal*, Vol. 44, No. 3, 2005.

[7] Sturm, J., Bakx, I., Cranen, B., Terken, J., Wang, F., "Usability Evaluation of a Dutch Multimodal System for Train Timetable Information", in Proceedings of 3rd Internation Conference on Language Resources and Evaluation (Las Palmas, Spain, May 2002).

[8] Kurschl, W., Mitsch, S., Prokop, R., Schönböck, J., "Development Issues for Speech-Enabled Mobile Applications", to be published in Proceedings of SE 07 – the Conference on Software Engineering (Hamburg, Germany, March 2007).

[9] Kurschl, W., Mitsch, S., Prokop, R., Schönböck, J., "Gulliver – A Framework for Building Smart Speech-Based Applications", in Proceedings of 40th Hawaii International Conference on System Sciences HICSS-40 (Big Island of Hawaii, USA, January 2007), IEEE Computer Society Press, 2007.

[10] Kaikkonen, A., Kallio, T., Kekäläinen, A., Kankainen, A. & Cankar, M., "Usability Testing of Mobile Applications: A Comparison between Laboraty and Field Testing", *Journal of Usability Studies*, Issue 1, Vol. 1, Usability Professionals' Association, Bloomingdale, USA, November 2005, pp. 4-16.

[11] van Welie, M., Trætteberg, H., "Interaction Patterns in User Interfaces", in Proceedings of 7th Pattern Languages of Programs Conference, 13-16 August 2000, Allerton Park Monticello, Illinois, USA.