CTC WP4

Usability Guidelines for Use Case Applications

Task 4.1, MS4, April 30, 2010

Version 1.5

Monday, April 26, 2010
@DFKI

Contacts:

Dr. Daniel Sonntag
Daniel.Sonntag@dfki.de

Phone: +49 681 857 75-5254
Fax: +49 681 857 75-5020

Colette Weihrauch
Colette.Weihrauch@dfki.de
Oliver Jacobs
Oliver.Jacobs@dfki.de
Daniel Porta
Daniel.Porta@dfki.de

Supported by

Federal Ministry
of Economics
and Technology

This research has been supported by the THESEUS Programme funded by the German Federal Ministry of Economics and Technology (01MQ07016). Thanks go out to Robert Nesselrath, Yajing Zang, Matthieu Deru, Simon Bergweiler, Gerhard Sonnenberg, Norbert Reithinger, Gerd Herzog, Alassane Ndiaye, Tilman Becker, Norbert Pfleger, Alexander Pfalzgraf, Jan Schehl and Jochen Steigner for the implementation and evaluation of the dialogue infrastructure.

Contents

Abstract

Usability Guidelines for Use Case Applications serves as an introduction to the general topic of usability, i.e., how user-friendly and efficient a THESEUS prototype is. In these guidelines, we emphasize the importance of usability testing, particularly during the development of a given THESEUS prototype. We discuss the many advantages of testing prototypes and products in terms of costs, product quality, and customer satisfaction. Usability testing can improve development productivity through more efficient design and fewer code revisions. It can help to eliminate over-design by emphasizing the functionality required to meet the needs of real users. Design problems can be detected earlier in the development process, saving both time and money. In these *Guidelines* we provide a brief overview of testing options, ranging from a cognitive walkthrough to interviews to eye tracking. Different techniques are used at different stages of a product's development. While many techniques can be applied, no single technique alone can ensure the usability of prototypes. Usability is a process with iterative steps, meaning the cycle is repeated but in a cumulative fashion, similar to software development.

In order to test, a prototype must be available and we devote some time in the *Guidelines* to an overview of different tools and ways to build the necessary prototypes. We also describe some options such as paper prototyping, prototypes from Visio, PowerPoint, HTML, Flash and others, and working prototypes (Java, C++, etc.) before addressing the actual tests. Before any testing is conducted, the purpose of the test should be clarified. This will have considerable impact on the kind of testing to be done. A test plan should also be written before the start of the test which considers several different aspects including, for instance, the duration of the test, where it will take place, or who the experimenter will be. A pilot test is also recommended to avoid misunderstandings and other problems during the actual test. In this context, the *Guidelines* also discuss other important aspects such as budget, room set-up, time, and limitations of the experimenter and test subjects themselves.

To provide an overview of some of the projects THESEUS is concerned with in the context of usability, we supply explicit recommendations that result in proposed scenarios for use cases in the *Guidelines*. The THESEUS program consists of six use cases: ALEXANDRIA, CONTENTUS, MEDICO, ORDO, PROCESSUS, and TEXO. In order to come up with the different testing scenarios, each of which has specific design and testing recommendations, we first extracted some substantial information from the different use cases in different user settings: we discerned between those who will use the system, where they will use the system, and what they will do with the system. After considering the results, we determined that the THESEUS program works with seven different scenarios. We provide a decision tree that leads to specific recommendations for designing and testing with prototypes for each of the different scenarios and user settings. General recommendations concerning various input methods, the design, and the testing itself have also been included in the *Guidelines*. Following that, we emphasize what we find important for the design and testing of each of the seven testing scenarios. We address, for instance, the appropriate input method (keyboard, mouse, speech, etc.), according to the type of test subject (e.g., administrator or mobile user), or also which prototype could be used for the usability test.

We will also challenge the usability of traditional usability guidelines. Oftentimes, guideline descriptions and explanations are unsatisfactory, remaining vague and ambiguous in explanation The *Guidelines* close with an extensive list of recommended further information sources.

Executive Summary

The goal of this document is to offer some guidelines for designing a usable interface and conducting usability studies. The following sections will provide you with an overview of engineering methods, standards, tools, and ways to build prototypes.

In particular, we present methods for usability evaluation and give an overview of needed resources, especially in the context of the THESEUS use cases. We tried to specify the adequate usability testing scenarios for the project and give recommendations for designing and testing.

In the last section, you will find recommendations for further reading and more detailed information on specific usability topics.

Conventions used in this document

Framed text outlines the most important concepts and provides a short summary of the respective paragraph.

⚠ This sign indicates important information.

1 Why Usability Testing?

"The ultimate metric that I would like to propose for user friendliness is quite simple: if this system was a person, how long would it take before you punched it in the nose?" –Tom Carey

1.1 What is Usability?

Usability is quite simple to define—it means that people can use a product easily and efficiently to accomplish their tasks. Products that are usable enable workers to concentrate on their tasks and do real work, rather than paying attention to the tools they use to perform their tasks.

A usable product:

- Is easy to learn;
- Is efficient to use;
- Provides quick recovery from errors;
- Is easy to remember;
- Is enjoyable to use;
- Is visually pleasing.

Usability applies to every aspect of a product with which a person interacts (hardware, software, menus, icons, messages, documentation, training, and on-line help). Every design and development decision made throughout the product cycle has an impact on that product's usability. As customers depend more and more on software to get their jobs done and develop into more critical consumers, usability can be the critical factor that ensures that products will be successful and used.

1.2 Benefits of Usability

Usability engineering provides important benefits in terms of cost, product quality, and customer satisfaction. It can improve development productivity with a more efficient design and fewer code revisions. It can help to eliminate over-design by emphasizing the functionality required to meet the needs of real users. Design problems can be detected earlier in the development process, saving both time and money. It can save further costs through reduced support costs, lower training requirements, and greater user productivity. A usable product means a greater number of satisfied customers and a better reputation for the product and for the organization that developed it. Today, the application of usability engineering methods results in a competitive advantage in the development of a product.

2 Usability Engineering Methods

Usability engineering involves various techniques that can provide important information about how customers work with your product. Some of these usability techniques include:

- **User and task observations**—observing users at their jobs, identifying their typical work tasks and procedures, analyzing their work processes, and understanding people in the context of their work;
- **Interviews, focus groups, and questionnaires**—meeting with users, finding out about their preferences, experiences, and needs;
- **Benchmarking and competitive analysis**—evaluating the usability of similar products in the marketplace;
- **Participatory design**—participating in the design of the product and bringing the user's perspective into the early stages of development;
- **Paper prototyping**—including users early in the development process before coding begins with prototypes prepared on paper;
- **Creation of guidelines**—helping to assure consistency in design by developing standards and guidelines;
- **Heuristic evaluations**—evaluating software against accepted usability principles and making recommendations to enhance usability;
- **Usability testing**—observing users performing real tasks with the application, recording what they do, analyzing the results, and recommending appropriate changes.

Different techniques are used at different stages of a product's development. For example, as processes are being engineered and requirements developed, observations and interviews may be the techniques of choice. Later in the development cycle, as the look and feel of a product is being designed, benchmarking, prototyping, and participatory design may be useful techniques. Once a design has been determined, usability testing may be used more appropriately. Usability testing can even be done on a paper prototype, long before a single line of code has been written!

There are many techniques that can be applied to ensure the usability of products, which no single technique alone can ensure, however. Usability is a process with iterative steps, meaning the cycle is repeated but in a cumulative fashion, similar to software development. The usability process works best if it is done in partnership with product development (see Figure 1).

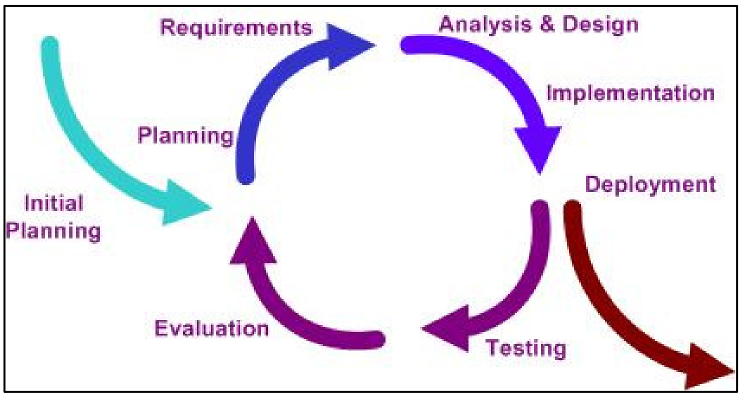


Figure 1: Usability Cycle

In the initial planning you will need to know the requirements of the system. This means the users of the system and the goals they will achieve must be defined. Analyzing the gathered information should lead into the first design phase of the user interface. In the early phases of the implementation you will use a prototype for testing and evaluation. These results should be used to refine the requirements and the design. After a number of cycles, the interface prototype will be implemented into a running system. Now, the system should be used to conduct further testing cycles until it is ready to be deployed. If the system is in use, user feedback and additional studies can help to improve functionality and usability of the system.

Table 1 presents the most important usability engineering methods. A detailed description of each individual method is given later in this section (the subsection is listed under the respective method in the table). This overview tells you when a certain method is appropriate and how many users are approximately needed. In addition, each method's main pro and con is listed.

Method Name	Life Cycle Stage	Users Needed	Main Advantage	Main Disadvantage
Heuristic Evaluation (2.1)	Early Design	None	Finds individual usability problems. Can address expert user issues.	Does not involve real users, so does not find "surprises" relating to their needs.
Thinking aloud (2.4)	Iterative design, formative evaluation	3 – 5	Pinpoints users' misconceptions. Inexpensive test.	Unnatural for users. Hard for expert user to verbalize.
Observation (2.5)	Task analysis, follow-up studies	3 or more	Ecological validity; reveals users' real tasks. Suggests functions and features.	Appointments hard to set up. No experimenter control.
Questionnaires (2.6)	Task analysis, follow-up studies	At least 30	Finds subjective user preferences. Easy to repeat.	Pilot work needed (to prevent misunder-standings).
Interviews (2.6)	Task analysis	5	Flexible, in-depth attitude and experience probing.	Time consuming. Hard to analyze and compare.
Focus groups (2.7)	Task analysis, user involvement	6 – 9 per group	Spontaneous reactions and group dynamics.	Hard to analyze. Low validity.
Performance measures (2.8)	Competitive analysis	At least 10	Results in hard numbers, easy to compare.	Does not find individual usability problems.
Logging actual use (2.9)	Final testing, follow-up studies	At least 20	Finds highly used (or unused) features. Can run continuously.	Analysis programs needed for huge mass of data. Violation of users' privacy.
User feedback (2.10)	Follow-up studies	Hundreds	Tracks changes in user requirements and views.	Special organization needed to handle replies.

Table 1: Summary of Usability Methods (Nielsen 1993)

We will now describe several methods in detail. As shown in the table above, some methods will be more effective in particular life cycle stages of the development than others.

2.1 Heuristic Evaluation

Heuristic evaluation is done by **looking at an interface** and trying to come up with an opinion about **what is good and bad** about it. The heuristics in this section can be tested **without any user**. This can be done according to certain rules, listed in typical guidelines, or based on one's own intuition and common sense. The goal of heuristic evaluation is to determine the usability problems in a user interface design so that they can be attended to as a part of an iterative design process. Heuristic evaluation involves having a small set of evaluators examine the interface and judge its compliance with recognized usability principles, for example, is important information emphasized in headings with the use of typographic features? The result of the heuristic evaluation method is a list of usability problems in the interface, annotated with references to those usability principles the design violated in each case. This is all based on the opinion of the evaluator.

2.1.1 Usability Heuristics

We will now present some basic characteristics of usable interfaces based on Jakob Nielsen's book *Usability Engineering*. The issues that are described in the following paragraphs are not hard facts and the usefulness depends on the given situation. Hence, discussion within the developing team is always advisable. The advantage of these characteristics is that they can be evaluated without any user. This evaluation can and should be done in the beginning of the development phase to avoid extensive correction and redevelopment work later.

Simple and Natural Dialogue

Ideally, the exact amount of information the user needs—and no more—should be presented exactly when and where it is needed. Less is more. Extraneous information not only risks confusing a novice user, but also slows down an expert user. Screen layouts should follow the gestalt rules for human perception to increase the user's understanding of relationships between the dialogue elements. These rules say that things are mentally grouped if they are close together, enclosed by lines or boxes, if they move or change together, or look alike with respect to shape, color, size, or typography.

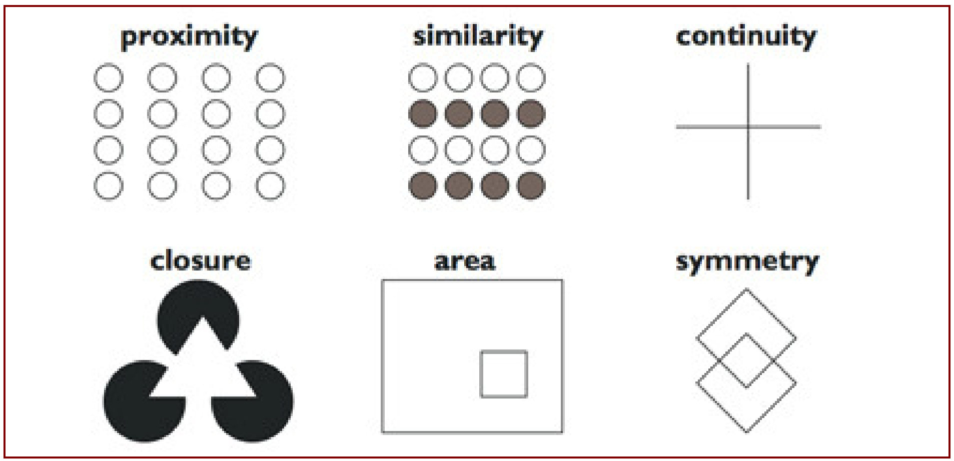


Figure 2: Gestalt Principles

Gestalt is a psychological term that means "unified whole." It refers to theories of visual perception developed by German psychologists in the 1920s. These theories attempt to describe how people tend to organize visual elements into groups or unified wholes when certain principles are applied. These principles are:

- **Proximity**, which occurs when elements are placed close together. They tend to be perceived as a group.
- **Similarity**, which occurs when objects look similar to one another. People often perceive them as a group or pattern.
- **Continuity**, which occurs when the eye is compelled to move through one object and continue to another object.
- **Closure**, which occurs when an object is incomplete or a space is not completely enclosed. If enough of the shape is indicated, people perceive the whole by filling in the missing information.
- Dividing and depth-creating **areas** are often part of what is called the figure-ground phenomenon. The phenomenon captures the idea that in perceiving a visual field, some objects take a prominent role (the figures) in the foreground while others recede into the background (the ground). The visual field is thus divided into basic parts, with some figures overlapping others.
- The law of **symmetry** captures the idea that when we perceive objects we tend to perceive them as symmetrical shapes that form around their centre. Most objects can be divided in two more or less symmetrical halves and when we see two unconnected elements that are symmetrical, for example, we unconsciously integrate them into one coherent object (or percept). The more alike objects are, they more they tend to be grouped.

Principles of graphic design can help users to prioritize their attention to a screen by making the most important dialogue elements stand out. With respect to the use of color in screen designs, the three most important guidelines are:

1. **Don't over-do it**. An interface should not look like an angry fruit salad of wildly contrasting, highly saturated colors.
2. **Make sure that the interface can be used without color**. Many people are colorblind, so any color-coded information should be supplemented by redundant cues so that the screens can be interpreted even if someone can not differentiate between the colors.

3. **Try to use colors only to categorize, differentiate, and highlight**, and not to give information, especially quantitative information.

Speak the Language of the User

As a part of user-centered design, the terminology in user interfaces should be based on the users' language and not on system-oriented terms. Dialogues should be in the users' native language as much as possible and not in a foreign language. Speaking the users' language also involves viewing interactions from their perspective. User interface metaphors are one possibility to creating an analogy between the interface and some reference system known to the user in the real world. This can reduce learning time and error rates. Care should be taken to present the metaphor as a simplified model of a more detailed conceptual model of the system and not as a direct representation of the system.

Minimize User Memory Load

Computers are capable of remembering things very precisely, so they should take over the burden of memory from the user as much as possible. The computer can therefore display dialogue elements to the users and allow them to edit these elements or to choose from computer-generated items. It is much easier for the user to modify information displayed by the computer than to have to generate all of the desired result from scratch. Whenever users are asked to provide input, the system should describe the required format and, if possible, provide an example.

Consistency

Consistency is one of the most basic usability principles. If users know that the same command or the **same action** will always have the **same effect**, they will feel more confident in using the system. The same information should be presented in the same location on all screens and in all dialogue boxes and it should be formatted in the same way to facilitate recognition. Consistency is not just a question of screen design, but includes considerations of the task and functionality of the system.

Feedback

The system should continuously inform the user about what it is doing and how it is interpreting the user's input. System feedback should not be expressed in abstract and general terms but should restate and rephrase the user's input to indicate what is being done with it. For example, it is a good idea to give a warning message in case the user is about to perform an irreversible action, such as deleting a file.

Response Time

Feedback becomes especially important in case the system has long response times for certain operations. The basic guidelines regarding response times have remained roughly the same for many years:

0.1 second is about the limit for having the user feel that the system is reacting **instantaneously**, meaning that no special feedback is necessary except to display the result.
1.0 second is about the limit for the user's flow of thought to remain **uninterrupted**, even though the user will notice the delay. Normally, no special feedback is necessary during delays of more than 0.1 but less than

1.0 second, but the user does loose the feeling of operating directly on the data.
10 seconds is about the **limit** for keeping the user's **attention** focused on the dialogue. For longer delays, users will want to perform other tasks while waiting for the computer to finish, so they should be given feedback indicating when the computer expects to be done. Feedback during the delay is especially important if the response time is likely to be highly variable, since users will then not know what to expect.

Normally, response times should be as fast as possible, but it is also possible for the computer to react so fast that the user cannot keep up with the feedback. For example, a scrolling list may move so fast that the user cannot stop it in time for the desired element to remain within the available window.

System Failure

Informative feedback should also be given in case of a system failure. Many systems are not designed to do this and therefore simply stop responding to the user when they go down. Unfortunately, no feedback is almost the worst possible feedback since it leaves the user guessing what is wrong. The system should be capable of giving users useful information about the likely cause and the location of any system failure.

Clearly Marked Exits

Users do not like to feel trapped by the computer. In order to increase the user's feeling of being in control of the dialogue, the system should offer her clearly marked exits out of as many situations as possible. For example, all dialogue boxes and system states should have a **cancel button** or other facility to bring the user back to the previous state. In many cases, exits can be provided in the form of an **undo facility** that reverts to the previous system state. As mentioned above, system response time should be as fast as possible. In cases where the computer cannot finish its processing within the 10-second attention holding limit, it should always be possible for the user to interrupt the computer and cancel the operation. The various exit and undo mechanisms should be made visible in the interface and should not depend on the user's ability to remember some special code or obscure combination of keys.

Shortcuts

Even though it should be possible to operate a user interface with the knowledge of just a few general rules, it should also be possible for the experienced user to perform frequently-used operations especially quickly by using dialogue shortcuts. Users should be allowed to jump directly to the desired location in large information spaces, such as a file or menu hierarchy. Users should also be able to reuse their interaction history. It is common practice for a user to accept system-provided shortcuts than specifying the shortcut themselves.

Good Error Messages

Error situations are critical for usability for two reasons. First, by definition they represent situations where the user is in trouble and will potentially be unable to use the system to achieve the desired goal. Second, they present opportunities for helping the user understand the system better since the user is usually motivated to pay some attention to the content of error messages, and since the computer will often have some knowledge of what the problem is.

Error messages should basically follow four simple rules:

- They should be phrased in **clear language** and avoid obscure codes.
- They should be **precise** rather than vague or general.
- They should constructively **help the user** to solve the problem.
- Error messages should be **polite** and should not intimidate the user or put the blame explicitly on the user.

In addition to having good error messages, systems should also provide good error recovery. For example, users should be allowed to undo the effect of erroneous commands, and they should be able to edit and reissue previous commands without having to enter them from scratch.

Instead of putting all useful information into all messages, it is possible to use shorter messages that will be faster to read as long as the user is given easy access to a more elaborate message. Ideally, it should be possible to steadily work oneself further along into a set of messages until the error has been identified.

Prevent Errors

Avoiding the error situation altogether would be even better than having good error messages. There are many situations that are known to be error-prone, and systems can often be designed to avoid putting the user in such situations in the first place. For example, every time the user is asked to spell out something, there is a risk of spelling errors, so selecting a file name from a menu rather than typing it in is a simple way to redesign a system to eliminate an entire category of errors.

In different modes, the system will interpret the user’s input in different ways. The user can be confused, therefore, **avoid modes!**

Modes, though generally avoided nowadays, are a frequent source of user errors and frustration and should be avoided if possible. The classic example of modes comes from early text editors, which had separate modes for inserting and editing text. When the user was in the insert mode, all keyboard input was interpreted as text to go into the file, and when the user was in the edit mode, all keyboard input was interpreted as commands. Modes basically partition the possible user actions with the result that not all actions are available all the time. This can be frustrating. If modes cannot be avoided completely, many mode errors can at least be prevented if the modes in the interface are immediately recognizable. System feedback should be sufficiently varied to provide additional differentiation between modes.

Help and Documentation

Preferably, a system should be so easy to use that no further help or documentation is needed to supplement the user interface itself. However, this goal cannot always be met. But, “it is all explained in the users’ manual” should never be the system designer’s excuse when users complain that an interface is too difficult for them to deal with. The fundamental truth about documentation is that most users do not read manuals. Users prefer spending their time on activities that make them feel productive, so they tend to jump the gun and start using the system without having read all the instructions. Normally, when users do want to read the manual then it is because they need immediate help and are in panic. Online information has the potential to provide users with the precise, required information faster than a paper book thanks to features like hypertext, good search facilities, and context sensitivity.

2.2 Requirement Analysis

In systems engineering and software engineering, requirements analysis encompasses those tasks that go towards **determining the needs** or conditions which must be met for a new or altered product. Possibly conflicting requirements of the various stakeholders, such as beneficiaries or users, should be taken into account.

2.3 Cognitive Walkthrough

A cognitive walkthrough starts with a task analysis that specifies the sequence of steps or actions a user requires to accomplish a task, and the system's responses to those actions. The designers and developers of the software then walk through the steps as a group in dialogue, asking themselves a set of questions at each step. Data is gathered during the walkthrough, and afterwards, a report of potential issues or problems is compiled. Finally, the software is redesigned to address the issues identified.

2.4 Think Aloud

Think aloud protocols involve users thinking aloud as they are performing a set of specified tasks. The participants are asked to **say whatever they are looking at, thinking, doing, and feeling**, as they go about their task. This enables observers to see first-hand the process of task completion (rather than only its final product). Observers at such a test are asked to objectively take notes of everything that users say, without attempting to interpret their actions and words. Test sessions are often audio recorded and video taped so that developers can go back and refer to what participants did, and how they reacted. The purpose of this method is to make explicit what is implicitly present in subjects who are able to perform a specific task. The experimenter sometimes has to encourage the test user; most people are not accustomed to saying what they are thinking when using a device, especially in the presence of others.

2.5 Observation of the User

Simply visiting the users to **observe them work** is an extremely important usability method with benefits both for task analysis and for the collection of information about the true field usability of installed systems. Observation is really the simplest of all usability methods since it involves visiting one or more users and then doing as little as possible in order not to interfere with their work. The observer's goal is to become virtually invisible to the users so that they will perform their work and use the system in the same way they normally do.

2.5.1 Hierarchical Task Analysis (HTA)

HTA **breaks down the steps of a task** (process) as performed by a user and describes the task as seen at various levels of detail. Each step can be decomposed into lower-level sub-steps, thus forming a hierarchy of sub-tasks. The highest level of detail might be something like: open the word processor → type your document → print it → quit. However, opening a word processor is not a one-step process. It can be broken down into something like: locate the word processing application icon → click on the icon → select Open from the File menu. A common level to break the task down to is the "keystroke level," where every mouse movement, mouse click, and key click is registered. It is based on the keystroke level model, which is a simple GOMS technique dealing mainly with observable events and organized as a single stream of sequential operators. GOMS is an acronym that stands for Goals, Operators, Methods, and Selection Rules, the components of which are the building blocks for a GOMS model.

The starting point for constructing a hierarchy is a comprehensive list of the tasks that make up a job or function.

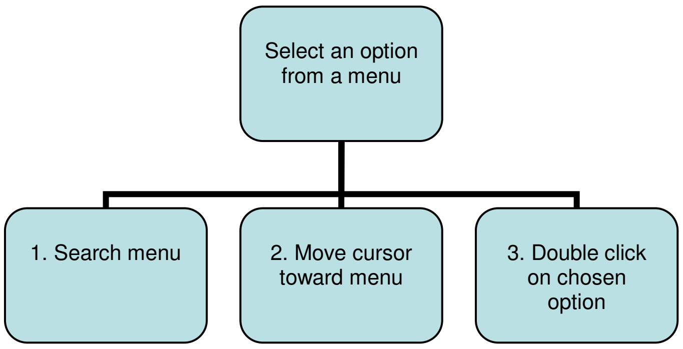

Select an option from a menu
1. Search menu
2. Move cursor toward menu
3. Double click on chosen option

Figure 3: Hierarchical Task Analysis

For example, in the diagram above, task "Select an option from a menu" has been decomposed into its enabling tasks. This implies that the learner cannot perform the task until she has performed the first, second, and third task respectively.

2.6 Interviews and Questionnaires

Many aspects of usability can best be studied by simply **asking the users**. This is especially true for issues relating to the users' subjective satisfaction and possible anxieties, which are hard to measure objectively. Interviews and questionnaires are also useful methods for studying how users use systems and what features those users particularly like or dislike.

Interviews are well suited for exploratory studies where it is still uncertain what one is looking for; the interviewer can adjust the interview to the situation. You can set up a guideline for an interview or you can choose a free talk. Guidelines have the advantage that you won't forget to discuss anything. On the other hand, a free talk can disclose issues you never thought about.

Questionnaires may contain open questions where the users are asked to write their own reply in natural language, but users often do not bother to do so, or they may write cryptic statements that are hard to interpret. Therefore, questionnaires normally rely heavily on closed questions, where users have to supply a single fact, go through a checklist, or state their opinion on a rating scale.

2.7 Focus Groups

Focus groups are a somewhat informal technique that can be used to assess users' needs and feelings both before the interface has been designed and after it has been in use for some time. In a focus group, about **six to nine users** are brought together to discuss new concepts and identify issues over a period of about two hours. Each group is run by a moderator who is responsible for maintaining the focus of the group on whatever issues are of interest. From the users' perspective, a focus group session should feel free-flowing and relatively unstructured, but in reality, the moderator has to follow a preplanned script for what issues to bring up.

2.8 Qualitative and Quantitative Testing

Qualitative testing focuses on observing how a relatively small number of test subjects interact with the product, which has already been developed at this point. This observation can reveal the majority of significant issues with a website or a system. It is the most common form of usability testing since it is relatively quick, cheap, and easy to perform. The results are subjective because they are based on the users' opinions.

Quantitative testing focuses on quantifying metrics and uses a larger number of test subjects, namely enough to get statistically significant results. In contrast to qualitative testing, quantitative testing produces hard numbers as results to questions such as whether the task was successfully completed, or what the task performance time was. It is best suited for refining and improving known processes, rather than identifying unknown issues.

The combination of quantitative and qualitative tests can give you additional information. Quantitative results show how things are objectively and qualitative results can indicate why they are like this.

2.9 Logging Software (e.g., Morae, Interact)

Logging software can be used to log data during the whole test, whether the design phase has been completed or not. It can operate cameras, record the screen and all accompanying events (e.g., mouse clicks, text input, etc.), and let the experimenter make annotations. The amount of data collected by this software is useful for statistical analysis and can therefore indicate which aspects of the program need to be improved. See http://www.techsmith.com/morae.asp for user experience testing and market research; consider the Interact product for the statistical and graphical evaluation http://www.mangold-international.com/?id=11&L=1 .

Some software can also be used to log data when testing is not the main objective. CogTool, for instance, is a prototype but also offers data logging possibilities (see 3.2.6. for more information on CogTool). Some logging software can be used when the user performs the task in a normal setting. This gives developers insight into the success of their program's features even after development has been completed. This in turn indicates which adjustments could be made in later models or for updates.

2.10 User Feedback

For installed systems, the users themselves can form a major source of usability information if one is willing to listen to their feedback. This has several advantages:

- It is initiated by the users, so it shows their immediate and pressing concerns.
- It is an ongoing process, so feedback will be received without any special efforts to collect it.
- It will quickly show any changes in the users' needs, circumstances, or opinion, since new feedback will be received whenever such changes occur.

Of course, one will tend to hear mostly from dissatisfied users and from the most vocal ones, so user feedback may not always be representative of the majority of users. Combining user feedback with other methods and a representative set of test users is recommended.

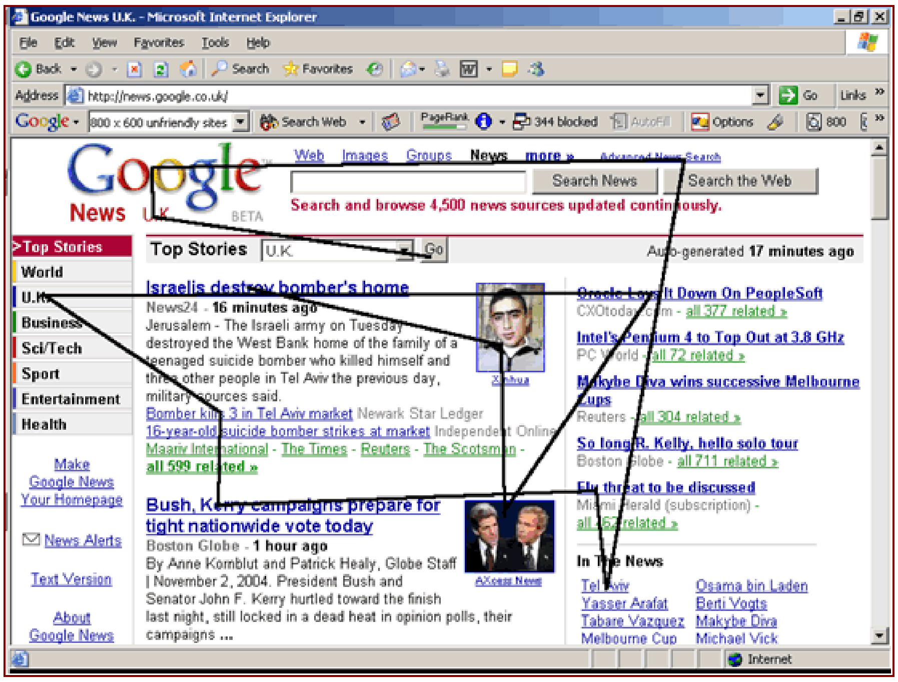
2.11 Audio Recording or Video Taping

This facilitates the experimenter's job so she can concentrate on asking appropriate questions. Afterwards, she can transcribe the interview. This method allows emotions to be recorded as well, which can be useful in the analysis. Another reason to videotape the tests is the possibility of showing them to the development team (or the management) in order to support usability issues.

Normally, if you use logging software, you could connect some cameras to it. Sometimes you will need additional cameras if you are testing a mobile device in a realistic scenario where people move around while using the system.

2.12 Eye Tracking

Eye trackers are systems that synchronize the eye movement with a special camera. It is also possible to make the **actual focus of the user** visible on the screen. It can be useful to know where and when people are looking and how often their eyes return to a certain spot when using a (mobile) device. Especially mobile scenarios allow you to learn how to reduce the cognitive load by analyzing the eye tracking data.


Google News U.K. - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://news.google.co.uk/
Go Links
Google 800 x 600 unfriendly sites Search Web PageRank 344 blocked AutoFill Options 800
Web Images Groups News more » Advanced News Search
Search News Search the Web
News U.K. BETA
>Top Stories
World
U.K.
Business
Sci/Tech
Sport
Entertainment
Health
Make Google News Your Homepage
News Alerts
Text Version
About Google News
Top Stories U.K. Go
Internet

Search and browse 4,500 news sources updated continuously.

Auto-generated **17 minutes ago**

Israelis destroy bomber's home
News24 - **16 minutes ago**
Jerusalem - The Israeli army on Tuesday destroyed the West Bank home of the family of a teenaged suicide bomber who killed himself and three other people in Tel Aviv the previous day, military sources said.
Bomber kills 3 in Tel Aviv market Newark Star Ledger
16-year-old suicide bomber strikes at market Independent Online
Maariv International - The Times - Reuters - The Scotsman -
all 599 related »
Xinhua

Bush, Kerry campaigns prepare for tight nationwide vote today
Boston Globe - **1 hour ago**
By Anne Kornblut and Patrick Healy, Globe Staff | November 2, 2004. President Bush and Senator John F. Kerry hurtled toward the finish last night, still locked in a dead heat in opinion polls, their campaigns ...
AXcess News

Oracle Lays It Down On PeopleSoft
CXOtoday.com - all 377 related »

Intel's Pentium 4 to Top Out at 3.8 GHz
PC World - all 72 related »

Makybe Diva wins successive Melbourne Cups
Reuters - all 304 related »

So long R. Kelly, hello solo tour
Boston Globe - all 711 related »

Flu threat to be discussed
Miami Herald (subscription) -
all 462 related »

In The News

Tel Aviv	Osama bin Laden
Yasser Arafat	Berti Vogts
Tabare Vazquez	Makybe Diva
Melbourne Cup	Michael Vick

Figure 4: Visualized Data of an Eye Tracker

3 Tools and Ways for Building Prototypes

Here are some ways and tools to build prototypes.

3.1 Paper Prototyping

Paper prototypes normally consist of several sheets of paper, and every sheet represents one state of the interface.

The quality of the sheets ranges from sketches to printouts, depending on the available time and effort. According to the users' action, the experimenter has to select the sheet of paper depicting the next state.

The biggest problem when conducting tests with paper prototypes is that you can only test basic navigation commands, like buttons and links. It is hardly possible to test speech or gesture input with this kind of prototype. You have to ask questions like, "Could you imagine what happens when you click on this button?" or, "Where do think you will find command xx?" In the worst case you will test the subjects' imagination and not the usability of the actual design. You have to keep this in mind when analyzing the data. The experimenter has to know the storyboard of the mock-up very well in order to keep the test going and ask the right questions at a specific state.

With a paper prototype, you can have users test early design ideas at a low cost. You can read more about paper prototyping on Jakob Nielsen's blog entry: http://www.useit.com/alertbox/20030414.html

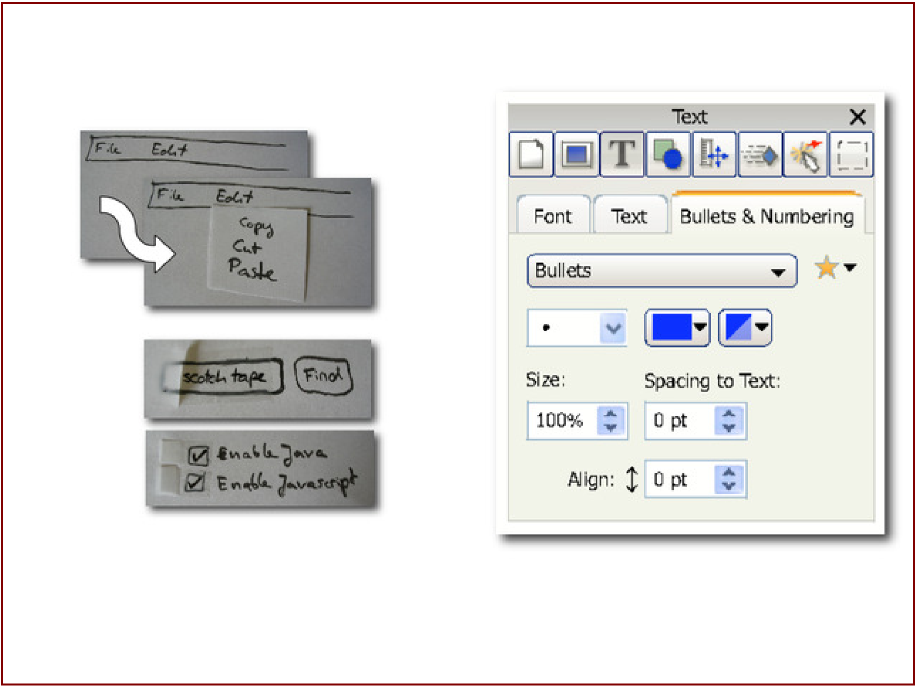


Figure 5: Examples for Paper Prototyping

3.2 Prototypes from Visio, PowerPoint, etc.

These prototypes can be somewhat more functional and show more visual similarity with the final interface. Although you cannot test special input possibilities (e.g., speech or gesture), these prototypes give the user a living idea of how the system will work. A benefit for the experimenter is the decrease of her cognitive load because she does not need to choose the next frame or picture to be presented to the subject. She can concentrate on the subject's behavior and statements.

There are several ways to build such prototypes. HTML, Visio, PowerPoint, and Flash are some of the options. Which tools you use will depend on your preferences or resources.

3.2.1 HTML

An important interaction design benefit of HTML is the way it lends itself to ongoing user testing. Because of their interactivity, HTML wireframes can be regularly used to test design ideas on the fly with people in the office, friends, or anyone who happens to stop by. When there is no one around, you can post designs on the web, instant message a friend to try them out, and get some quick, useful feedback. You can find more detailed information on the following site: http://www.digital-web.com/articles/just_build_it_html_prototyping_and_agile_development/

3.2.2 Visio

The reason why Visio is the favorite prototyping tool of many interaction designers is because of its ready-made interface objects, which you can drag-and-drop onto pages as well as its ability to link pages together and export them as web pages. You can read more about prototyping with Visio here: http://www.guuui.com/issues/02_03_02.php

3.2.3 PowerPoint

Its familiar graphics front end makes the development of the prototype graphics quick and easy. Although PowerPoint doesn't have as much graphic power as, say, Photoshop, the Format-Autoshape dialogue (right click on an object) provides control when creating graphics that are precise enough for interface prototyping.

Because most people in business already have PowerPoint and Excel, designers and researchers can develop prototypes, usability tests, and experiments with tools they already own instead of having to buy special-purpose products. Furthermore, in some cases, the files for a study can be sent to the user/subject and run remotely on their own machines. This makes running a small study much more convenient than it might otherwise be. Victor Riley gives an introduction to PowerPoint and prototyping:
http://victorriley.blogspot.com/2007/02/powerpoint-prototyping-introduction.html

3.2.4 Flash

Flash is a good tool for building broadly used prototypes, and has a scripting language that allows for a certain amount of depth. Its most common use is for quickly adding animation, sound, and video to a prototype. You can find more information on this website: http://www.johnniemanzari.com/flashlabs/. Adobe Flash is available at http://www.adobe.com/products/flash/.

3.2.5 Axure RP Pro

Axure RP is commercial software used to build prototypes. It can enable application designers to create wireframes, flow diagrams, prototypes, and specifications for applications and websites. You can read more about this software on the product page:
http://www.axure.com/

3.2.6 CogTool

CogTool leverages the concept of a design storyboard to create accurate models of performance behavior. Designers can simply draw onto an existing image of a device or software interface to create a single frame of the storyboard. Individual screens can be linked to form a complete storyboard (Figure 6) for one or more tasks. Complex interfaces can be mocked up quickly using these simple techniques.

Once the mockup has been created, a designer can demonstrate the steps of a particular task by directly interacting with the frames of the storyboard. As the demonstration proceeds, CogTool builds a model of the task using heuristics to place "think" operators that approximate cognitive processing time. You will find more information about this software and a download link on this page:
http://www.cs.cmu.edu/~bej/cogtool/index.html

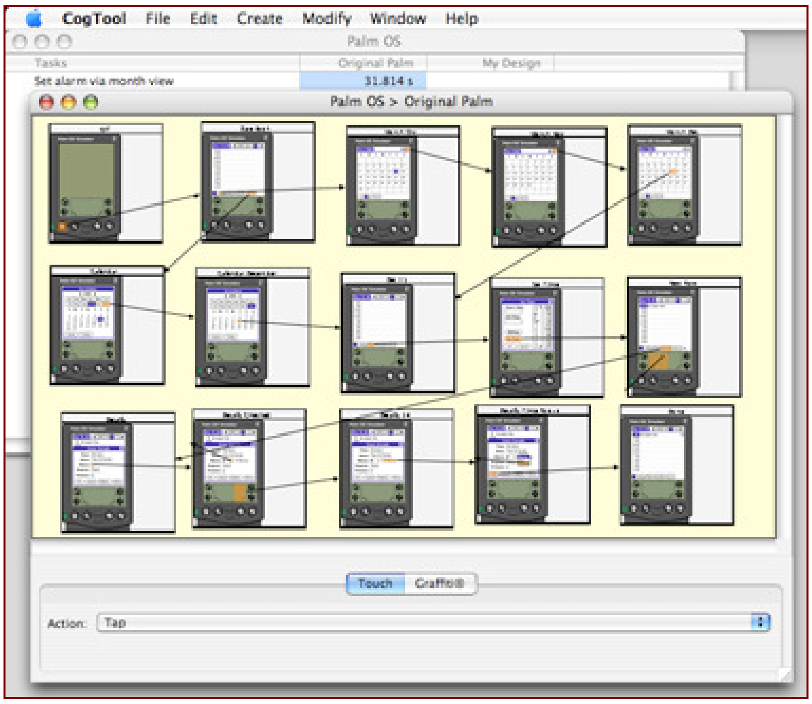


Figure 6: Editing a Design Storyboard in CogTool

3.3 Working Prototypes (Flash, Java, C++, etc.)

If parts of the system are already stable and running, you can use them for testing. You will receive direct feedback on the real system and you can test the input possibilities if implemented. Naturally, this is the best way to test, but if usability issues occur, it will also take a lot of work to get them fixed. It is not useful to develop and test with a software prototype until the concept of the system is proven to be flawless. Otherwise, it may be necessary to restructure the entire system according to the results of the test.

4 Usability Studies

Usability studies (or usability tests) evaluate a prototype by testing it on users.

4.1 Test Goals and Test Plans

Before any testing is conducted, the purpose of the test should be clarified. This will have considerable impact on the kind of testing to be done. A major distinction is whether intention of the test is as a formative or summative evaluation. Formative evaluation is done in order to help improve the interface as a part of an iterative design process. The main goal of a formative evaluation is to learn which aspects of an interface are positive and which are negative. The design can be improved according to the results. A typical method to use for a formative evaluation is a thinking-aloud test. Summative evaluation, by contrast, aims at assessing the overall quality of an interface. For example, to test which is more successful, a user may be asked to choose between two different approaches to complete a task. Summative evaluation may also be part of a competitive analysis; in this case the user performs the same task but with two different versions of the program prototype. As a result, the time required to finish a certain task or the number of errors made by the users can be tested for each version to determine whether the change made was really an enhancement.

4.1.1 Test Plans

A test plan should be written before the start of the test. The following issues should be addressed:

- The goal of the test: What do you want to achieve?
- Where and when will the test take place?
- How long is each test session expected to take?
- What equipment, e.g., hardware and software, will be needed for the test?
- What should the state of the system be at the start of the test?
- What should the system and network load and the response time be?
- Who will serve as experimenter for the test?
- Who will the test users be and how will you get a hold of them?
- How many test users are needed?
- What test tasks will the users be asked to perform?
- What criteria will be used to determine when the users have finished each of the test tasks correctly?
- What user aids (manuals, online help, etc.) will be made available to the test users?
- To what extent will the experimenter be allowed to help the users during the test?
- What data is going to be collected, and how will it be analyzed once it has been collected?
- What will the criteria be for declaring the interface a success?

4.1.2 Test Budget

The test plan should also include a budget for the test. Indirect costs, such as company staff or rooms, may or may not be formally charged to the usability budget for a specific project, depending on the accounting mechanisms, but they should be included in the internal budget. Typical cost elements of a user test budget are:

- Usability specialists (if available) to plan, run, and analyze the tests.
- Administrative assistants to schedule tests users, enter data, etc.
- Software developers to modify the code to include data collection or other desired test customization.
- The test users' time.
- Computers used during testing and analysis.
- The usability lab or the room used for the test.
- Videotapes, DVDs, paper, and other materials.

4.1.3 Pilot Test

No usability test should be performed without first having tried out the test procedure on a few pilot subjects. This will typically show if instructions are in any way incomprehensible or if questionnaires could be misinterpreted. It may be that a briefing is needed before the test session. You can check if the time for a session matches the estimated time. Most commonly, tasks are more difficult than expected, but sometimes it may also be the case that some tasks are too easy. You have to decide if the tasks need to be adapted in order to challenge the user. However, some easy tasks let the user get used to the system and the test situation. Furthermore, the pilot test can reveal problems in the definition of user errors or task completion.

4.2 Resources

Depending on limitations, we can suggest different methods and equipment to conduct studies.

4.2.1 Financial Limitations

The biggest expense is the setup of a usability lab. Depending on your budget, your facilities can range from a simple testing room to a well-equipped laboratory.

Different Room Setups

This is the simplest way to conduct a usability test with a paper prototype. The experimenter is sitting face to face with the subject and can arrange the printouts according to the statements of the subject. To ease the work of the experimenter, it is useful to videotape the subject. This way, she can concentrate on the subject's statements and ask the appropriate questions. The session will be transcribed afterwards.

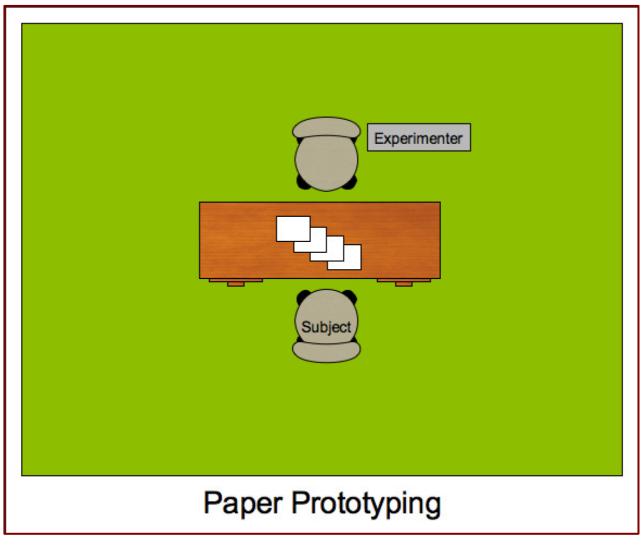
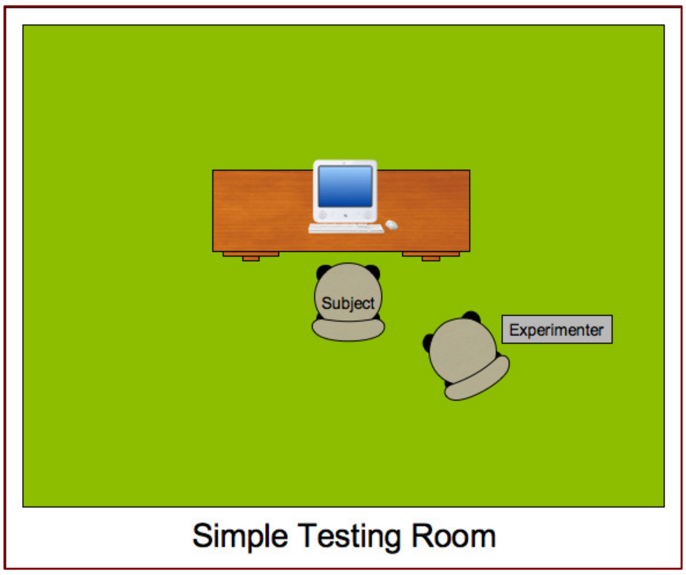


Figure 7: Simple Test Setup for a Paper Prototype



This is the basic configuration for usability testing. You need a room with the system and two chairs in it. The subject will use the (desktop) system and the experimenter writes down the subject's actions and comments. Again, videotaping the subject will ease the work of the experimenter.

Figure 8: Basic Testing Room

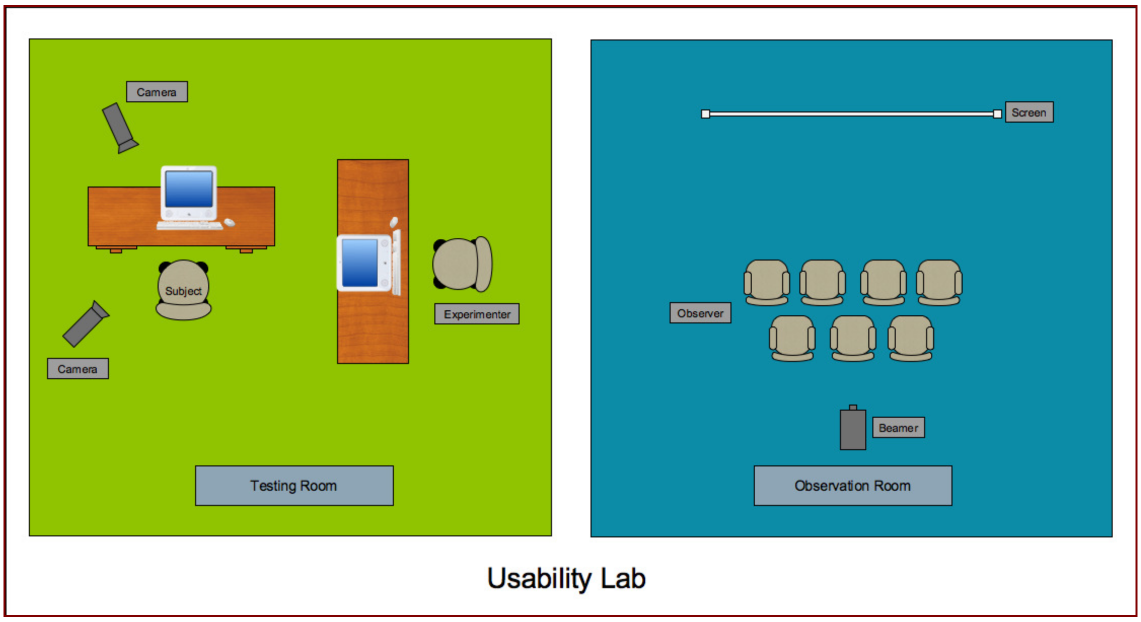


Figure 9: Usability Lab

The graphic above shows a usability lab. It consists of a testing room and an observation room. The advantage of this split is that more people can watch the testing sessions and the crowd of people does not irritate the subjects. Cameras record the subjects and the screen and appropriate software combines the pictures with annotations by the experimenter. All of the information will be transmitted to the observation room where the observers can take their own notes or discuss the events and the subjects' reactions.

Logging Software

Logging software is a good instrument to collect statistical data with. Moreover, you might see coherences between design problems and the reactions of the subjects. But logging software is not essential for usability testing. There are some freeware loggers and a lot of freeware available which replaces some functions, like screen capturing.

4.2.2 Time Limitations

The question of time can be divided into what is required to realize the whole study and the time spent on a single session.

Realization of a Study

A study consists of different phases: (1) building a prototype, (2) creating appropriate tasks and choosing subjects, (3) preparing questionnaires, (4) conducting the tests, and (5) analyzing the data.

1. The time to build a prototype depends on its character. A paper prototype is built faster than a hi-fi prototype but there are differences in the significance of the results.

2. The creation of tasks is connected to the selection of the subjects. If completing the tasks requires special skills or knowledge, you will need a special user group to test the system. Otherwise you can randomly choose people you know.

3. There are a lot of questionnaires dealing with usability. Choosing the most adequate and then customizing it to your needs is an easy way to construct one. If you have to stick to specific ISO standards or other specifications, you will spend more time designing your own questionnaires.

4. The time for the actual test procedure should not exceed 60 minutes. Take preparation time for the experimenter and the equipment into account and keep in mind that subjects can drop out. Allow at least one month for a usability test. Depending on the equipment, you will need about a week to prepare the room, software, cameras, and questionnaires. At the same time you have to find adequate subjects. Depending on the number of subjects, you will need another week to conduct the tests. For the analysis of the data, the formulation of the results, and the transformation into appropriate suggestions, you should schedule about a week, too. Also, it is always good to allow extra time for unforeseen events.

5. Usually, a summary of qualitative statements takes longer than an analysis of quantitative data. After that, you have to transform the results into issues that need to be discussed by the development team.

Specific Session

A single session should not exceed an hour. Bear in mind that it is a test and the subjects could perceive the situation as exhausting. If a session needs to be longer than an hour, allow extra time for breaks.

Rapid Testing

Sometimes it is sufficient to ask some colleagues for their opinions about a design. This is an easy and fast way to deal with smaller design issues during a development phase.

4.2.3 Limitations on Manpower

Experimenter

There is no absolute need to have a skilled experimenter, but it is better to have an experienced person conducting the tests. You need someone who understands the concepts of usability and can deal with people. The most important thing is for her to watch the subject closely and ask the right questions at the appropriate time. This should be practiced before interviewing subjects.

Subjects

Normally, for a qualitative test, four to five subjects are sufficient in order to find most of the usability issues. Qualitative testing collects the users' subjective responses to the program or feature being tested.

You will need more subjects to conduct a quantitative test. A quantitative test disregards the users' opinions and focuses on collecting hard numbers, i.e., how many mouse clicks the user required to perform a certain task, or how long the user required to locate a particular button on the interface.

If you do not have any potential subjects at hand, ask colleagues or friends what they think about the design. It is better to have some, albeit biased, comments than none at all.

Dealing with Subjects

There are some things to keep in mind when working with subjects:

- It is important that the subjects know that the system will be tested and not they. This will reduce stress and improve the relationship between experimenter and user.
- Tell the users in detail what they have to do and that you will ask questions. This way they are prepared before the test starts and you will not have to interrupt their tasks.
- Think of what to pay. If you pay too little, the subjects may be frustrated and judge the system too harshly. If you pay too much, they may oversee some flaws. It is best to have unbiased volunteers, who do not expect compensation.

5 Projects and Use Cases

The THESEUS program consists of six use cases: ALEXANDRIA, CONTENTUS, MEDICO, ORDO, PROCESSUS and TEXO.

The recommended scenarios for the use cases are explained later in section 6. We will start by giving a short description of the use cases and then explain how we extracted the specific scenarios.

5.1 ALEXANDRIA

ALEXANDRIA will be a consumer-oriented dialogue interface for accessing a knowledge database. This means that users with no special knowledge use the system to browse through a large amount of data. They can upload and manipulate their own files. The system can be used on a desktop computer and a mobile device as well. Administrators are able to control and maintain a complex database system. To evaluate ALEXANDRIA, we propose using scenarios 1, 3, 4, and 7.

5.2 CONTENTUS

CONTENTUS will allow members of a large user group to make use of digital data sources like books, texts, images, music, sound archives, and videos and, in turn, improve them and make additions with their own information. There are also plans for experimental development of a new, dual-purpose services platform; this will allow the community-based semantic management of digital IT values, and make the semantic analysis of entire documents possible. Scenarios 2 and 6 should be appropriate in evaluating CONTENTUS.

5.3 MEDICO

In MEDICO, a universally usable search engine for medical images will be created. It will permit direct semantic access to medical image databases to support individualized diagnoses and therapy plans, as well as biomedical and epidemiological research. The target group consists of physicians, researchers, and application developers in the areas of medical informatics and healthcare IT. We believe that scenario 2 is adequate for the evaluation of MEDICO.

5.4 ORDO

The application scenario ORDO is intended to research and develop semantic technology, thereby creating new services and software tools that will enable users to organize their entire store of digital information. Since the organization process will be automatic, it will also be transparent, and require no effort by the user. The problem-free processing of extremely large quantities of data is required to make the graphic visualization of that data in a knowledge model possible. Highly scalable solutions are therefore aimed at. Unlike the solutions used until now, this personalized linking allows unstructured and structured data to be organized in a uniform manner, making efficient, individual knowledge management possible. ORDO should be evaluated using scenario 2.

5.5 PROCESSUS

The objective of the PROCESSUS application scenario is to create an IT-based corporate control system. PROCESSUS is mounted on the technical platform used in the TEXO application, and provides a number of additional services that define the semantic Business Integration Platform. This will involve integrating the various TEXO and PROCESSUS modules and components into a single platform, with the addition of some new functions. PROCESSUS can be evaluated using scenarios 3 and 5.

5.6 TEXO

The TEXO system implements an interaction scenario where the user works with e-business tickets in order to purchase services online. This will be used foremost in mobile devices but may also be used on a desktop computer. TEXO not only develops a search feature to help find and arrange offered services on the platform, but also tries to combine this with the possibility of building a personalized desktop. Appropriate scenarios for TEXO will be scenarios 4 and 2.

6 Defining the Scenarios

A given scenario will provide specific design and testing recommendations. In order to come up with the different scenarios, we first extracted the following substantial information from the different projects. We discerned between the people who will use the system, where they will use the system, and what they will do with the system (Figure 10).

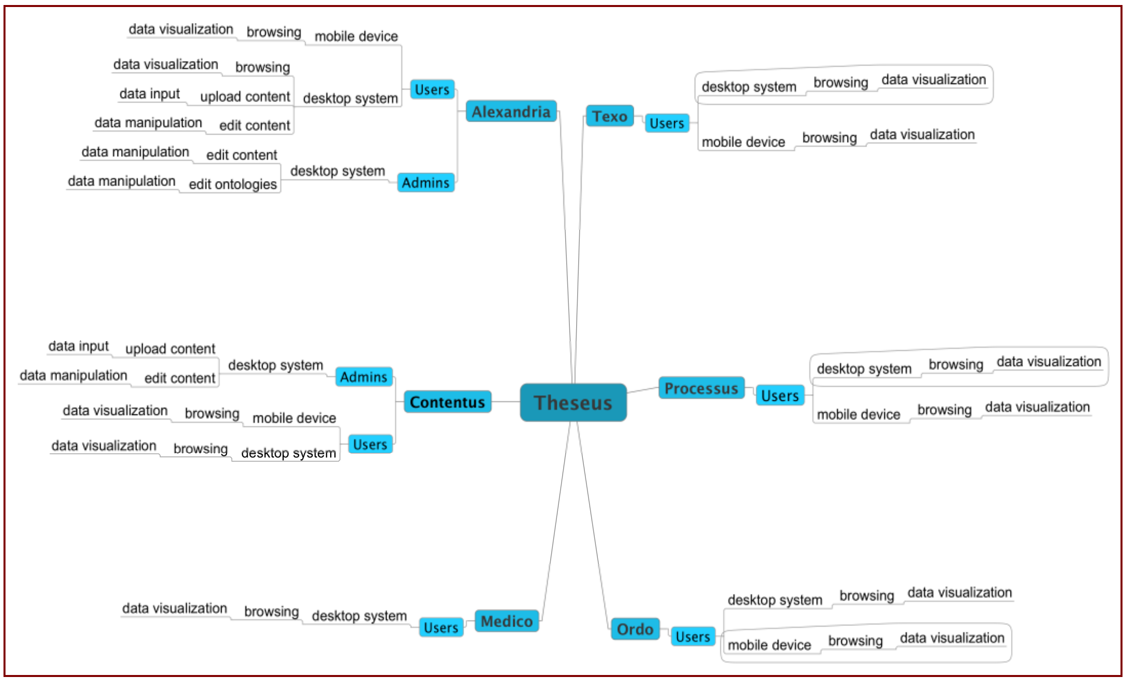


Figure 10: Use Cases in THESEUS

After considering the results, we were able to say the THESEUS program works with seven different scenarios (Figure 11).

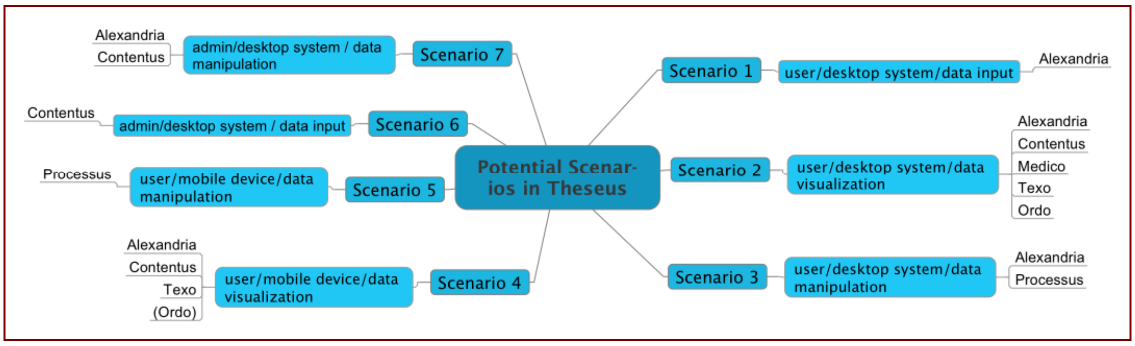

Potential Scenarios in Theseus
Scenario 1
user/desktop system/data input
Alexandria
Scenario 2
user/desktop system/data visualization
Alexandria
Contentus
Medico
Texo
Ordo
Scenario 3
user/desktop system/data manipulation
Alexandria
Processus
Scenario 4
user/mobile device/data visualization
Alexandria
Contentus
Texo
(Ordo)
Scenario 5
user/mobile device/data manipulation
Processus
Scenario 6
admin/desktop system / data input
Contentus
Scenario 7
admin/desktop system / data manipulation
Alexandria
Contentus

Figure 11: Potential Scenarios in THESEUS

Based on these facts, we can provide a decision tree that leads to specific recommendations for designing prototypes and testing them. To find an adequate scenario, follow the tree in Figure 12 from left to right according to the corresponding requirements. This will lead you to specific design and testing recommendations.

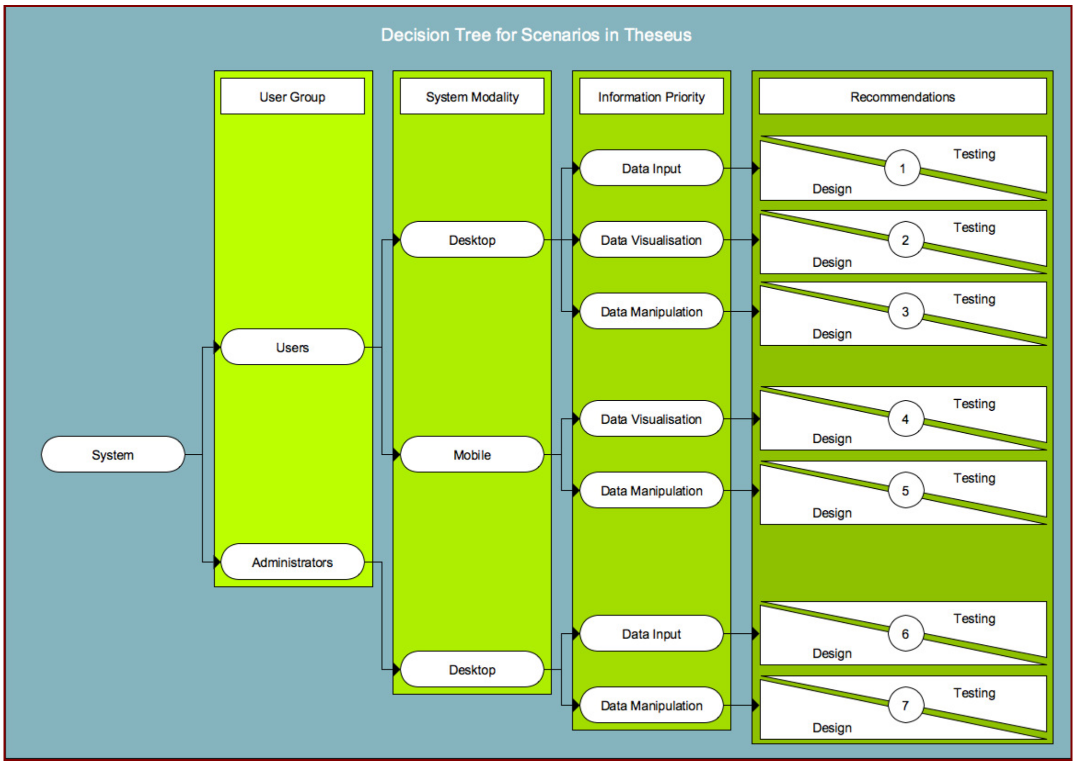


Figure 12: Decision Tree for Recommendations

6.1 Discrimination Criteria

The specific scenarios can differ depending on their categorization in three fields, which are essential for the design. The first criterion is the user group because the system has to be adapted to the user's skills and needs. Second, you have to think about situation in which the system is being used, because mobile interaction differs from desktop use. The third discrimination is called information priority. In this category answers are found to the questions, "What is the focus of the system?" and "Will the main task of the user be browsing or data input?"

6.1.1 User Groups

There will be at least two user groups: system end users and administrators. They have to be differentiated between because they perform different tasks with the system. Administrators control or manipulate the functioning of the system while users use the system to solve a problem. At a later stage, we can include users with more or less special skills for more focused user groups. If you design a system for administrators or users with more special skills, you should test the system primarily with this user group. Otherwise, the results are biased because of the lack of task knowledge. In case the system is used by average users, group them according to age and gender to get more valid results.

6.1.2 System Modality

The user's interface can be a mobile device or a desktop system. Even though it is possible for administrators to use a mobile device to manipulate data, it is rather inappropriate. Depending on the system modality, we can define some constraints. When a mobile device is used, display size and input modes are restricted. Depending on the current situation and task to be performed (e.g., the input of people's names), the user may not be able to use speech input or a keyboard. Privacy concerns can also keep people from using speech input in public. In addition, typing an email while you are walking requires a lot of attention. When using a mobile device, the environment can constantly change. Thus, it is important to test the system output under different conditions. Audio signals can be missed if side noises occur and the display may be hard to read in the changing light.

When testing the system, try different tasks under various conditions to see if the system works as expected and whether the users can reach the goal.

6.1.3 Information Priority

The tasks can be grouped into data input, data visualization and data manipulation. The description indicates the main task or the priority. Data input can be seen as a kind of upload. The user can add files or other data to the systems. Data visualization is the way in which the system presents data to the user. In this case, the user enters a search request and the system offers results. Data manipulation is a combination of input and visualization. In this case, the system presents data to the user and she can change them. After that, the system has to visualize the changes. The focus lies on the interaction between user and system.

If the system focuses on data input, the interface should provide tools to support the user and simplify her work. To enter a larger amount of data, the input modes are not suited in equal measure. For most people using a keyboard is the best way to formulate a request, but speech or multimodal input can be a good alternative. As mentioned before, the need for privacy or the possibility of disturbing others may keep people from using speech input. When testing the system, it is good to ask people when they would use the different input modes.

There are a lot ways to present data to the user: functions, graphics, tables, rules, etc. Some are more appropriate for presenting large amounts of data than others. You have to consider that people have different ways of reading data. Some like textual descriptions while others prefer graphical visualizations.

If possible, let the user decide on the visualization method and allow personalization.

The manipulation of data mass could be done by using natural language. But to control the changes, some users might prefer a graphical presentation. In advanced interface environments there are other ways to handle huge amounts of data, like gestures or touch screens, but users must decide what is best for them.

6.2 General Recommendations for Designing and Testing

Before we give recommendations, we will discuss several input methods. Choosing the appropriate input method is essential for the design process and also affects the evaluation.

6.2.1 Input Methods

Because input is essential for all of the projects, we will introduce the most common input methods and give a short description.

Keyboard and Mouse

Keyboard and mouse are the standard input methods of a desktop computer system. Keyboard layouts can vary according to localization issues, e.g., German or French, or ergonomic issues, e.g., the Microsoft Natural Keyboard. The use of trackballs, tablets, or other pointing devices instead of a mouse is less common and normally depends on special circumstances.

Speech (Natural Language)

Speech recognition engines convert the acoustic signal to a digital signal and deliver recognized speech as text to the application. Most speech recognition engines support continuous speech, meaning a user can speak naturally into a microphone at the speed of normal conversations. In order to interact by natural language, a dialogue system is needed that accepts the automatic speech recognition (ASR) as input.

Touch Screen

A touch screen is a display that can detect the location of touches within the display area. This allows the display to be used as an input device, removing the keyboard or mouse as the primary input device for direct interaction with the display's content.

Gesture

We must distinguish between drawing a sign, e.g., with a finger on a touch screen, or moving a device, such as a PDA, in space. The first possibility is used, for example, in a Firefox plug-in to execute commands by right clicking and moving the mouse in an appropriate direction. The second type is used in some iPhone games, for instance, to control balls or bricks by tilting the device in the appropriate direction.

Comparison of Input Methods

Speech input and keyboard input are adequate methods for entering larger amounts of text. In mobile scenarios, keyboard input is less appropriate than speech input because of the size of a usable keyboard. It has also been shown that speech input is very useful for executing commands like speech dial on mobile phones.

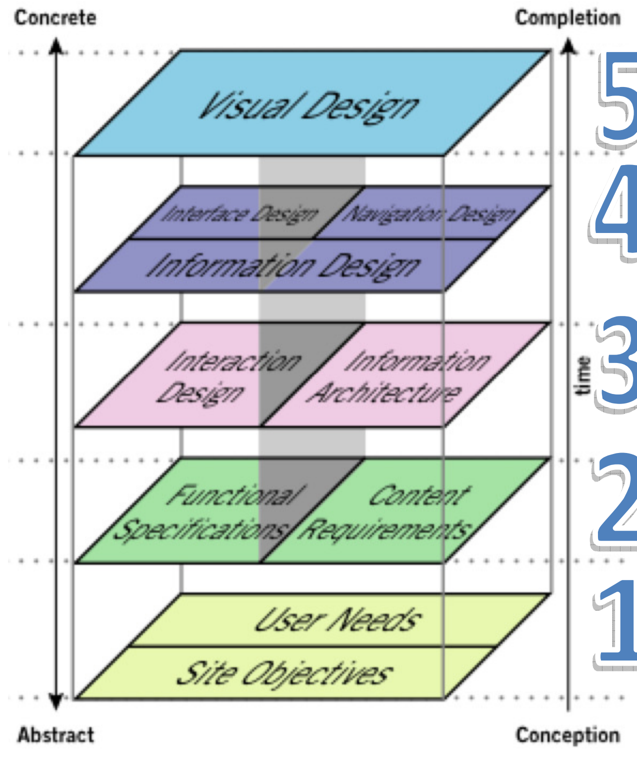
Mouse, touch screen, and gesture input are useful methods for executing commands or manipulating the display content. We can distinguish between direct and indirect manipulation.

A mouse is an indirect input device, because the user has to move the mouse on the table to indicate a point on the screen, whereas a direct input device has a unified input and display surface. Direct input devices, such as touch screens, are not necessarily easier to use than indirect devices. Occlusion is a major design challenge. The finger or pen occludes the area at which the user is pointing, so the user may not realize that she has activated a control. Occlusion by the hand or an arm also may cause the user to overlook pop-up menus, dialogue boxes or status indicators. Though options are limited, gesture input can cope with some of these challenges because moving the whole device will not occlude the display.

In mobile scenarios a touch screen is better than a mouse, because no additional space is needed. The direct input method uses the screen to display and manipulate data.

6.2.2 Recommendations

To give design and testing recommendations we will follow Jesse James Garret's sectioning in his book *The Elements of User Experience*. He describes five cumulative planes. Every plane has its own issues that must be considered. From abstract to concrete, these are the strategic plane (1), the scope plane (2), the structure plane (3), the skeleton plane (4), and the surface plane (5). We will give a short summary of recommendations for designing and testing on these planes that will count for all scenarios. After that we will focus on the scenarios in detail.



5 The Surface Plane is what the user will see on the respective user interface.

4 The Skeleton Plane is the concrete expression of the structure. Here, we describe how design elements are arranged in the interface.

3 The Structure Plane defines how the information or pages in the system are connected and how the user will get from one to another.

2 The Scope Plane constitutes the features and functions of the website or system.

1 The Strategy Plane incorporates why users want to use the system and what the operator wants to get out of the system (system/site objectives).

Figure 13: The Elements of User Experience

6.2.3 Design Recommendations

1 ***Strategy Plane***

When considering the strategy plane, you need to know which goals the user is supposed to achieve by using the system.

Defining the users and their needs is the first step in the design process. It is useful to create personas that represent a special user group.

A persona is a fictional character constructed to represent the needs of a whole range of real users.

With a persona in mind it is easier to estimate the importance or usefulness of certain features. Depending on the system you can generally differentiate between power users who will use the system routinely and those who will only use the system occasionally. This can lead to different designs. Take, for instance, a video recorder. When you program one every day, after a while, you don't need a manual to get the job done. But if you only do it once a month, by the time you have to do it again, you will have forgotten the procedure and will need the manual.

2 *Scope Plane*

On the scope plane you have to define the system's capacity and then the technical requirements.

Once this is settled, there are less discussions of what needs to be programmed. For better understanding, you can use personas and create little stories around them. A story can consist of several scenarios because a person can use the system in different ways. A scenario is a narrative describing how a person can use the system to complete a task. For example, Tom can use his mobile phone to check his email while sitting in the bus to work. Later at work, he will use his desktop computer for the same task. Another way to find out about requirements is to ask users. If you let people talk about the system, they may find out about requirements they never thought of. The problem that will occur is having too many requirements for the system. Then you have to estimate several requirements' priorities.

3 *Structure Plane*

On the structure plane you should transform the abstract issues from the scope plane into more concrete factors to determine what the user experience will be.

This experience consists of a cycle of action and reaction. Either the user acts and the system reacts or the other way around. Every time the user uses a system, she will improve her mental model of the system. But this only works if the conceptual model of the system matches the user's mental model. If the user can predict what the system will do, she is more willing to do trial and error. For example, an electronic shop should follow a concept similar to a real shop. This is a good way to prevent users from making errors, because they have a working model that can be easily assigned to a new activity. It is somewhat harder to introduce a concept that the user cannot compare with something known. Then you have to prevent the user from making errors by excluding certain actions at certain times. As you cannot exclude every error, you have to provide possibilities to recover from errors, for example, an "undo" function.

4 *Skeleton Plane*

The skeleton plane brings together interface design, navigation design, and information design.

Interface design implies standards and agreements like using radio buttons instead of checkboxes, if you have an exclusive choice. Depending on the available desktop space and the number of possible choices, it can be adequate to use dropdown lists or list boxes. This depends on the widgets used in the interface. Such decisions should be made in the course of team discussions to determine a majority opinion. **Navigation design** is a special part of interface design and involves three aspects. First, the user must know how to get from A to B. This is the basic function of navigation. Second, it must provide information about the hierarchy and relationships between the different elements. The user can, however, rank the offered links. Third, the navigation design has to show the relationship between the content and the page the user is currently viewing. This helps the user find appropriate information according to her goal. **Information design** is about how to present (and represent!) information so that people can understand it. It ranges from an adequate diagram to the right sequence of information. For example, when specifying your address, you would not name the zip code before the street.

5

Surface Plane

The skeleton plane deals with the logical arrangement of the design elements. On the surface plane we are concerned with the visual presentation of the elements.

It is about how to keep the user's attention on the important elements. There are consistent patterns in how people move their eyes across a newspaper or a web page. You can also guide the user's attention by using contrasts and uniformity. Contrasts will draw the attention and uniformity will keep it, if used in the adequate way. For example, a headline that is highly contrastive will draw a user's attention. The associated article underneath the headline is presented more neutrally to keep the user's attention. Colors are another way to guide the user's attention and divide sections. Fonts can provide additional information. It is useful to present links in a different font than the rest of the text to make them more visible to the user. Consistency in terms of a corporate identity is also important since it tells the user which elements belong to an application. Style guides define colors, fonts, and the page layout all over the interaction device.

6.2.4 Testing Recommendations

1

Strategy Plane

To test the design of the strategy plane, you should ask potential users if they are satisfied with the features that you provide in the system.

This is best done with a questionnaire. List the features that you will provide in the system and let the user estimate the usefulness of each on a scale. In this phase of the design it may be useful to reach as many (potential) users as possible. An online questionnaire may be advisable where the URL can be spread by email. On the other hand, you must ask yourself if you can develop the features the users want. The best way to find out about this is through group discussion within the developing team.

2

Scope Plane

The first thing to do for the scope plane is to evaluate if it is **sensible** and **possible** to implement the requirements. Then, a decision is reached about which requirements should be implemented.

You have to discuss how feasible it is to implement the requirements. Do they correspond with the strategy? These issues should be discussed with the development team. You can apply the cognitive walkthrough technique using realistic scenarios to check whether the requirements for completing a certain task are useful, i.e., does the scope cover the task examples?

3

Structure Plane

The structure can be tested with non-working prototypes, for example, paper prototypes.

An advantage is that they are cheap and if you find a flaw, it is easy to correct. During this phase, you should involve the potential user directly. An appropriate way to test a conceptual model is the “think aloud” technique. You can record the user’s thoughts while she works with the prototype. This will show you if the user’s mental model fits with the concept. Here you can discover possible user errors by identifying differences between the conceptual model of the system and the user’s mental model.

4

Skeleton Plane

The best way to test in the skeleton plane is to build or sketch different prototypes and let the subjects rate them with a questionnaire.

There are many design issues and everyone has their own preferences, for example, favoring list boxes to dropdown menus. A questionnaire will provide statistical information on how many subjects like or dislike particular features presented in the prototypes. Furthermore, it might be useful to discuss these issues within the development team. Normally, the discussion will end with several potential solutions.

5

Surface Plane

In the surface plane, it should be tested if the user looks at the right place at the appropriate time.

The most advanced way to test this is by using an eye tracker. An eye tracker is a device which measures eye positions and eye movements. It will show you exactly where the subjects are looking and for how long. The most popular variant uses video images from which the eye position is extracted. A simpler way is to ask the subjects where they are looking first and what draws their attention. You can also determine the most dominant elements on the entire screen or in a specific window by looking at the display from the other end of the room. What catches your eye first?

6.3 THESEUS Scenarios

Following the classification we introduced in the previous section (see Figure 11), we will now describe the seven scenarios and what we think is important for the design and testing of each. The recommendations will follow the differe nt planes we have just presented. The scenarios are as follows:

- Scenario 1: User at a desktop computer, focus on data input
- Scenario 2: User at a desktop computer, focus on data visualization
- Scenario 3: User at a desktop computer, focus on data manipulation
- Scenario 4: User with a mobile device, focus on data visualization
- Scenario 5: User with a mobile device, focus on data manipulation/input
- Scenario 6: Administrator at a desktop computer, focus on data input
- Scenario 7: Administrator at a desktop computer, focus on data manipulation

We will adapt the scenarios and give more detailed recommendations when more information about the use cases is available.

6.3.1 Scenario 1: User at a desktop computer, focus on data input

Scenario 1 concentrates on data input for a user on a desktop system. In this case we can assume that environmental factors such as light and noise level will remain relatively unchanged. Furthermore, the user will have a keyboard and mouse for data input and the processing power of the system is high.

Recommendations

Strategy Plane

Designing

As previously described, the most important thing in this phase is to define the user groups and which goals they will achieve using the system. Creating personas is a useful means for this. Normally, a persona has habits and preferences like a real person. With a persona in mind, it is easier to estimate the importance of particular features. You can think of different input modalities. Will people use speech input or do they prefer the keyboard because they are used to it? Some situations are unfavorable for the use of speech input. For example, if the user is not alone, she might disturb other people.

Testing

To validate your decisions you have to consult potential users of the system. The created personas will help you determine which real users should be approached for testing. The best way to get the information you need is to give them a questionnaire. They can rate the importance of features, for example, “How valuable is speech input for you?” The easiest way to reach a lot of people is to use an online questionnaire. You may find potential users in discussion boards or by email.

Scope Plane

Designing

In the scope plane you have to decide which features will be implemented according to the results of your survey in the strategy plane. A discussion within the development team about these results can help determine the feasibility.

Testing

To validate your decisions it is helpful to use the cognitive walkthrough method. With a persona in mind you can try to go through the steps of a task and ask yourself whether you will use the feature adequately to achieve your goal.

Structure Plane

Designing

After you have defined the requirements, you need to know how to structure these features. You have to create a concept that includes the features and defines how they will work. If you can fall back on a concept that already exists, it will simplify users’ understanding. In this case you can use the concept of a word processor, because a lot of people use one to write letters or other documents. If you stick to a specific word processor, like Microsoft Word, it will reduce the learning time and error rate because most users are familiar with the interface and how to work with it.

Testing

The intention here is to test if the concept of the system matches the user's mental model. Discrepancies will be visible when the user is confused or makes a mistake. You can use the "think aloud" method for non-working prototypes and a testing session, including logging software, for a working prototype.

Skeleton Plane

Designing

Some things must be considered when you plan the design of the interface. Will the user use this system principally? Should the system fit into the operating system and collaborate with other programs? If the user works exclusively on that system, you can choose the elements that fit best. You have to face an extended learning phase, but it can speed up the workflow. When the user works with several programs at the same time, it is recommended to adapt the elements of the system in order to avoid misunderstandings and confusion.

Testing

Depending on the design, the test should follow the considerations above. If the system will be used together with other programs, it should be tested in a scenario with these programs. When you use your own design elements, give the subjects time to become familiar with the system and ask them explicitly for their opinions about the design.

Surface Plane

Designing

The visual design should not exclusively follow aesthetical issues. The design should reinforce the structure of the system and clarify the functions. Colors and fonts should not only enhance the recognition of a corporate identity but also improve readability.

Testing

This is the appropriate time to perform a final evaluation. The scenario should be as real as possible and include factors such as time pressure and noise. Hardware that will be used at work should be employed. Tasks similar to the later work should be constructed. If you need statistical significance, we recommend using logging software to collect data.

6.3.2 Scenario 2: User at a desktop computer, focus on data visualization

In this scenario, the user sends a search request to the system and receives results. The visualization of the data is the focus in this scenario. The users' main action can be defined as browsing.

Recommendations

Strategy Plane

Designing

You have to define the user groups and tasks they will perform. In this scenario they will use the system to browse through data. This means the system should provide features to ease the task. You could think of hardware aids like a touch screen or software features like a radar windows. A brainstorming session with the design team can generate a lot of possible, interesting features.

Testing

You can create personas and try to determine if they would use the features in order to determine how useful they are. List the features in a questionnaire to be distributed amongst possible user groups. Another way is to watch people performing similar tasks with an existing system and ask them what they think about your ideas.

Scope Plane

Designing

In the scope plane you have to decide what features will be implemented according to the results of your survey in the strategy plane. A discussion within the development team about these results can help to determine their feasibility.

Testing

To validate your decisions it is helpful to use the cognitive walkthrough method. With a persona in mind you can try to go through the steps of a task and ask yourself whether you will use the feature adequately to achieve your goal.

Structure Plane

Designing

After you have defined the requirements, you need to know how to structure these features. You have to create a concept that includes the features and defines how they will work. If you can fall back on a concept that already exists, it will simplify users' understanding. Concepts that are familiar to a lot people are Internet browsers or file managers. If possible, try to follow at least parts of these concepts to reduce learning time and errors.

Testing

The intention here is to test if the concept of the system matches the user's mental model. Discrepancies will appear when the user is confused or makes a mistake. You can use the "think aloud" method for non-working prototypes, and a testing session, including logging software, for a working prototype.

Skeleton Plane

Designing

Some things must be taken into account when you plan the design of the interface. Will the user use this system principally? Should the system fit into the operating system and collaborate with other programs? If the user works exclusively on that system, you can choose the elements that fit best. You have to face an extended learning phase, but it can speed up the workflow. When the user works with several programs at the same time, it is recommended to adapt the elements of the system in order to avoid misunderstandings and confusion.

Testing

Depending on the design, the test should follow the considerations above. If the system will be used together with other programs, it should be tested in a scenario with these programs. When you use your own design elements, give the subjects time to become familiar with the system and ask them explicitly for their opinions about the design.

Surface Plane

Designing

The visual design should not exclusively follow aesthetical issues. The design should reinforce the structure of the system and clarify the functions. Colors and fonts should not only enhance the recognition of a corporate identity but also improve readability.

Testing

At this point, a global evaluation should be performed. The scenario should be as real as possible, accounting for possible factors such as time pressure and noise. Try to construct tasks similar to the later work. If you need statistical significance, we recommend using logging software to collect data.

6.3.3 Scenario 3: User at a desktop computer, focus on data manipulation

This scenario emphasizes the interaction between the user and the system. Data manipulation means that the system will present data which the user can check and correct. The system has to react in an appropriate timeframe to give feedback to the user. Furthermore, the system should provide possibilities to visualize data in a way that is useful for editing operations.

Recommendations

Strategy Plane

Designing

You have to define the user groups and the tasks they will perform. In this scenario they will use the system to browse through data and change it. This means the system should provide features to ease the task. For example, you could think of additional hardware like a second screen; one display would be for visualization and one for editing data. A brainstorming session with the design team can generate a lot of possible, interesting features.

Testing

To find out about the usefulness of the features, you can create personas and use the cognitive walkthrough method to determine whether people would use the features. List the features in a questionnaire to be distributed amongst possible user groups. Another way is to watch people performing similar tasks with an existing system and ask them what they think about your ideas.

Scope Plane

Designing

In the scope plane you have to decide what features will be implemented according to the results of your survey in the strategy plane. A discussion within the development team about these results can help determine their feasibility.

Testing

To validate your decisions it is helpful to use the cognitive walkthrough method. With a persona in mind the development team can try to go through the steps of a task and ask, whether the feature will be used adequately to achieve their goal.

Structure Plane

Designing

After you have defined the requirements, you need to know how to structure these features. You have to create a concept that includes the features and defines how they will work. If you can fall back on a concept that already exists, it will simplify users' understanding. If possible, try to follow at least parts of these concepts to reduce learning time and errors.

Testing

The intention here is to test if the concept of the system matches the user's mental model. Discrepancies will appear when the user is confused or makes a mistake.

You can use the “think aloud” method for non-working prototypes, and a testing session, including logging software, for a working prototype.

Skeleton Plane

Designing

Some things must be considered when you plan the design of the interface. Will the user use this system principally? Should the system fit into the operating system and collaborate with other programs? When the user works exclusively on that system, you can choose the elements that fit best. You have to face an extended learning phase, but it can speed up the workflow afterwards. When the user works with several programs at the same time, it is recommended to adapt the elements of the system in order to avoid misunderstandings and confusion.

Testing

Depending on the design, the test should follow the considerations above. If the system will be used together with other programs, it should be tested in a scenario with these programs. When you use your own design elements, give the subjects time to become familiar with the system and ask them explicitly for their opinions about the design.

Surface Plane

Designing

The visual design should not exclusively follow aesthetical issues. The design should reinforce the structure of the system and clarify the functions. Colors and fonts should not only enhance the recognition of a corporate identity but also improve readability.

Testing

At this point, a final evaluation should be performed. The scenario should be as real as possible accounting for possible factors such as time pressure and noise levels. Try to build tasks similar to the later work. If you need statistical significance, we recommend using logging software to collect data.

6.3.4 Scenario 4: User with a mobile device, focus on data visualization

This scenario emphasizes data visualization for a user with a mobile device.

A mobile scenario can be influenced by numerous factors (for example, lighting and noise level can change rapidly). Normally, the user performs several tasks simultaneously, for example, telephoning while walking. She has to switch her attention between the tasks permanently and at the right time. Due to the small size of the device, input and output modalities are restricted. Compared to a desktop system, the processing power of mobile devices is relatively low.

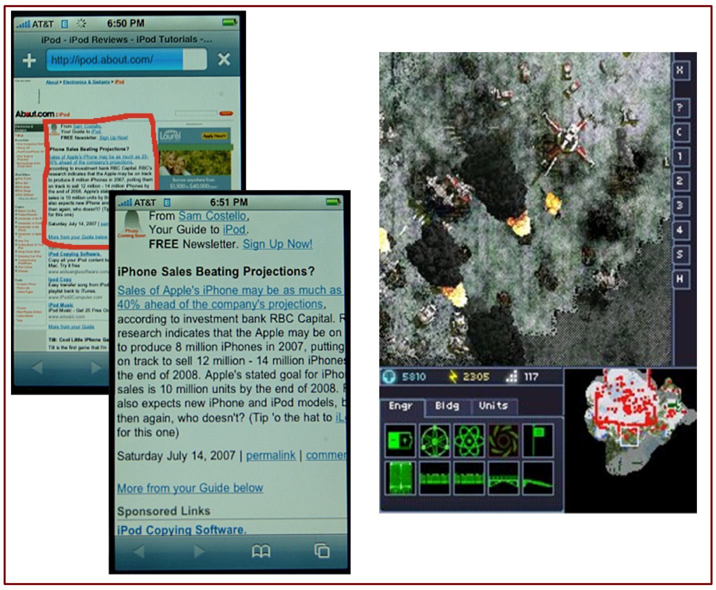
Recommendations

Strategy Plane

Designing

When considering the strategy plane, you need to know what the user will get out of the system. Defining the users and their needs is the first step in the design process. Creating personas that will represent a special user group can be useful. A persona is a fictional character constructed to represent the needs of a whole range of real users. With a persona in mind it is easier to estimate the importance or usefulness of certain features.

In this scenario, it means (a) that you must understand how the user will address the search request to the system, (b) how the data should be visualized and (c) which tasks the user performs simultaneously. The way the user will input the search request can depend on the situation (for example, with a lot of people around, it can be too noisy to use speech input). An additional consideration is that (d) walking in traffic when using the touch screen can be dangerous. (e) The visualization of data on a small screen is another challenge. The amount of data to present can be a problem, as well as the user preferences. Some people understand relationships between data better when presented graphically. Other people prefer textual descriptions, for example, "A is part of B." (f) Most users like to see the big picture, so it is useful to provide either a zoomable interface (Figure 14 left) or extra windows with a radar view (Figure 14 right). Figure 14 (on the left) shows the Safari Internet browser on the iPhone. You can zoom into a website by tapping on the screen. On the right, you can see a screenshot of a game running on Windows Mobile. In the lower right corner is a radar window that shows the whole area. Within this window is a rectangle that shows the position of the section displayed in the upper part of the screen.



From Sam Costello,
Your Guide to iPod.
FREE Newsletter. Sign Up Now!

iPhone Sales Beating Projections?
Sales of Apple's iPhone may be as much as
40% ahead of the company's projections,
according to investment bank RBC Capital. R
research indicates that the Apple may be on
to produce 8 million iPhones in 2007, putting
on track to sell 12 million - 14 million iPhone
the end of 2008. Apple's stated goal for iPho
sales is 10 million units by the end of 2008. R
also expects new iPhone and iPod models, b
then again, who doesn't? (Tip 'o the hat to iL
for this one)

Saturday July 14, 2007 | permalink | commen

More from your Guide below

Sponsored Links
iPod Copying Software.

Figure 14: (Left) Safari Internet Browser on an Apple iPhone; (Right) Windows Mobile Game

Testing

To test the design of the strategy plane, you should to ask potential users how satisfied they are with the features that you provide with the system. This is best done with a questionnaire. List the features that the system will include and let the user estimate their usefulness on a scale. Also, you can ask questions like, "Given situation A, B, C, etc., where would you prefer speech input? Why?" This will provide you with the users' concerns. In this phase of the design, it may be useful to reach as many (potential) users as possible. An online questionnaire is therefore advisable, which can be sent as a URL in an email. On the other hand, you must ask yourself if you can even implement the features the users want. The best way to find out about this is in the course of a group discussion within the developing team.

Scope Plane

Designing

You have to decide what kind of input modality will be implemented according to the results of your user survey and your team discussion. Will people use the different types of input as you expect or do you have to change your concept? Do users prefer a zoomable interface for data browsing or do they prefer the design with a radar window?

Testing

Using the cognitive walkthrough method allows you to decide whether certain input methods are feasible or not. This may apply to the visualization method as well. You have to bear in mind the situation in which the system is used. Mobile systems should allow users to perform a second task simultaneously. The cognitive load must be divided between two tasks without neglecting the situation.

Structure Plane

Designing

Are there already concepts in the real world that can be applied to your system? Can you define a concept that is easy to understand for the user? In this case, it means that speech input may work like an ASR system on a telephone. People who worked with an ASR system can imagine how to operate your system.

Testing

By using the "think aloud" method you can validate the concept of your system. While using the prototype, user speaks her thoughts about the system and what she is doing out loud. This can be challenging in a mobile scenario, but you have to be sure that the user understands the scenario and experiences the situation given by the test instructions. There can be varied situations when using a mobile device. For example, it may be noisy or very bright. This all will affect the way in which users interact with the system. You will see how the user operates the system in the given environment and if there are difficulties.

Skeleton Plane

Designing

When it comes to the interface design you can follow one of two different strategies. First, you can use the standard elements of the operating system (for example, Windows Mobile) because users are used to them and they appear as a standard "look & feel" in many applications. Second, you can design interface elements of your own; your system will run on different devices and the user can use elements she is familiar with according to the well-known (portable) interface. The second strategy is more useful, because portability can be achieved much easier. In addition, using the same widgets on all devices leads to consistency within the system. This is more important than consistency with an operating system. Navigation design on a mobile system is more challenging. For example, if the navigation bar on a website is on the left side, there are no problems when using a mouse. But a touch screen can be obstructive in this case for right-handers. It should be possible for the user to choose how to navigate.

Testing

Working prototypes are a great advantage when testing these issues. But to reduce testing time and effort, it is possible to use mock-ups. You have to make sure that the scenario is as real as possible in order to receive valid results. It is not advisable to test a mobile system with a desktop setup when designing the interface.

Surface Plane

Designing

The visual design for a mobile device is critical because of the limited screen size. On the one hand, you want to provide as much information as possible, but on the other hand, you have to be certain that information is easy to read and navigation elements are faultless to use.

Testing

When developing a system prototype for a mobile scenario, it is often advisable to use an eye-tracker for evaluation. An eye tracker is a device for measuring eye positions and eye movements. It tracks the point of gaze ("where we are looking") and the motion of an eye relative to the head. The most popular variant uses video images from which the eye position is extracted. This is important in order to understand where cognitive load comes from. Since it allows us to see where vision is concentrated, we can find out if and when the user has to switch attention to other areas. For example, if a person is driving, their visual focus should be on the road. If she wants to change the radio station, her visual area will move away from the road and she will not concentrate all of her attention on driving anymore.

6.3.5 Scenario 5: User with a mobile device, focus on data manipulation

Because of the limitations of a mobile device, technical implementations can be difficult to put into action. You have to find a way to visualize data that is appropriate for editing. Furthermore, you have to think of ways to edit data without a physical keyboard. In addition, the limited processing power of such devices raises the question, how to distribute the system into a client-server application to keep the response time of the system short.

Recommendations

Strategy Plane

Designing

You have to define user groups and tasks they will perform. In this scenario they will use the system to edit presented data. This means the system should provide features that ease the task. Discussing these issues within the development team can generate some interesting features.

Testing

To find out about the usefulness of the features, you can create personas and use the cognitive walkthrough method to determine whether people would use the features. Another way is to list features in a questionnaire for distribution amongst possible user groups. Furthermore, you can watch people performing similar tasks with an existing system and ask them what they think about your ideas.

Scope Plane

Designing

In the scope plane you have to decide what features will be implemented according to the results of your survey in the strategy plane. A discussion within the development team about these results can help determine their feasibility.

Testing

To validate your decisions it is helpful to use the cognitive walkthrough method. With a persona in mind, the development team can try to go through the steps of a task and ask whether people will use the feature adequately to achieve the goal.

Structure Plane

Designing

After you have defined the requirements, you need to know how to structure these features. You have to create a concept that includes the features and defines how they will work. If you can fall back on a concept that already exists it will simplify users' understanding. If possible, try to follow at least parts of these concepts to reduce learning time and errors.

Testing

The intention here is to test if the concept of the system matches the user's mental model. Discrepancies will appear when the user is confused or makes mistakes. You can use the "think aloud" method for non-working prototypes, and a testing session, including logging software, for a working prototype. There can be varied situations when using a mobile device. For example, it might be noisy or very bright. This will all affect the way in which users interact with the system. You will see how the user operates the system in the given environment and if there are difficulties.

Skeleton Plane

Designing

When designing an interface (**Interaction Design**) you can differentiate between two different strategies. First, you can use the standard elements of the operating system (for example Windows Mobile) because users are used to them; it employs a standard "look & feel" as the known interaction widgets and metaphors appear in most of the applications. Second, you can design interface elements of your own; your system will run on different devices and the user can use elements she is familiar with according to the well-known (portable) interface. The second strategy is more useful, because portability can be achieved much easier while retaining the corporate design. (Please note, the CogTool described earlier can be used to import corporate design for buttons, texts, etc. This means you can use the widgets users are familiar with even at the skeleton prototyping phase.) Furthermore, using the same widgets in all platforms leads to better consistency within the system. **Navigation Design** on a mobile system is challenging, too. For example, the navigation bar on a website is usually on the left side. When using a mouse, there are no problems, but a touch screen as input device can be obstructive for right-handers because of screen occlusion. It should be possible, however, for the user to choose how to navigate.

Testing

Working prototypes are a great advantage when testing these issues. But to reduce the time and effort, it is possible to use non-working prototypes. You have to make sure that the scenario is as real as possible in order to receive valid results. It is not advisable to test a mobile system with a desktop setup when defining the interface design.

Surface Plane

Designing

The visual design should not follow aesthetical issues exclusively. The design should reinforce the structure of the system and clarify the functions. Colors and fonts should not only enhance the recognition of a corporate identity but also improve readability. Especially on a mobile device, the visual design can be challenging. On the one hand you want to provide as much information as possible, but on the other hand, you have to be certain that information is easy to read and navigation elements are faultless to use.

Testing

At this point an overall evaluation should be performed. The scenario should be as real as possible and account for, for example, the changing light and different noise levels. Try to construct tasks similar to the later work. If you need statistical significance we recommend using logging software to collect data. In addition to logging software, we propose the use of an eye-tracker. When the system is used in the real world, you can see if the user is looking at the device or somewhere else. This is important in order to understand where cognitive load comes from. Since it allows us to see where vision is concentrated, we can find out if and when the user has to switch attention to other areas. For example, if a person is driving, their visual focus should be on the road. If she wants to change the radio station, her visual area will move away from the road and she will not concentrate all of her attention on driving anymore.

6.3.6 Scenario 6: Administrator at a desktop computer, focus on data input

In this case we can assume that the environmental factors, for example, light and noise level, will stay relatively unchanged. Furthermore, the administrator will have a keyboard and mouse for data input and the processing power of the system is high. We can assume that administrators use the system more frequently than users. It is advisable to keep this in mind when developing the system, especially for user training:

> You can accept an extended learning time if this may accelerate the workflow later on.

Recommendations

Strategy Plane

Designing

In this scenario it is easy to define the user group, because we assume it will be an administrator by definition. The creation of an appropriate persona again is useful to keep her knowledge, habits, and preferences in mind. With this persona in mind, it is easier to estimate the importance of certain features. You can consider of different input modalities. Will people use speech input or do they prefer the keyboard because they are used to it? Speech input may be unfavorable if the user is not alone in the room since she will disturb other people.

Testing

To validate your decisions you have to ask potential users of the system. The created personas will help you find real users. The best way to get the information you need is by using a questionnaire. The users can rate the importance of features by answering questions, for example, "How valuable is speech input for you?" The easiest way to reach a lot of people is to use an online questionnaire. You may find potential users in discussion boards or by email. Another possibility is to watch administrators performing similar tasks with an existing system and ask them what they think about your ideas.

Scope Plane

Designing

In the scope plane you have to decide what features will be implemented according to the results of your survey in the strategy plane. A discussion within the development team about these results can help to determine the feasibility.

Testing

To validate your decisions it is helpful to use the cognitive walkthrough method. With a persona in mind you can try to go through the steps of a task and ask yourself if you will use the feature adequately to achieve your goal.

Structure Plane

Designing

After you have defined the requirements, you need to know how to structure these features. You have to create a concept that includes the features and defines how they will work. If you can fall back on a concept that already exists it will simplify the user's understanding. In this case you can use the concept of a word processor, but is not necessary because of the special user group. Because the members of this user group use the system professionally and regularly, they will readily learn new concepts if it will speed up their work.

Testing

The intention here is to test whether the concept of the system matches the user's mental model. Discrepancies will be evident when the user is confused or makes a mistake. You can use the "think aloud" method for non-working prototypes, and a testing session, including logging software, for a working prototype.

Skeleton Plane

Designing

Some things must be taken into account when you plan the design of the interface. Will the user use this system principally? Should the system fit into the operating system and collaborate with other programs? When the user works exclusively on that system, you can choose the elements that fit best. You have to face an extended learning phase, but it may speed up the workflow. If the user works with several programs at the same time, it is recommended to adapt the elements of the system in order to avoid misunderstandings and confusion.

Testing

Depending on the design, the test should follow the considerations above. If the system will be used together with other programs, it should be tested in a scenario with these programs. If you use your own design elements, give the subjects time to become familiar with the system and ask them explicitly for their opinions about the design.

Surface Plane

Designing

The visual design should not follow aesthetical issues exclusively. The design should reinforce the structure of the system and clarify the functions. Colors and fonts should not only enhance the recognition of a corporate identity but also improve readability.

Testing

At this point you should conduct a final evaluation. The scenario should be as real as possible and consider, for example, time pressure and noise. Try to develop tasks similar to the later work. If you need statistical significance we recommend using logging software to collect data.

6.3.7 Scenario 7: Administrator at a desktop computer, focus on data manipulation

This scenario emphasizes the interaction between an administrator and the system. We can assume that administrators use the system more frequently than users. It is advisable to keep this in mind when developing the system. You can accept an extended learning time if you accelerate the workflow. Data manipulation means that the system will present data which the administrator can check and correct. The system has to react in an appropriate timeframe to give feedback to the administrator. Furthermore, the system should provide possibilities to visualize data in a way that is useful for editing and provide appropriate tools for manipulation.

Recommendations

Strategy Plane

Designing

In this scenario it is easy to define the user group, because we assume it will be an administrator by definition. The creation of appropriate personas is useful to keep their knowledge, habits, and preferences in mind. In this scenario administrators will use the system to control and edit data. This means that the system should provide features to ease the task. For example, additional hardware like a second screen may be useful; one display would be for visualization and one for editing data. A discussion with the development team can generate interesting features.

Testing

To find out about the usefulness of the features, you can construct personas and use the cognitive walkthrough method to determine whether people would use the features. List the features in a questionnaire and distribute it to possible user groups. Another way is to watch people performing similar tasks with an existing system and ask them what they think about your ideas.

Scope Plane

Designing

In the scope plane you have to decide what features will be implemented according to the results of your survey in the strategy plane. A discussion within the development team about these results can help to determine the feasibility.

Testing

To validate your decisions it is helpful to use the cognitive walkthrough method. With a persona in mind the development team can go through the steps of a task and ask if people will use the feature adequately to achieve their goal.

Structure Plane

Designing

After you have defined the requirements, you need to know how to structure these features. You have to create a concept that includes the features and defines how they will work. If you can fall back on a concept that already exists it will simplify users' understanding. This scenario combines at least two activities and you can think of a combination of two or more concepts. You can combine the concept of a word processor and a browser, but is not necessary because of the special user group.

Users are willing to learn new concepts if it speeds up their work and when the system is used professionally and regularly.

Testing

The intention here is to test whether the concept of the system matches the user's mental model. Discrepancies will be evident when the user is confused or makes a mistake. You can use the “think aloud” method for non-working prototypes, and a testing session, including logging software, for a working prototype.

Skeleton Plane

Designing

Some things should be considered when you plan the design of the interface. Will the user use this system principally? Should the system fit into the operating system and collaborate with other programs? If the user works on that system exclusively, you can choose the elements that fit best. You have to face an extended learning phase, but it may speed up the workflow afterwards. If the user works with several programs at the same time, customized workspaces are recommended to in order to avoid misunderstandings and confusion.

Testing

Depending on the design, the test should follow the considerations above. If the system will be used together with other programs, it should be tested in a scenario with these programs. If you use your own design elements, give the subjects time to become familiar with the system and ask them explicitly for their opinions about the design.

Surface Plane

Designing

The visual design should not exclusively follow aesthetical issues. The design should reinforce the structure of the system and clarify the functions. Colors and fonts should not only enhance the recognition of a corporate identity but also improve readability.

Testing

This is when you should perform a global evaluation. The scenario should be as real as possible and consider, for example, time pressure and noise. Try to develop tasks similar to the later work. If you need statistical significance we recommend using logging software to collect data.

7 Usability Testing in the TEXO Use Case

We finalized the prototype of a mobile client application for the iPhone that allows a user to intuitively search for services on the TEXO platform. Using the ODP framework and leveraging the device's capabilities, the user can interact with the system in a multimodal fashion.

7.1 Strategy Plane

We started by choosing a suitable platform. We examined the official THESEUS Program web page on three mobile browsers (mobile Internet Explorer, mobile Opera, Minimo) with default settings on Microsoft's Windows Mobile 6 platform. They are completely different from each other and, furthermore, from the original appearance. Thus, displaying large web pages which make use of CSS on these three browsers is not feasible since not all browsers are capable of visualizing them as intended.

When we consider other mobile platforms like Symbian S60, Google's Android or Apple's iPhone, the quality of visualization increases. But although these platforms use the same core HTML rendering engine (WebKit[1]), the behavior of initially displaying a web page differs. While the S60 and Android browsers show the upper left part of a page, the mobile Safari on the iPhone zooms out such that a page is fully displayed in its width (Figure 15).

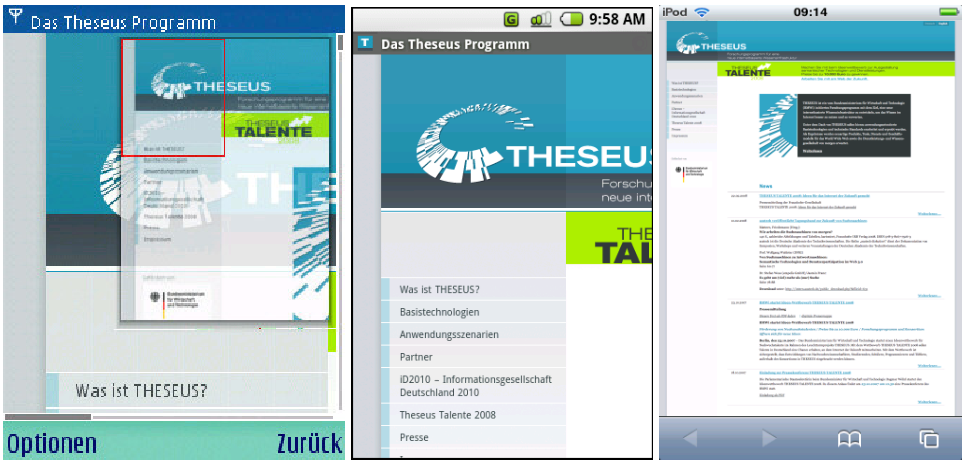


Figure 15: Rendering of the THESEUS Program web page in a resolution of 320x240 in a S60 device (left, with mini map) and in a resolution of 480x320 on Google Android (center) and Apple iPhone (right).

Browsing through web pages requires user interaction. Zooming on the iPhone is seamlessly done without visual interface element just by performing a multi-touch gesture on the screen. This contrasts other mobile browsers as displayed in Figure 16.

[1] See http://webkit.org/.

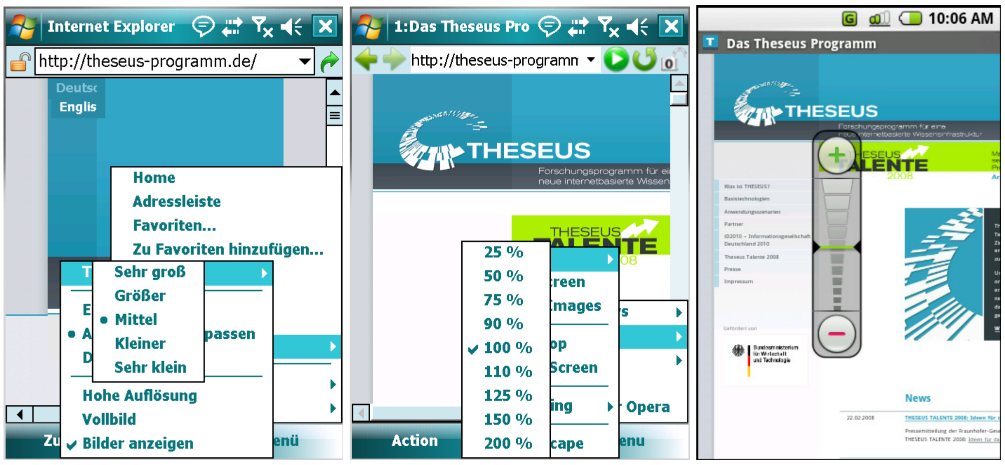


Figure 16: Zooming on mobile Internet Explorer (left), mobile Opera (center), and Android (right)

7.2 Scope Plane

We address the factors for *enjoyment* and *ease of use*, and our application is aligned with the Human Interface Guidelines of the target platform. However, there will also be a stationary browser-based client that does not need to follow platform-specific guidelines. So, a trade-off between both worlds (platform-specific Human Interface Guidelines vs. project-specific Human Interface Guidelines) is required (prioritizing).

Since mobile devices suffer from small screens, multimodal interaction including speech leads to reduced visual interfaces that save space. This space can then be consumed by relevant content that is displayed to the user according to her profile information. The Apple iPhone platform provides the best User Experience and sufficient hardware capabilities in order to meet the technical requirements. In our opinion, this also justifies slightly higher development effort on this platform.

Regarding the *usefulness* of the application, we proposed a feasible extension to the existing TEXO "EcoCalculator" scenario that requires mobile access to the TEXO service ecosystem. It takes place in the Design Phase of the Product Life-cycle and in the Service Offering/Consumption Phase of the Service Life-cycle. The extended **storyline** adopts results of the Methods Taskforce and starts when the designer David intends to use a service in his Product Life-cycle Management (PLM)/Computer Aided Design (CAD) system that he discovered on the TEXO Platform:

1. For several reasons (e.g., too expensive), he may not buy the service on his own.
2. So, David must ask his superior Susan for confirmation. The system determines the correct receiver of the formal confirmation request.
3. This request is sent as an email, containing a URL as entry point to the company's TEXO-enabled Enterprise Resource Planning (ERP) System where only a few employees (in this case only Susan) are allowed to finally buy products/services.
4. However, Susan is currently away on business. But she always carries her TEXO-enabled mobile handset with her.
5. Susan's company uses a Microsoft Exchange Server, so the above-mentioned email is automatically and instantly sent to Susan's mobile inbox.

6. Susan reads the email and clicks on the link (e.g. similar to a cFolders-Link) to the TEXO-enabled ERP-System.
7. This action automatically opens the mobile TEXO client since the mobile interface to the company's TEXO enabled ERP-System is embedded into the client as a view.
8. After login, the link redirects her to the ERP section of the mobile TEXO client as starting point.
9. There, she…
    - …is presented with all of the relevant information about the requested service(s) from a business perspective (who initiated the buy, rating, community feedback, key performance indicators, licensing, price, etc.)
    - …can remotely buy the service so that David is not impeded in his work. This has to be done explicitly in the TEXO-enabled ERP-System itself since the service can only be bought from Susan's account.
    - …can browse through related services and propose another one to David for discussion. This again is not possible in the form of a simple email, but is especially useful if Susan is not fully satisfied with one of the parameters of David's proposed service.
    - …can interact with the system in the most appropriate way depending on the current situation, e.g., using speech, she can define/refine filters for the service search which replaces lots of typing on a mobile device.
10. Finally, Susan is convinced and buys the service which is then put into the company-wide "service repository" as long as the license is valid. There, all licensed services of the company are listed.
11. David, as the person who initiated the buy, is notified and can now integrate the service from the repository into his PLM-System for usage (the PLM-System must only be able to access the repository).

The exact action sequence Susan is taking in steps 10 and 11 of the cognitive walkthrough to complete the task is listed in the table below. When "walking through" each step of the action sequence, we check our observations with the gathered requirements. As a result, we gain a revised, finer grained set of requirements by identifying potential pitfalls and also develop preliminary design ideas that will flow into the further development.

No.	Action	System feedback
1	Go to main menu (SpringBoard)	A grid of icons which are linked to applications is shown; the email icon is extended by showing the number of new messages arrived.
2	Tap the icon of the email application	The email application is opened.
3	Find an unread message header referring to "request for confirmation" and touch on it	The body of the message is opened; besides some text it contains a link to an external application which is visually emphasized.
4	Read the message and follow the link to the ERP-System	The mobile TEXO client application is opened and completely fills the screen; the client shows the ERP-System-View with the appropriate ticket "request for confirmation" opened and presents its content as kind of a form on the screen. Among other things, there is a link available in order to get more

		detailed information about the service. The TEXO client also provides a permanently visible navigation bar which allows switching between the different components, e.g., ERP-module, service inspection, service search. Part of this bar is a specific PushToTalk-button to activate a speech interaction with the system.
5	Tap the "more information"-link of the purchase item described by the ticket form	The client switches to the DetailedInformation-View which fills the complete screen. A structured document is presented. It provides an inspection of the service description according to different facets and additionally a thumbnail image of the visual appearance of the service GUI is shown; the user can enlarge the image on demand by double-tap on it.
6	Press the StandBy-button of the iPhone to interrupt the current work	Screen turns black.
7		The phone rings and the phone application comes into foreground; some specific buttons are shown
8	Tap on the Accept-button	Not of interest for usability testing.
9	Tap on the HangUp-button	The TEXO client application is shown; the state is unchanged.
10	Tap on the ERP-Symbol on the navigation bar	The ERP-System-View is shown w.r.t. the ticket in focus.
11	Tap the "more information"-link of the purchase item described by the ticket form	The client switches to the DetailedInformation-View which fills the complete screen. A structured document is presented. It provides an inspection of the service description according to different facets and additionally a thumbnail image of the visual appearance of the service GUI is shown; the user can enlarge the image on demand by double-tap on it.
12	Perform VerticalWipe gestures to find the information about CommunityRating	Screen scrolls vertically according to user's interaction.
13	Tap on the ServiceSearch-item of the navigation bar	The TEXO client switches to the search view and automatically takes the current service description as input to query for alternatives. A ranked result list of services each represented by a short summary of its description is presented.
14	Tap on the PTT-button to activate speech interaction	The button changes its color from red to green symbolizing an open microphone.
15	Utter a speech command like "please order the results according to community rating"	The client shows some activity indication until the request is processed and then the result list is reordered. The Push-To-Talk-button (PTT) changes its color form green to red symbolizing a closed microphone.
16	Tap on the PTT-button to activate speech input	The button changes its color from red to green symbolizing an open microphone.

17	Utter a speech command like “please show me services which support a pay-per-use accounting”	The client shows some activity indication until the request is processed and then the result list is filtered accordingly. The PTT-button changes its color from green to red symbolizing a closed microphone.
18	Tap on the ERP-Symbol on the navigation bar	The ERP-System-View is shown w.r.t. the ticket in focus.
19	Select the Confirm-button	The button is highlighted.
20	Tap on the CloseTicket-button	The view is closed and the main screen of the ERP-System is shown.

Table 2: Exact Action Sequence Required to Complete the Task in TEXO

7.3 Structure Plane

Using PowerPoint mockups, we rapidly designed the first structure and skeleton(s) of the mobile client’s user interface according to the identified requirements and the results of the cognitive walkthrough. We further explored how navigation and interaction can intuitively be realized by creating a more sophisticated mock-up with the CogTool[2] that supports basic interaction by means of navigating from one screen to another.

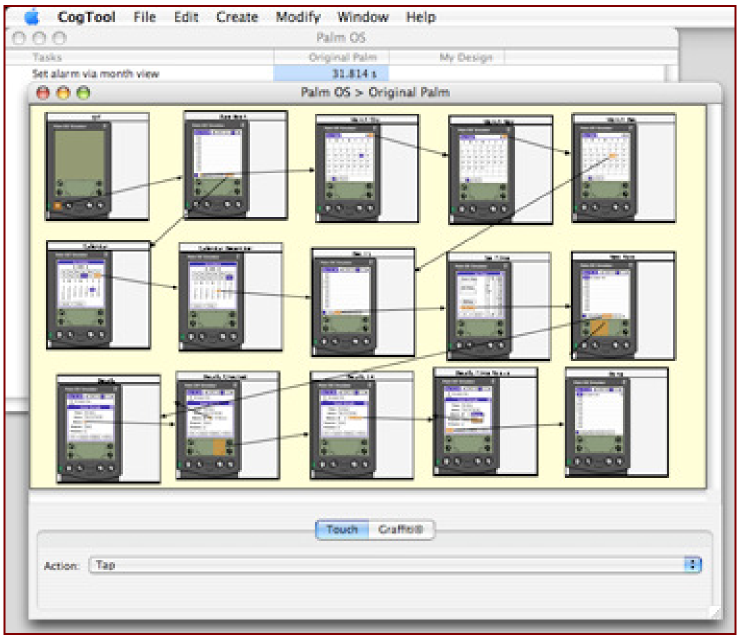


Figure 17: Editing a Design Storyboard in CogTool

In this context, we employed a graphical template for Adobe’s Photoshop (an updated version can be found on Teehanlax[3]) as well as the IUI-framework[4] for the

[2] See http://cogtool.hcii.cs.cmu.edu/ or section 3.2.6. for more information.
[3] See http://www.teehanlax.com/blog/?p=447.
[4] See http://code.google.com/p/iui/.

rapid development of HTML/AJAX-based iPhone-like web prototypes which imitate the actual look-and-feel of basic iPhone applications quite well.

Of course, simulating a mobile interface on a desktop PC and using a mouse is far away from being a realistic mobile scenario, but the built-in analysis features of the CogTool helped us to verify (or falsify) our general approach.

Finally, we decided to offer three components that are aligned with the scenario:

- The Enterprise Resource Planning (ERP) interface shows a list of ERP tickets and is able to show a detailed view of each ticket in the list.
- The Service Description (SD) component shows, as the name already implies, the service description of a distinct service
- The Search component visualizes search results and actually consists of two separate visualizations:
  - a list-based visualization of search results
  - and an OpenGL-based three-dimensional visualization of search results

While the ERP and the Search component have to be directly accessible by the user, showing a service description only makes sense in the context of an ERP ticket or after a search has been performed. Therefore, the SD component is accessible through the ERP or Search component, not directly.

On top of these application-specific components, we developed the so-called Multimodal Interaction Controller as an abstraction layer to the Ontology-based Dialogue Platform that is developed and adapted to TEXO in CTC-WP4. In the following, we will discuss the overall system architecture as a result from the decisions we made so far.

7.3.1 The Ontology-based Dialogue Platform

The Ontology-based Dialogue Platform (ODP) is a domain-independent, generic framework which greatly facilitates the development of multimodal dialogue systems. ODP defines both a generic modeling framework and run-time environment for multimodal dialogue applications supporting advanced dialogue interaction.

The run-time environment of ODP is based on a flexible middleware platform which connects system components following a hub-and-spoke architecture. Dialogue processing is accomplished by means of an ontology-based production rule system. The system components of ODP are completely implemented in the Java programming language and thus operable on a number of system platforms.

Conceptually, ODP's architecture comprises a number of functional components that deal with tasks like modality-specific interpretation, context-based interpretation, interaction and task management, target control, presentation management and modality-specific generation. All functional components are generally application-independent and are configured by respective models.

7.3.2 Client Architecture

The mobile client consists of several modules as depicted in Figure 18. The Multimodal Interaction Controller is an abstraction layer for the actual client application modules. It takes care of the connection to and the interaction with the ODP server application. To do so, the Handshake Channel initializes a new session by requesting and establishing ports for the required communication channels.

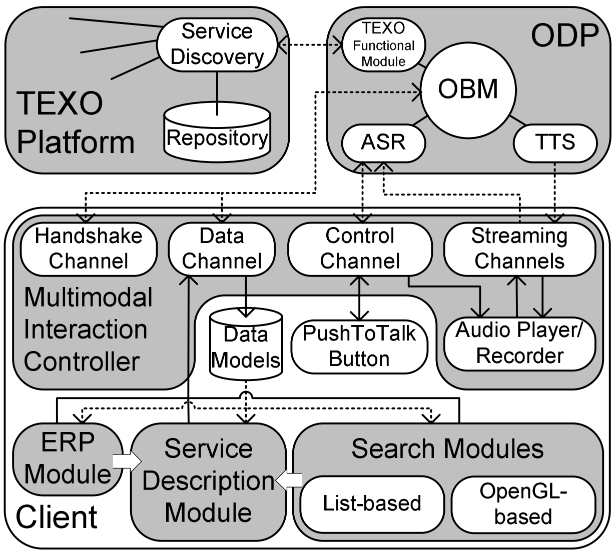


Figure 18: Overall system architecture. The mobile client is connected with the ODP server which in turn can communicate with the TEXO backend.

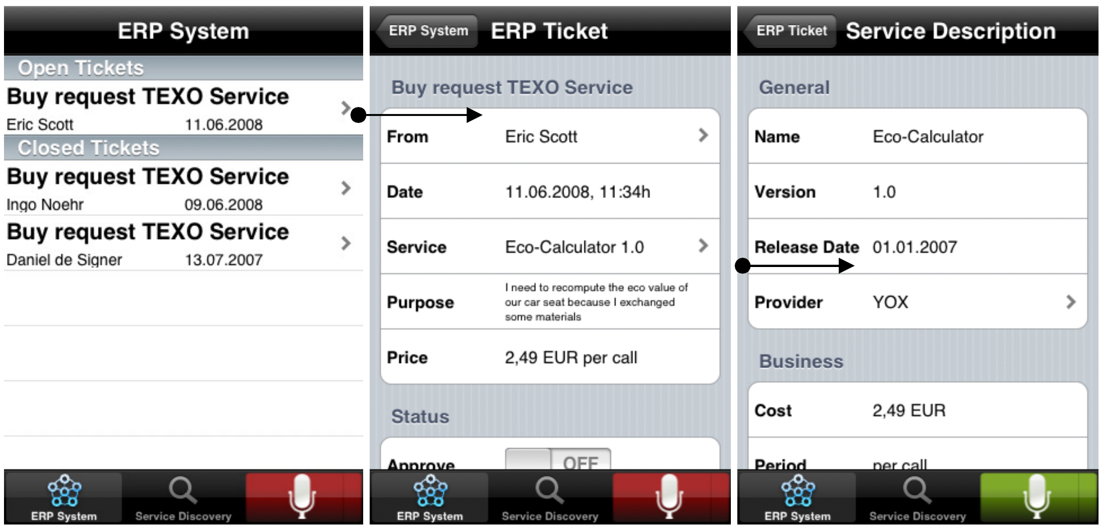
All communication from the client to the TEXO backend is done via the ODP which, for this purpose, utilizes a dedicated functional module that communicates with the Service Discovery component. The service repository of the TEXO platform contains all registered services, annotated with all kinds of meta-data (e.g., quality of service, business and legal aspects, community feedback, etc.).

7.4 Skeleton Plane

The skeleton plane describes how a single view of the application is arranged. Besides the actual content of a view, each view contains a navigation bar on top of the screen that shows the name of the current view and optionally holds a back button. Each view also contains a tab bar at the bottom of the screen that holds buttons that allow the current view to be changed as well as the push-to-activate button that is required for speech interaction. The push-to-activate button is not visible initially but slides into the tab bar using built-in animations when the client has successfully established a connection to the ODP server. We believe that this deviation[5] from Apple's human interface guidelines for iPhone applications does not confuse users because the push-to-activate button is colored differently, making it distinguishable from standard tab-bar buttons. This became necessary since the device's hardware buttons are not accessible for developers and we needed a constant position for this button.

7.4.1 Sample Interaction Sequence

When launching the client application, an overview of recent ERP tickets is given (Figure 19, left). The push-to-activate button slides into the bottom tab-bar. Tapping on a ticket provides the user with details of the ticket. In this case, the Eco-Calculator service is requested (Figure 19, center). Tapping on the service entry shows detailed information about the selected service in the Service Description module (Figure 19, right). To check for alternatives, the mobile user can initiate a respective search by tapping the push-to-activate button and uttering "Show alternative services," in which case the user implicitly refers to the service currently in focus.



Open Tickets

Buy request TEXO Service
Eric Scott 11.06.2008

Closed Tickets

Buy request TEXO Service
Ingo Noehr 09.06.2008

Buy request TEXO Service
Daniel de Signer 13.07.2007

Buy request TEXO Service

From	Eric Scott
Date	11.06.2008, 11:34h
Service	Eco-Calculator 1.0
Purpose	I need to recompute the eco value of our car seat because I exchanged some materials
Price	2,49 EUR per call

Status

Approve	OFF

General

Name	Eco-Calculator
Version	1.0
Release Date	01.01.2007
Provider	YOX

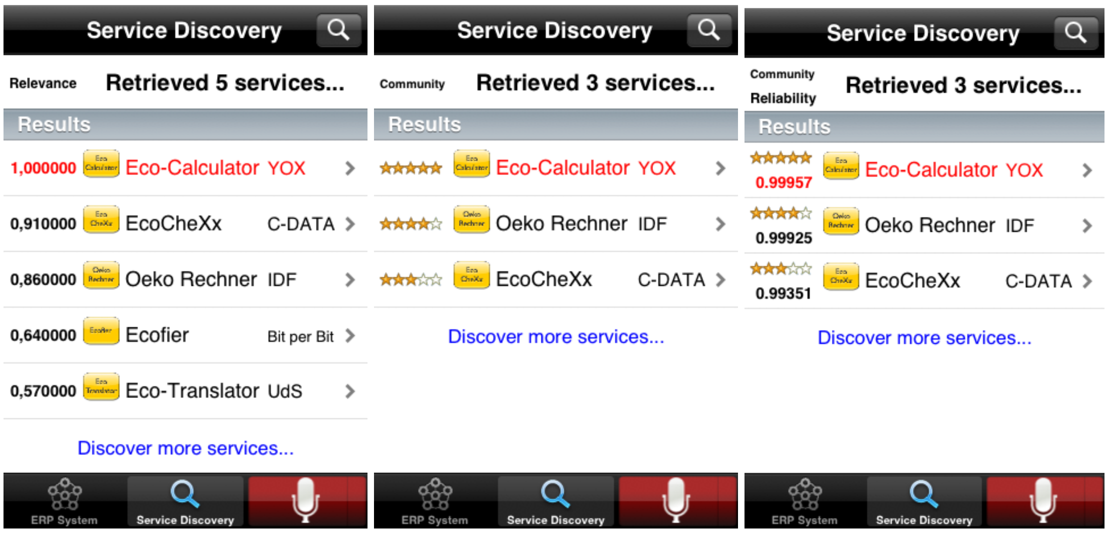
Business

Cost	2,49 EUR
Period	per call

Figure 19: Multimodal Interaction Sequence, ERP System

The ODP server recognizes the cross modal context of this request, uses the currently displayed service as seed and generates a multimodal output for the device, containing a (potentially large) list of functional equivalent services that it retrieved from the TEXO Service Discovery backend (Figure 20, left) as well as a corresponding speech synthesis. The user can now re-sort the list according to another service property and apply a filter to the new order simultaneously. To do so he utters "Which of the services have a better rating than 2 (stars)?" The filtered result list is displayed to the user (see Figure 20, center) accompanied by a corresponding speech output.

[5] Normally, buttons in a tab-bar only switch between different views of an application.


Service Discovery

Relevance Retrieved 5 services...

Results

1,000000 Eco-Calculator YOX

0,910000 EcoCheXx C-DATA

0,860000 Oeko Rechner IDF

0,640000 Ecofier Bit per Bit

0,570000 Eco-Translator UdS

Discover more services...

Service Discovery

Community Retrieved 3 services...

Results

★★★★★ Eco-Calculator YOX

★★★★☆ Oeko Rechner IDF

★★★☆☆ EcoCheXx C-DATA

Discover more services...

Service Discovery

Community Reliability Retrieved 3 services...

Results

★★★★★ 0.99957 Eco-Calculator YOX

★★★★☆ 0.99925 Oeko Rechner IDF

★★★☆☆ 0.99351 EcoCheXx C-DATA

Discover more services...

Figure 20: Multimodal Interaction Sequence, Service Discovery

Many tedious steps are omitted which usually require the result lists to be re-sorted and filtered. In a mobile context it is especially desirable to reduce the cognitive load of a user.

Since the domain-specific part of the ontology employed by the ODP server comprises knowledge about the preferred order of the different service properties, the best services according to a given property are always on top of the list (e.g., reliability is ordered descending while response time is ordered ascending).

When rotating the device to the landscape orientation or when applying two service properties as sort parameters (e.g., "Sort the list by community rating and reliability."), the ODP server notifies the client to initially switch from the list-based to the OpenGL-based visualization of search results. In this example, the device is rotated. In the visualization which appears, the services are lined up from left to right. To zoom and move the environment, common (multi-touch) gestures are utilized.

To visually re-sort the service items, the user can (as an alternative to speech) drag a property icon from the left over a (highlighted) axis and drop it there (Figure 21, left).

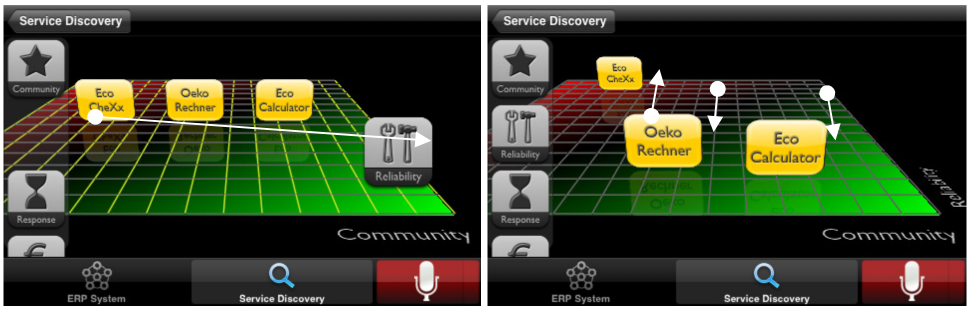


Figure 21: Multimodal Interaction Sequence, 3D Visualization

The ODP server notifies the event and changes the context appropriately. Subsequently, the items are animated to their new positions (Figure 21, right), such that the user is able to track the changes. If another property was already assigned to the selected axis, it is replaced. Items are arranged relatively to each other, allowing for a constant floor size. Since this visualization is intended to provide a fast orientation of the best services according to the specified sort parameters, we intentionally omit absolute values and instead position “good” services with respect to the applied sort properties always in lower right corner. This prevents “good” services from being occluded by poorer ones and ensures that a user can always find the best services (with respect to the applied properties) at the same position. Details about a service can be retrieved in this view by tapping the respective item. Then, after an animated camera movement towards the item, the Service Description module shows its content in a transparent popup view allowing the user to always recognize the background context. The view is discarded by slightly shaking the device, leveraging the built-in accelerometers.

By rotating the device again to portrait orientation, the list-based visualization is updated and the additionally applied sort property is employed (Figure 20, right). In the current scenario, the user can now switch back to the ERP System view and, if finally convinced, approve the buy.

7.5 Surface Plane

Concerning the Surface Plane which tries to communicate the brand identity by consistently using the same color palettes and typography, we are currently almost completely tied to the iPhone platform, since it provides a developer with already predefined visual styles: default blue, black translucent, and black opaque. We decided for the black opaque style.

Furthermore, we additionally applied a gradient floor coloring in the three dimensional visualization, meaning that “bad” services are positioned on the red part of the floor near the upper left corner whereas “good” are positioned in the opposite green corner.

7.6 Results of Usability Testing Process

We tested this issue in 12 business-related subtasks. Eleven participants were recruited from a set of 50 people who responded to our request (only that fraction was found suitable). Those selected were all students (most of them business and economics majors). Six of them were male, five female and they were partly acquainted with mobile phones. Four of them owned an iPhone/iPod touch. After five minutes of free exploration time with the application, and additional hints from the instructor, users had two attempts to successfully perform a task. For the second attempt, the instructor was allowed to give assistance in terms of clarifying the purpose of the task and the possible input. (However, if the instructor told someone exactly what to do, the subtask could not be regarded as performed successfully.)

From our analysis of the questionnaires, we conclude that our mobile B2B system can be valuable for the business users. Almost all users did not only successfully complete the subtasks (89% of a total of 132 subtasks), but many of them also provided positive feedback that they felt confident about the ticket purchase being successful. This means the "power test users" who are knowledgeable of the domain, were able to use the mobile interface for the domain-specific task. We also achieved high query recognition accuracy (> 90%) for the spoken queries that can be recognized (i.e., utterances that are modelled in the speech recognizer grammar).

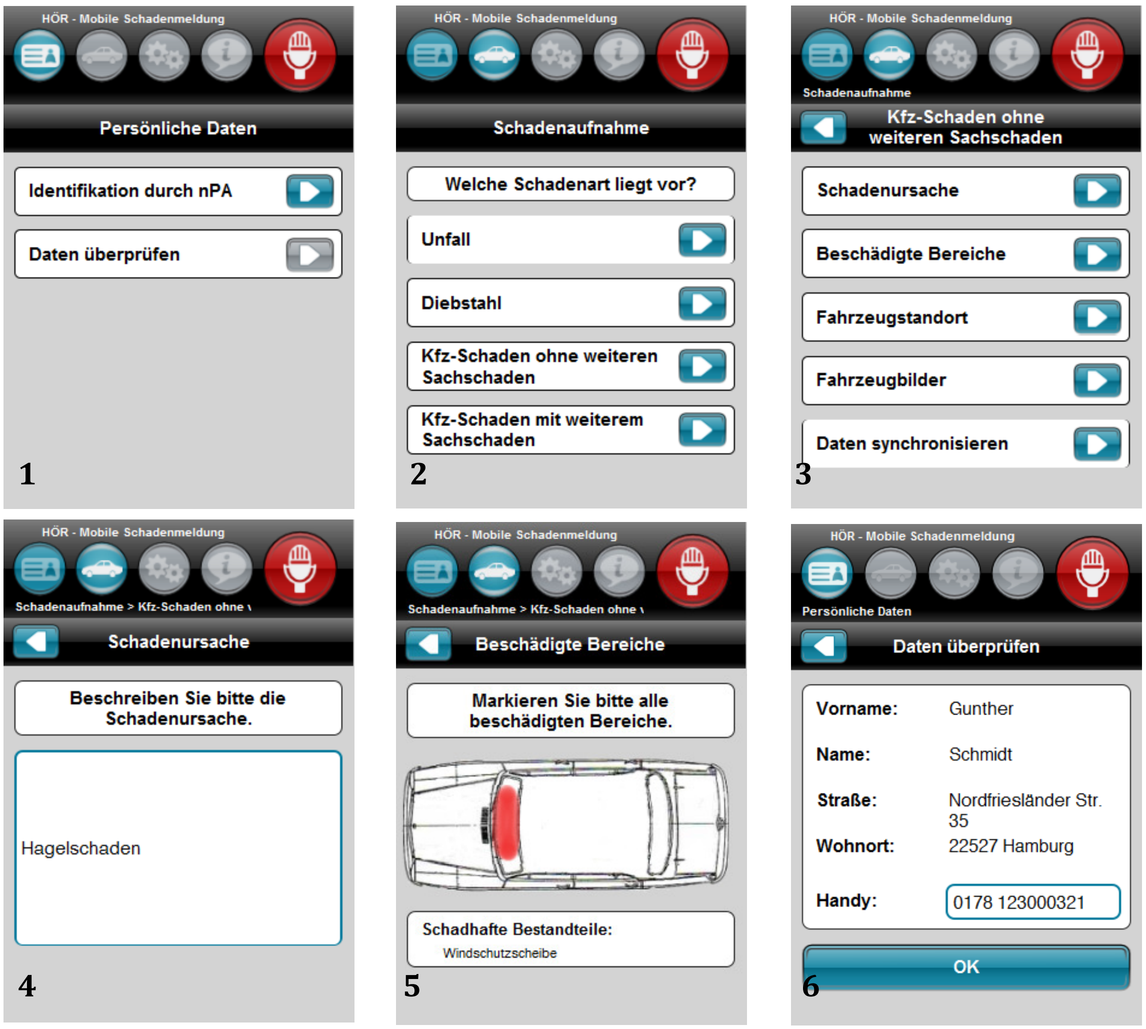
The current implementation has its limitations, though. Despite the obvious advantage of supporting speech input, flexible language input remains a challenge. All users reported difficulties when it came to finding appropriate speech commands before they got used to this input modality. In addition, it was rather unclear in which situations a speech command could be used. In the context of the 3-D visualization, eight users reported problems with the drag-and-drop functionality on the touchscreen. Often, users (7 of 11) were not able to capture the sense of multiple sorting criteria; many users reported that the icons were just too small. In contrast, the list representation of the service descriptions was perceived very intuitively while discovering different services. Here, we were able to display a proper description of the background services under investigation. This is due to the fact that every new service is properly described (semantic web services) in a semi-automatic fashion when added to the service repository. In this way, we avoid the problem of displaying search surrogates on the mobile screen.

7.7 Design for a Second Demonstrator

Based on the introduction of a new scenario in the TEXO use case, which is based on car insurance services, we recently developed a browser-based demonstrator for multimodal mobile claims notifications. Typically, claims notifications adhere to a strict business process specified by the insurance provider. In our scenario (mobile device and data visualization), the insurance provider requires a policy holder to pass through the following process steps in order to successfully submit a claims notification (Figure 22 shows the respective screenshots). All screens display a process view on top. It indicates the steps to follow during the claim notification process.

Figure 22, (1): This screen in the mobile claims notification application (*HOER: Mobile Schadenmeldung*) allows the user to input and check personal information (*Persoenliche Daten*). He can be identified through an identification number (nPA).

Figure 22, (2): The next screen "Damage Appraisal" (*Schadenaufnahme*) asks the users which type of claim they which to make. The options are: accident, theft, vehicle damage without further property damage, vehicle damage with further property damage. Please note that "Damage Appraisal" (*Schadenaufnahme*) appears under the round blue icons and above the back button and helps show the user where he is in the process.


1
HÖR - Mobile Schadenmeldung
Persönliche Daten
Identifikation durch nPA
Daten überprüfen

2
HÖR - Mobile Schadenmeldung
Schadenaufnahme
Welche Schadenart liegt vor?
Unfall
Diebstahl
Kfz-Schaden ohne weiteren Sachschaden
Kfz-Schaden mit weiterem Sachschaden

3
HÖR - Mobile Schadenmeldung
Schadenaufnahme
Kfz-Schaden ohne weiteren Sachschaden
Schadenursache
Beschädigte Bereiche
Fahrzeugstandort
Fahrzeugbilder
Daten synchronisieren

4
HÖR - Mobile Schadenmeldung
Schadenaufnahme > Kfz-Schaden ohne w
Schadenursache
Beschreiben Sie bitte die Schadenursache.
Hagelschaden

5
HÖR - Mobile Schadenmeldung
Schadenaufnahme > Kfz-Schaden ohne w
Beschädigte Bereiche
Markieren Sie bitte alle beschädigten Bereiche.
Schadhafte Bestandteile:
Windschutzscheibe

6
HÖR - Mobile Schadenmeldung
Persönliche Daten
Daten überprüfen

Vorname:	Gunther
Name:	Schmidt
Straße:	Nordfriesländer Str. 35
Wohnort:	22527 Hamburg
Handy:	0178 123000321

OK

Figure 22: Interaction Sequence authentication, validation of contact information, specification of the kind of damage, the cause of damage, damaged car parts, the location of the incident and images.

Figure 22, (3). The user has selected the third option, vehicle damage without further property damage, and a list appears of additional required information. From top to bottom these are: cause of damage, damaged area(s), vehicle location, pictures of the vehicle, and synchronize data.

Figure 22, (4). The user begins at the top and decides to give a cause of damage. The system requests, “Please describe the cause of damage.” The user can now describe the damage via natural speech, and the dialogue system extracts “Damage by hail” from his utterance. The user clicks on the back button to return to the option screen and chooses the next option, damaged area(s).

Figure 22, (5). In the screen for damaged areas, the system asks the user to please mark all damaged areas. He marks the windshield which appears on the screen in red. Underneath of the picture of a vehicle, a list is created of damaged areas (*Schadhafte Bestandteile*): windshield (*Windschutzscheibe*).

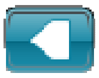
Figure 22, (6). In this final screenshot, the user wants to verify his personal information. His first name (Gunther), last name (Schmidt), street (Nordfriedlaender Str. 35), city and zipcode (22527 Hamburg), and cell phone number (0178 123000321) are displayed. The user may click on the “OK” button to verify the information or select any of the categories to change them.

The visual representation of complex business processes requires sophisticated navigational concepts. Thus, based on our usability analysis and obtained results for the first demonstrator (cf. Section 7.4.1), we implemented a set of patterns in order to guide a user in the learning phase of the second application. First, we followed the usability guidelines as demonstrated for the first demonstrator. Then we identified the requirements for a user who is familiar with the first demonstrator and implemented the planes accordingly (the details of this implementation are left out here). The most interesting step in the second development process is the identification of the set of patterns in order to guide a user in the learning phase of the second application. The patterns are as follows:

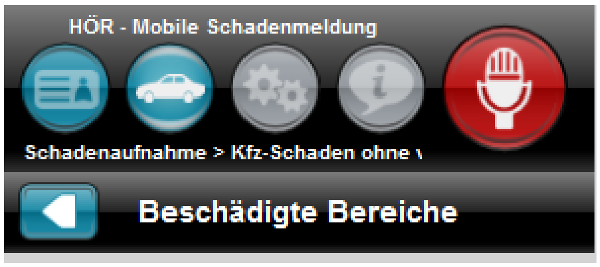
- Macro process steps are represented as tabs in the upper tab bar. They are always visible (green and gray round buttons).

- Each macro step consists of a set of micro steps that are initially displayed in the respective tab pane. Entering a micro step, the content of this step is shown in the tab pane (also cf. Figure 3 about hierarchical tasks).

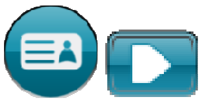
- The user can navigate back by clicking the back button.



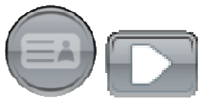
- So as not to get lost, the complete navigation hierarchy is displayed below the tab bar, between the back button and the navigational blue and gray buttons.


HÖR - Mobile Schadenmeldung
Schadenaufnahme > Kfz-Schaden ohne \
Beschädigte Bereiche

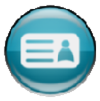
- Enabled steps, i.e., steps whose preconditions are met, are represented by a colored icon. Preconditions are met when the user has supplied the system with the necessary information, completed the necessary steps, or decided to follow a particular process.



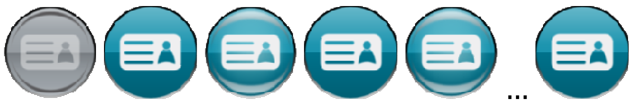
- Disabled steps, i.e., steps whose preconditions are not yet met, are represented by a gray icon. They may not be selected.



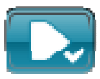
- The active macro step that the user is currently in is indicated by a highlighted icon.



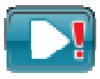
- Macro steps that become enabled change their visual representation from a gray to a colored icon and start to blink until the user enters this step. This is a signal to the user that his attention/interaction is required.



- Micro steps that have already been visited by the user are equipped with a check mark. This prevents a user from unnecessarily revisiting a step a second time.



- An exclamation mark indicates a micro step that contains new information which has not been noticed by the user yet and requires her attention.



8 Usability Testing in the MEDICO Use Case

We finalized a prototype of a speech based client application that allows a medical user to search for medical images in the radiology department while using the MEDICO platform. In the following, we will describe our usability considerations and the clinical workflow and interaction requirements. We will also focus on the design and implementation of a multimodal user interface for patient/image search or annotation and its implementation while using a speech-based dialogue shell.

Ontology structures are the basis of communication for our combined semantic search and retrieval architecture which includes the MEDICO server, the triple store, the semantic search API, the medical visualization toolkit MITK, and the speech-based dialogue shell, amongst others. We will focus on usability aspects of multimodal applications, our storyboard, and the implemented speech and touchscreen interaction design. The demonstrator implements multiple usability scenarios.

- Scenario 1: User at a desktop computer, focus on data input
- Scenario 2: User at a desktop computer, focus on data visualization
- Scenario 3: User at a desktop computer, focus on data manipulation

Using the ODP framework and leveraging the interaction device's capabilities, the user can interact with the system in a multimodal fashion. Figure 23 shows basic requirements issues: (1) support and implement the radiologist's clinical requirements (contents and medical interactions) from the medical application scenario and (2) integrate the implemented dialogue system into the medical environment.

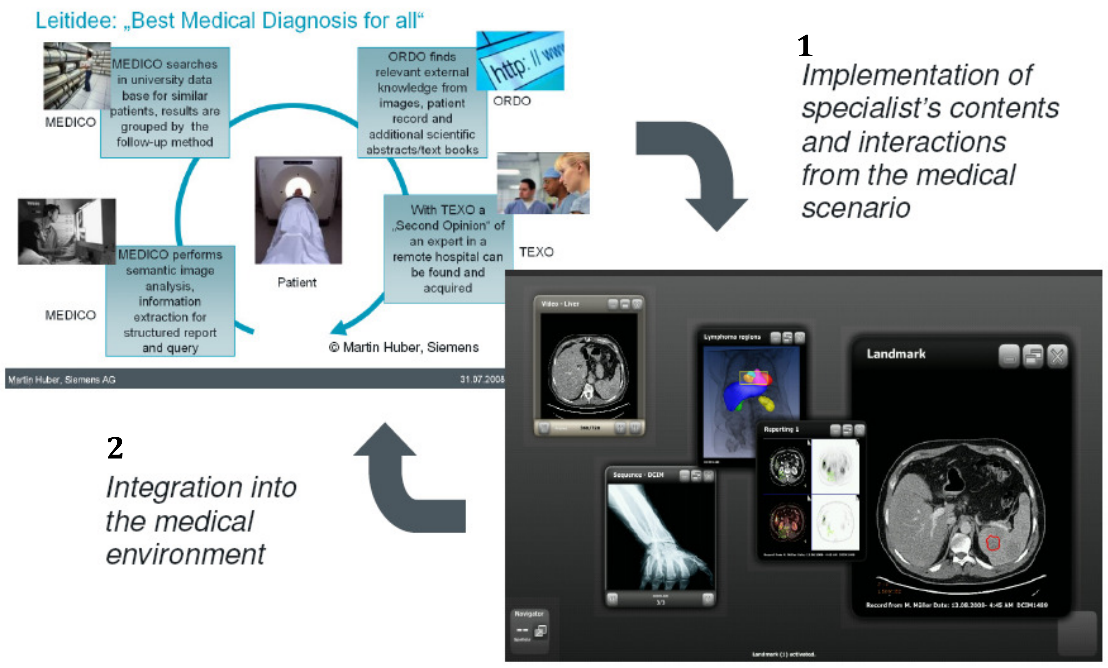

Leitidee: „Best Medical Diagnosis for all“
MEDICO searches in university data base for similar patients, results are grouped by the follow-up method
MEDICO
ORDO finds relevant external knowledge from images, patient record and additional scientific abstracts/text books
ORDO
http://www
With TEXO a „Second Opinion“ of an expert in a remote hospital can be found and acquired
TEXO
Patient
MEDICO performs semantic image analysis, information extraction for structured report and query
MEDICO
© Martin Huber, Siemens
Martin Huber, Siemens AG
31.07.2008
1
Implementation of specialist's contents and interactions from the medical scenario
2
Integration into the medical environment
Landmark

Figure 23: Basic Requirements Issues for MEDICO

8.1 Strategy Plane and Scope Plane

In order to specify the strategic and scope plane issues, we used several usability methods: *cognitive walkthrough*, *observation of the (medical) user*, and *hierarchical task analysis*.

We started by choosing a suitable platform. Since we have a direct connection to the MEDICO server platform (see Figure 24), we decided to use this as the implementation basis. We also examined the radiology department and provided a first storyboard for the interaction design. We named the final radiology speech system "RadSpeech"; here we describe the design and implementation stage of the prototype towards the implementation of the fully functional RadSpeech application. Towards this goal, the most important decision concerned the dialogue system setting and the interaction device.

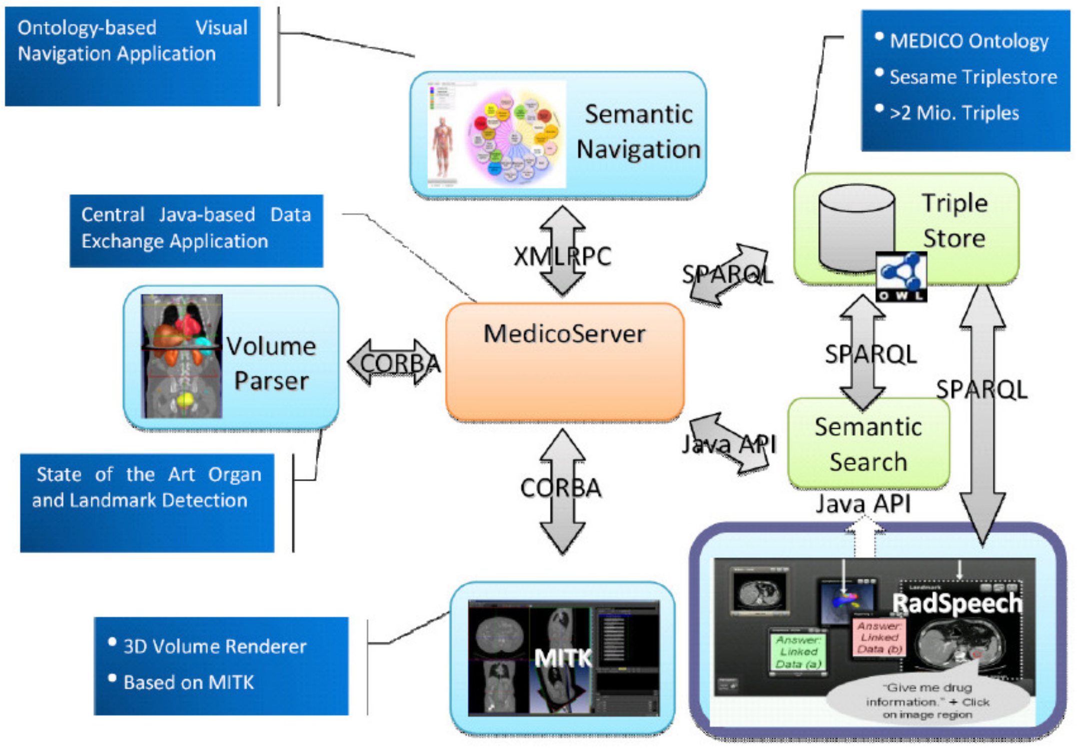


Figure 24: Connections to MEDICO Server

Figure 24 shows the overall architecture of MEDICO's approach for integrating manual and automatic image annotation. One of the main challenges was to integrate the C++ code for object recognition (left), the MITK-based image viewer and annotations tool (bottom; also in C++), and the Java-based components for knowledge base manipulation and semantic search (right). We came up with a distributed architecture with a CORBA (Common Object Requesting Broker Architecture) server as a mediator between our C++ and Java components.

The described prototype and the final RadSpeech application will extend the existing MEDICO architecture with a speech dialogue system that allows radiologists to annotate and search for medical images in a way that is more natural for them than standard interfaces.

For this purpose, we conducted a cognitive walkthrough usability test. This cognitive walkthrough started with a task analysis that specifies the sequence of steps or actions a user requires to accomplish a task, and the system's responses to those actions.

Simply visiting the users to **observe them work** was an extremely important usability method. The observer's goal is to become virtually invisible to the users so that they will perform their work and use the system the way they normally do.

Recently, structured reporting was introduced that allows radiologists to use predefined standardized forms for a limited but growing number of specific examinations. However, radiologists feel restricted by these standardized forms and fear a decrease in focus and eye dwell time on the images.

We visited the radiology department four times in 2008. A team of five to ten radiologists are working in such a department. After each observation session, we collected their feedback according to the new finding process (structured reporting) if there were no restrictions to the employed technology. After two visits, it became clear to us that a speech-based system would best fit this generally dark and quiet environment where the users very much focus on the image sequences.

Hierarchical task analysis (HTA) breaks down the steps of a radiologist's task as performed by a medical user and describes the task as seen at various levels of detail. Each step can be decomposed into lower-level sub-steps, thus forming a hierarchy of sub-tasks. We used this to specify the interaction system setting, the example dialogue for our speech-based dialogue system, and the interaction device.

To search and understand scalable and flexible semantic images, semantic labelling and the interlinking of the data of interest is required. This becomes technically possible when all semantic descriptions are stored in a knowledge base and efficiently linked to previous examinations of the same patient, patient records with a similar diagnosis or treatment, and/or external knowledge resources, such as publications, all of which are relevant in the context of the particular symptoms of the first diagnosis. Several approaches to the semantic annotation of medical images and radiology reports exist. All of these approaches are not only accomplished *offline* but are also quite time consuming and expensive due to the required user interaction.

We are concerned with answering the following questions:

- How can we enable the semantic annotation of patients' findings without interrupting the clinicians' workflow?
- How can we support the clinical daily tasks in a way that allows parallel semantic annotations of relevant clinical findings without additional efforts?

With traditional user interfaces, users may browse or explore visualized patient data, but little to no help is given when it comes to the interpretation of what is being displayed. Semantic annotations should provide the necessary image information, and a semantic dialogue system should be used to ask questions about the image annotations while engaging the clinician in a natural speech dialogue. This will be RadSpeech's motivation to design and implement a multimodal dialogue system for the radiologist. Dialogue-based semantic image retrieval should provide the basis for the help in clinical decision support and computer aided diagnosis.

Our proposed solution envisions an embedded dialogue system. The radiologist can still use the traditional desktop environment, but is able to select image regions on a second touchscreen (Figure 25) and annotate them via speech comments.

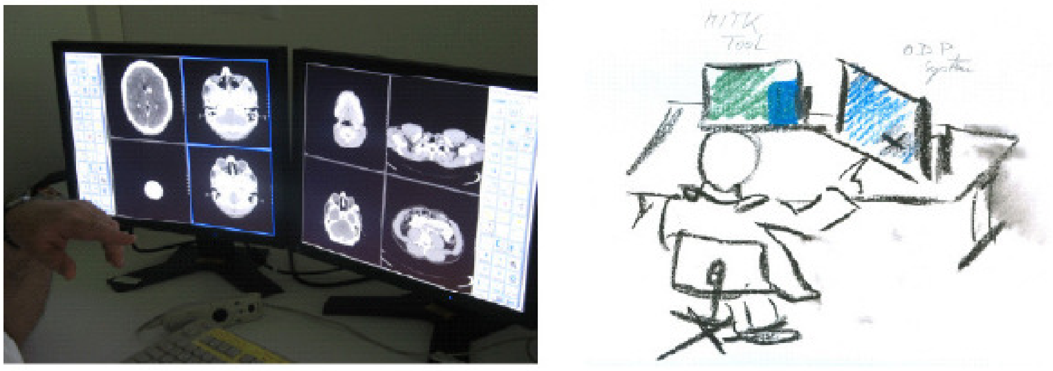


Figure 25: (Left) Traditional desktop environment. (Right) In the new environment, one of the monitors is now a touchscreen.

8.2 Structure Plane (and Detailed Storyboard)

Using PowerPoint mockups, we rapidly designed the first structure and skeleton(s) of the touchscreen installation user interface according to the identified requirements and the results of the cognitive walkthrough.

The design task for the structure plane consists of a cycle of action and reaction. Either the user acts and the system reacts or the other way around. Every time the user uses the dialogue system, she will improve her mental model of the system. But this only works if the conceptual model of the system matches the user's mental model. If the user can predict what the system will do, she is more willing to do trial and error. For this purpose, a storyboard is constructed and implemented by concrete SIEs (Semantic Interface Elements, see Sonntag et al., 2009). (Figure 26 shows an early drawing of the interaction sequences.) The realization of the RadSpeech implementation prototype can be derived from the motivation of the radiologist's daily task in the SIEMENS patient image finding stations as installed at the University Hospitals Erlangen: more efficiency during the medical finding process and more structured finding reports including semantic image annotations. For this purpose, we aim at the extension of the multimodal dialogue as provided by the first demonstrator which has been developed according to the usability test explained here. In order to produce a complete structural report in the context of a follow-up treatment of a lymphoma patient (also cf. prototype dialogue) in the real radiology environment, the RadSpeech Prestudy should allow a set of radiologists to comment on the requirements of the second RadSpeech prototype.

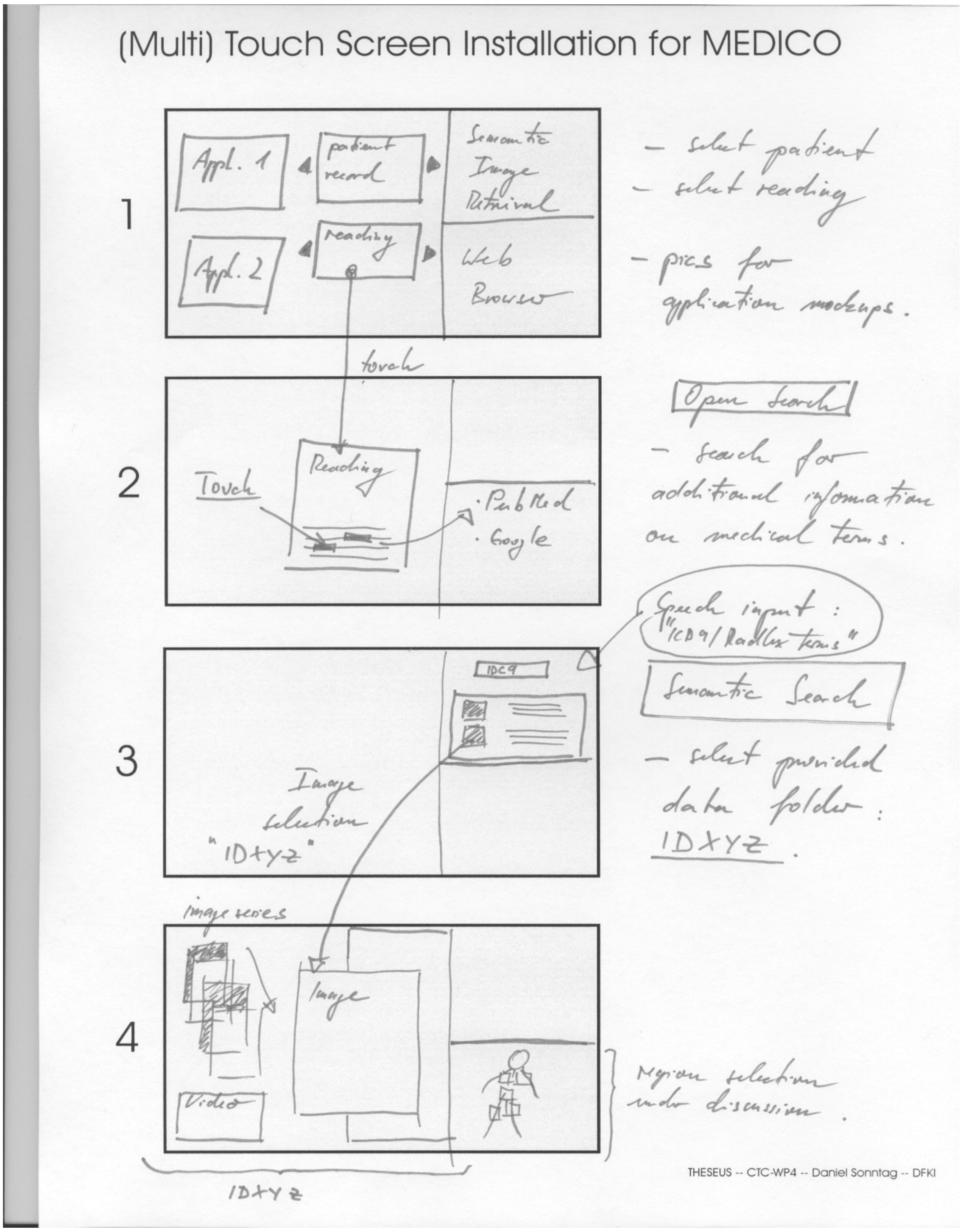
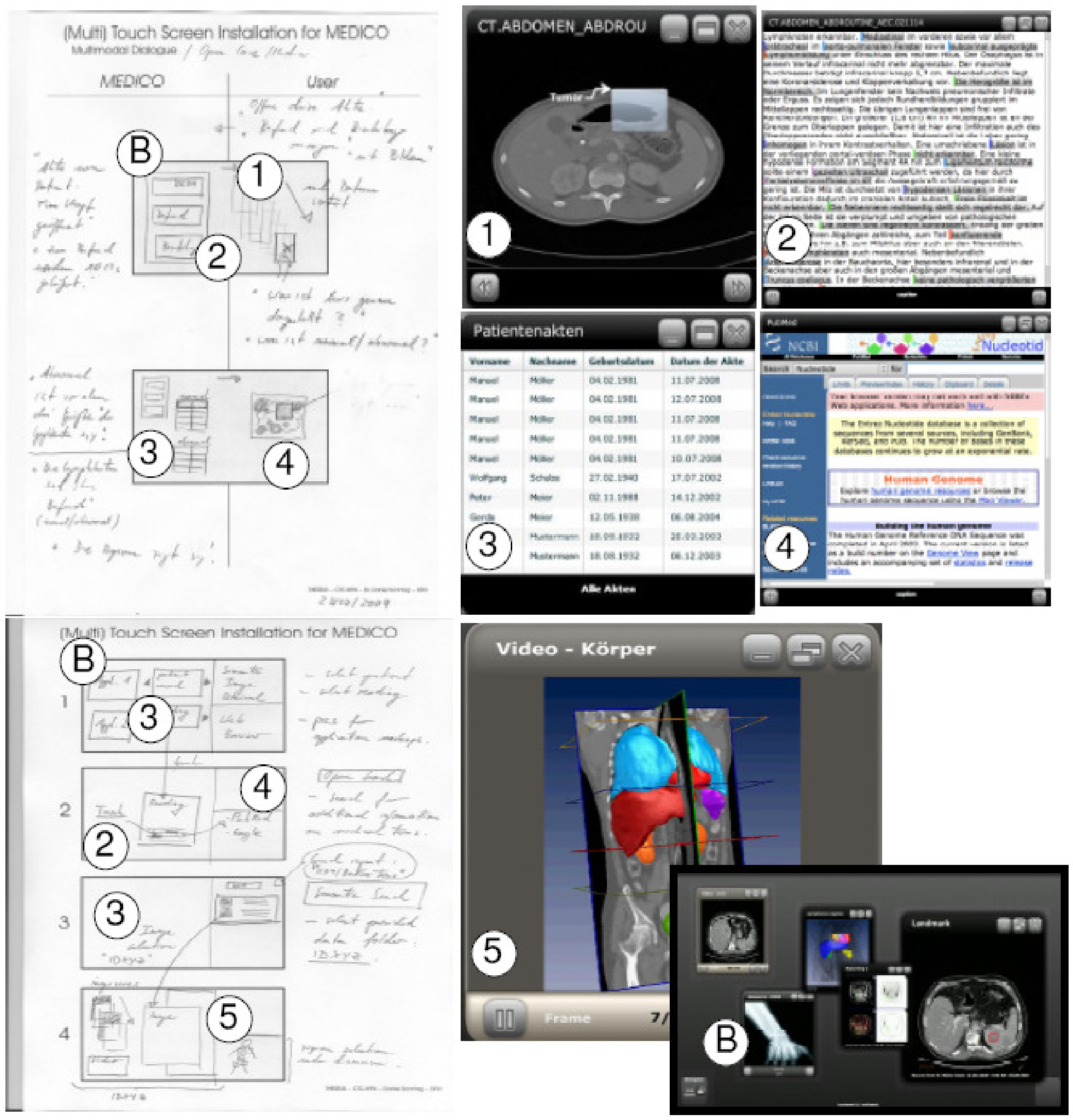

(Multi) Touch Screen Installation for MEDICO
1
Appl. 1
patient record
Semantic Image Retrieval
Appl. 2
reading
Web Browser
– select patient
– select reading
– pics for application mockups.
touch
2
Touch
Reading
· PubMed
· Google
Open Search
– search for additional information on medical terms.
Speech input: "ICD9/RadLex terms"
3
IDC9
Semantic Search
Image selection "ID+yz"
– select provided data folder: IDXYZ.
image series
4
Image
Video
region selection under discussion.
IDXYZ
THESEUS -- CTC-WP4 -- Daniel Sonntag -- DFKI

Figure 26: The Interaction Storyboard and the Included SIEs, i.e., Image Annotation SIE (1), Patient Finding SIE (2), Patient Search SIE (3), Browser SIE (4), and Video SIE (5)

Figure 27 shows the Image Annotation SIE (1), the Patient Finding SIE (2), the Patient Search SIE (3), the Browser SIE (4), and the Video SIE (5). The touchscreen background SIE is displayed in (B). The SIEs (1-5) represent the visual interaction elements for the graphical user interface. The implementation of the dialogical interaction sequences in the dialogue shell, and the reference dialogue, are based on these visual elements.



Vorname	Nachname	Geburtsdatum	Datum der Akte
Manuel	Möller	04.02.1981	11.07.2008
Manuel	Möller	04.02.1981	12.07.2008
Manuel	Möller	04.02.1981	11.07.2008
Manuel	Möller	04.02.1981	11.07.2008
Manuel	Möller	04.02.1981	10.07.2008
Wolfgang	Schulze	27.02.1940	17.07.2002
Peter	Meier	02.11.1988	14.12.2002
Gerda	Meier	12.05.1938	06.08.2004
[illegible]	Mustermann	18.09.1932	28.03.2003
[illegible]	Mustermann	18.08.1932	06.12.2003

Figure 27: Interaction Sequence after Implementation According to the Storyboard

On top of these application-specific components, we developed the so-called Multimodal Interaction Controller as an abstraction layer to the Ontology-based Dialogue Platform that is developed and adapted to MEDICO. In the following, we will discuss the dialogue system architecture as a result from the decisions we made so far.

8.2.1 The Ontology-based Dialogue Platform

The Ontology-based Dialogue Platform (ODP) is a domain-independent, generic framework which greatly facilitates the development of multimodal dialogue systems as needed for the implementation of the embedded radiology dialogue system. ODP defines both a generic modeling framework and a run-time environment for multimodal dialogue applications supporting advanced dialogue interaction.

The ODP framework[6] has been used to build prototype systems for other application scenarios besides MEDICO. TEXO Mobile (see Section 7: Usability Testing in the TEXO Use Case), developed within the THESEUS research program, provides a mobile, multi-modal interface for accessing business web services. It follows the implementation in Sonntag (2010). In the context of the MEDICO use case, we explain the dialogue platform in more detail because the touchscreen-based interaction uses all functionality the dialogue platform provides. This is illustrated in Figure 28.

Input and output components can be attached to the generic system. Such components include a speech recognizer (*ASR*) and a speech synthesis (*TTS*) module. Our approach relies on a flexible toolbox of generic and configurable dialogue shell building blocks. The exchange data between the different modules implemented upon the mentioned building blocks is based on ontology-based data using *extended Type Feature Structures* (eTFS) (Pfleger and Schehl, 2006).

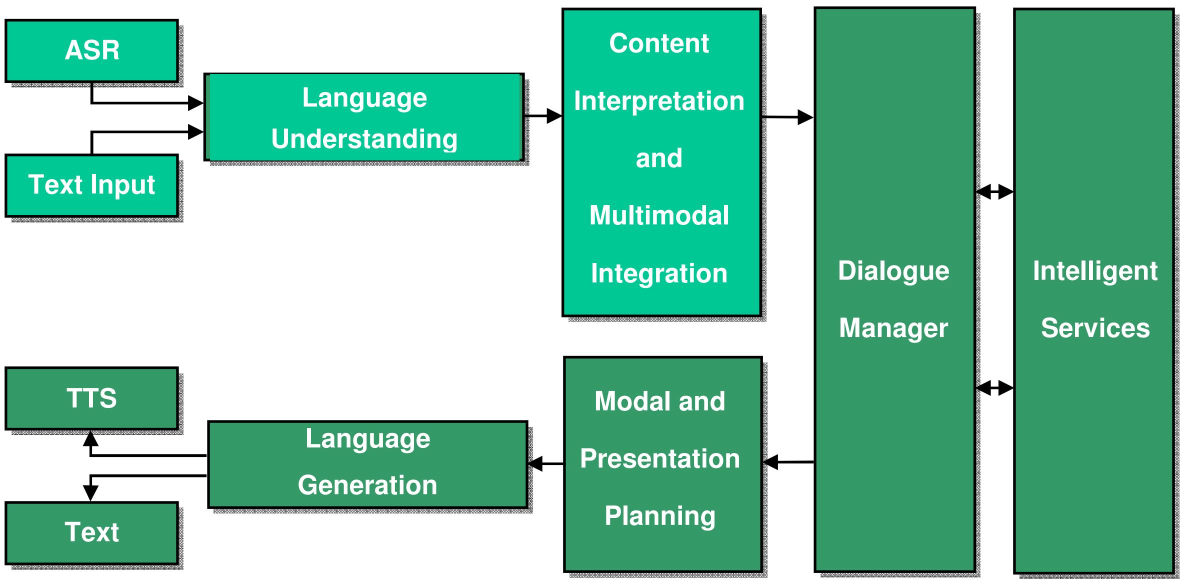


Figure 28: Architecture of a Multimodal Dialogue System

8.3 Skeleton Plane (Ontology Modeling and Interaction Sequence)

The information design on the skeleton plane is governed by the representational decisions we had to make in the context of the semantics of the contents of the images. This means the representational basis of the images is the medical ontologies (FMA, ICD-10, etc.) and these structures form the basis for any information exchanged between the user and the dialogue system.

[6] This ontology-based dialogue platform is available at http://www.semvox.de/.

As a result, we also used the medical ontologies and upper ontologies of the medical application domain (Figure 29) to generate the messages transferred between the dialogue system and the user. The resulting semantic annotations are in the format of the medical ontologies. In order to provide a translation between natural language expressions and these structures, a complex natural language understanding pipeline must be investigated. However, the ontology-based information design of the complete MEDICO search application can be accessed via a Java API and be used to implement the natural dialogue on a technical level.

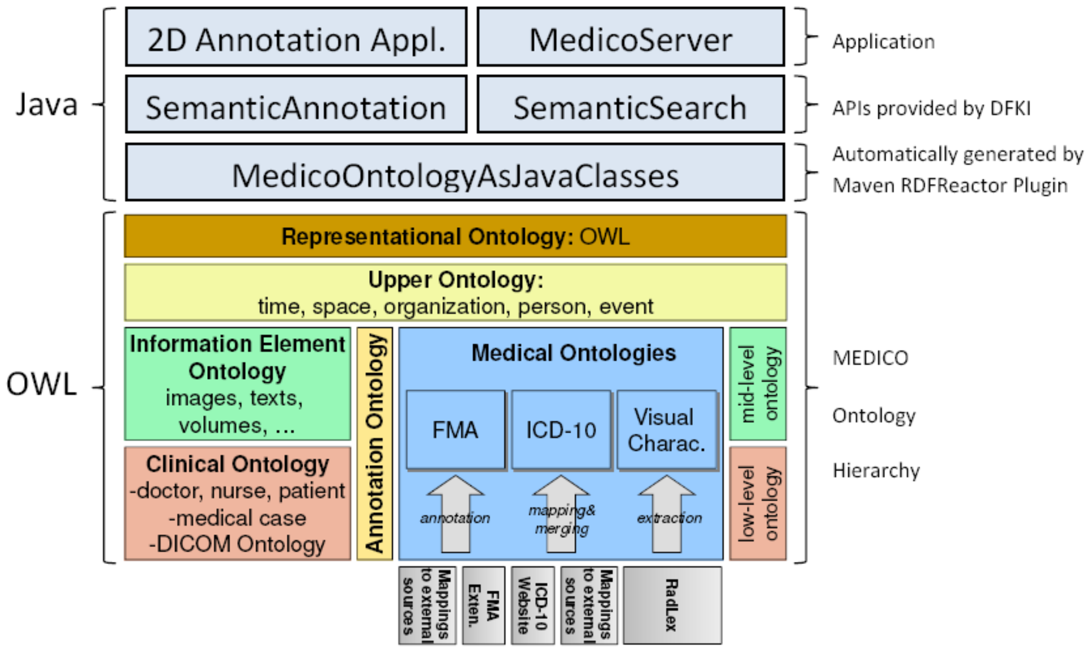

Java
2D Annotation Appl.
MedicoServer
Application
SemanticAnnotation
SemanticSearch
APIs provided by DFKI
MedicoOntologyAsJavaClasses
Automatically generated by Maven RDFReactor Plugin
OWL
Representational Ontology: OWL
Upper Ontology: time, space, organization, person, event
Information Element Ontology images, texts, volumes, …
Clinical Ontology -doctor, nurse, patient -medical case -DICOM Ontology
Annotation Ontology
Medical Ontologies
FMA
ICD-10
Visual Charac.
annotation
mapping& merging
extraction
mid-level ontology
low-level ontology
MEDICO
Ontology
Hierarchy
Mappings to external sources
FMA Exten.
ICD-10 Website
Mappings to external sources
RadLex

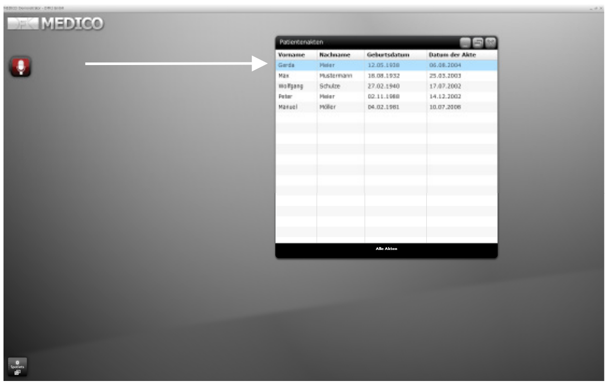
Figure 29: Medical Ontology and Upper Ontology Model

The resulting natural language interaction sequence is part of the skeleton plane. For convenience, we will explain it together with the surface plane. This way, the reader is able to follow not only the steps of the interaction sequence, but can also see how the interaction elements really look and how they are arranged.

8.4 Surface Plane (Implemented Interaction Sequence)

The surface plane tries to communicate the brand identity by consistently using the same color palettes and typography. In addition, we had further requirements in the radiology usage scenario. For example, the background cannot be bright and the usage of colors must be heavily restricted. The actual picture content is monochrome (black/white) and multicolored surface elements heavily distract from the radiology pictures.

This plane also deals with the logical arrangements of the design elements. In the case of a multimodal dialogue system, the logical arrangement results in a user-system natural dialogue whereby the user input is speech and touch and the system output is generated speech or the generation of SIEs which display windows for images, image regions, or other supported interaction elements. The implemented clinical workflow is best explained by example (Figure 30). Consider a radiologist (R) at his daily work of the *clinical reporting process* with the speech-based semantic dialogue shell (S):



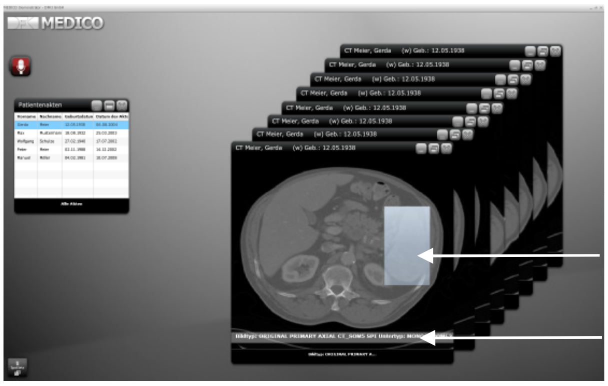
Vorname	Nachname	Geburtsdatum	Datum der Akte
Gerda	Meier	12.05.1938	06.08.2004
Max	Mustermann	18.08.1932	25.03.2003
Wolfgang	Schulze	27.02.1940	17.07.2002
Peter	Meier	02.11.1988	14.12.2002
Manuel	Möller	04.02.1981	10.07.2008

The potential application scenario (provided by Siemens AG) includes a radiologist which treats a lymphoma patient; the patient visits the doctor after chemotherapy for a follow-up CT examination.

R: *"Show me my patient records, lymphoma cases, for this week."*

S: Shows corresponding patient records.

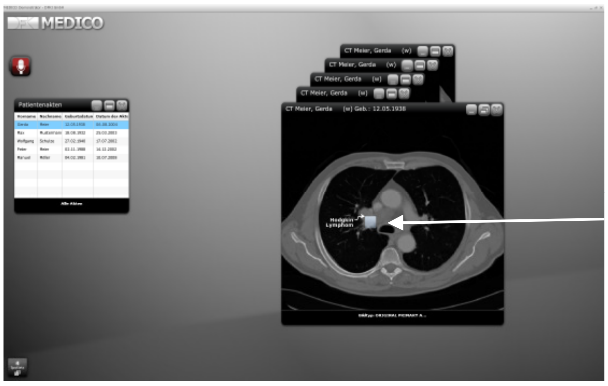
R: *"Open the images, internal organs: lungs, liver, then spleen and colon of this patient (+ pointing gesture (arrow))."*



S: Shows corresponding patient image data according to referral record.

The presentation planer of the dialogue system rearranges the semantic interface elements (SIEs). The top-most picture frame, showing the patient information in the header, is interactive; when touching it, special image regions and region annotations are highlighted (two arrows).

R: Switches to the 5th image and clicks on a specific region (automatically determined).

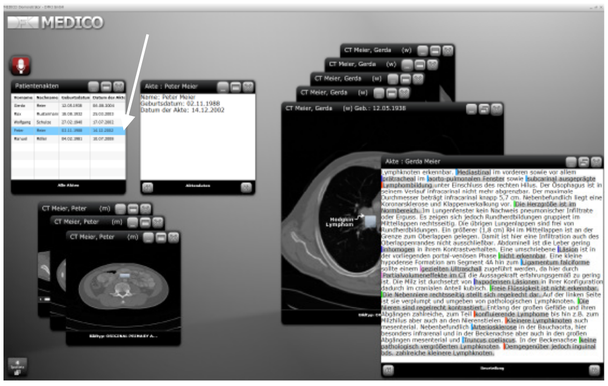


S: The system rearranges the semantic interface elements (SIEs) to signalize that the dialogue focus is on regions.

R: *"This lymph node here (+ pointing gesture), annotate Hodgkin-Lymphoma."*

S: Annotates the image with RDF annotations and displays a label for the recognized ICD-10 term.

R: *"Find similar lesions with characteristics: hyper-intense and/or coarse texture."*



Lymphknoten erkennbar. Mediastinal im vorderen sowie vor allem prätracheal im aorto-pulmonalen Fenster sowie subcarinal ausgeprägte Lymphombildung unter Einschluss des rechten Hilus. Der Ösophagus ist in seinem Verlauf infracarinal nicht mehr abgrenzbar. Der maximale Durchmesser beträgt infracarinal knapp 5,7 cm. Nebenbefundlich liegt eine Koronarsklerose und Klappenverkalkung vor. Die Herzgröße ist im Normbereich. Im Lungenfenster kein Nachweis pneumonischer Infiltrate oder Erguss. Es zeigen sich jedoch Rundherdbildungen gruppiert im Mittellappen rechtsseitig. Die übrigen Lungenlappen sind frei von Rundherdbildungen. Ein größerer (1,8 cm) RH im Mittellappen ist an der Grenze zum Oberlappen gelegen. Damit ist hier eine Infiltration auch des Oberlappenrandes nicht ausschließbar. Abdominell ist die Leber gering inhomogen in ihrem Kontrastverhalten. Eine umschriebene Läsion ist in der vorliegenden portal-venösen Phase nicht erkennbar. Eine kleine hypodense Formation am Segment 4A hin zum Ligamentum falciforme sollte einem gezielten Ultraschall zugeführt werden, da hier durch Partialvolumeneffekte im CT die Aussagekraft erfahrungsgemäß zu gering ist. Die Milz ist durchsetzt von hypodensen Läsionen in ihrer Konfiguration dadurch im cranialen Anteil kubisch. Freie Flüssigkeit ist nicht erkennbar. Die Nebenniere rechtsseitig stellt sich regelrecht dar. Auf der linken Seite ist sie verplumpt und umgeben von pathologischen Lymphknoten. Die Nieren sind regelrecht kontrastiert. Entlang der großen Gefäße und ihren Abgängen zahlreiche, zum Teil konfluierende Lymphome bis hin z.B. zum Milzhilus aber auch an den Nierenstielen. Kleinere Lymphknoten auch mesenterial. Nebenbefundlich Arteriosklerose in der Bauchaorta, hier besonders infrarenal und in der Beckenachse aber auch in den großen Abgängen mesenterial und Truncus coeliacus. In der Beckenachse keine pathologisch vergrößerten Lymphknoten. Demgegenüber jedoch inguinal bds. zahlreiche kleinere Lymphknoten.

S: MEDICO displays the search results in the record table (also see first screenshot) ranked by the similarity and match of the medical terms that constrain the semantic search (left) and opens the first hit, Peter Maier (arrow), the record, and his images that correspond to the search.

The system rearranges the SIEs for the two patients for a comparison.

R: *"Get the findings of this patient."*

S: Opens the findings (text) and highlights the medical terms in different groups.

Figure 30: MEDICO Dialogue Sequence

8.5 Findings from MEDICO Usability Studies

The user studies we conducted evaluated the design of the dialogue system and its potential to speed up the patient finding process while delivering semantic annotations that can be directly used for image retrieval.

In intensive discussions with clinicians we analyzed how the use of semantic technologies can support the clinician's daily work tasks, apart from the fact that in daily hospital work, clinicians can only *manually* search for *similar* images. After this initial period, we implemented our proposed solution, the semantic dialogue shell for radiologists. This pre-study involved three radiologists and eight medical experts who were responsible for the ontology models and automatic image annotations provided as input. The study reveals that all medical experts consider the image region annotation step for refined anatomy (FMA) and the disease (RadLex) as the real major knowledge acquisition bottleneck in this domain.

Accordingly, we tried to factor in the benefits of the speech-based annotation step in the contemporary clinical workflow. In the experimental setting, the prototype was used to refine the anatomy and disease annotations of 10 image series of different patients (approx. 100 annotations). This annotation step can be compared to a desktop-based semantic annotation tool where the user is presented a top-town menu to select the ontology-based annotations. The speech-based annotation system worked with the disease-relevant RadLex terminology (~6000 medical terms) and the dialogue competence as illustrated in Figure 30. The speech recognition accuracy is about 95% when using a commercial ASR component. (Please note that the speech grammar not only includes the medical terms, but also the complex expressions for patient retrieval and comparison.) The dialogue-based annotation can be done at a rate of about 6 annotations per minute (including the visual feedback phase) whereas the desktop-based annotation can be done at a rate of roughly 3 annotations per minute. In addition, the desktop-based tool cannot be used to retrieve and compare complete patient records. For this purpose, we designed the bigger touchscreen installation. Most importantly, the prototype dialogue system delivers semantic annotations which are unavailable in the current clinical finding process at the partner hospitals.

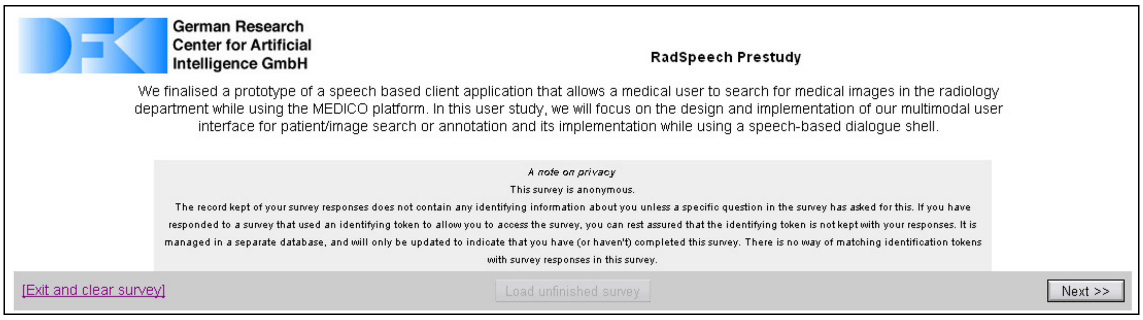
In further studies, we will try to assess the benefit of semantic annotations in general in terms of annotation accuracy/speed when compared to the process when the text-based form, which is currently used, has to be manually transferred into a machine-readable report.

8.6 MEDICO Usability Questionnaires

After the informal user study of the first prototype, we developed a second Web-based user study. For this purpose, we used the state-of-the-art user survey tool LimeSurvey, which is freely available (http://www.limesurvey.org/).

LimeSurvey is one of the leading open source tools for online surveys and it contains many survey features allowing you to do nearly every type of survey. For example, there are 20 different question types such as Arrays (5 Point choice), Multiple Numerical Input, Yes/No, Multiple Options with Comments, or Short Free Text. In addition, the Browser-based administration tool allows a very intuitive grouping of multiple questions and deployment possibilities.

We used LimeSurvey to create a userstudy “RadSpeech Prestudy” in order to guide our implementation towards the RadSpeech application. Figure 31 shows the starting screen; Figure 32 one of the Likert Scale usability questionnaires (following http://oldwww.acm.org/perlman/question.cgi?form=CSUQ, the IBM template from J. R. Lewis (1995)).


German Research Center for Artificial Intelligence GmbH

RadSpeech Prestudy

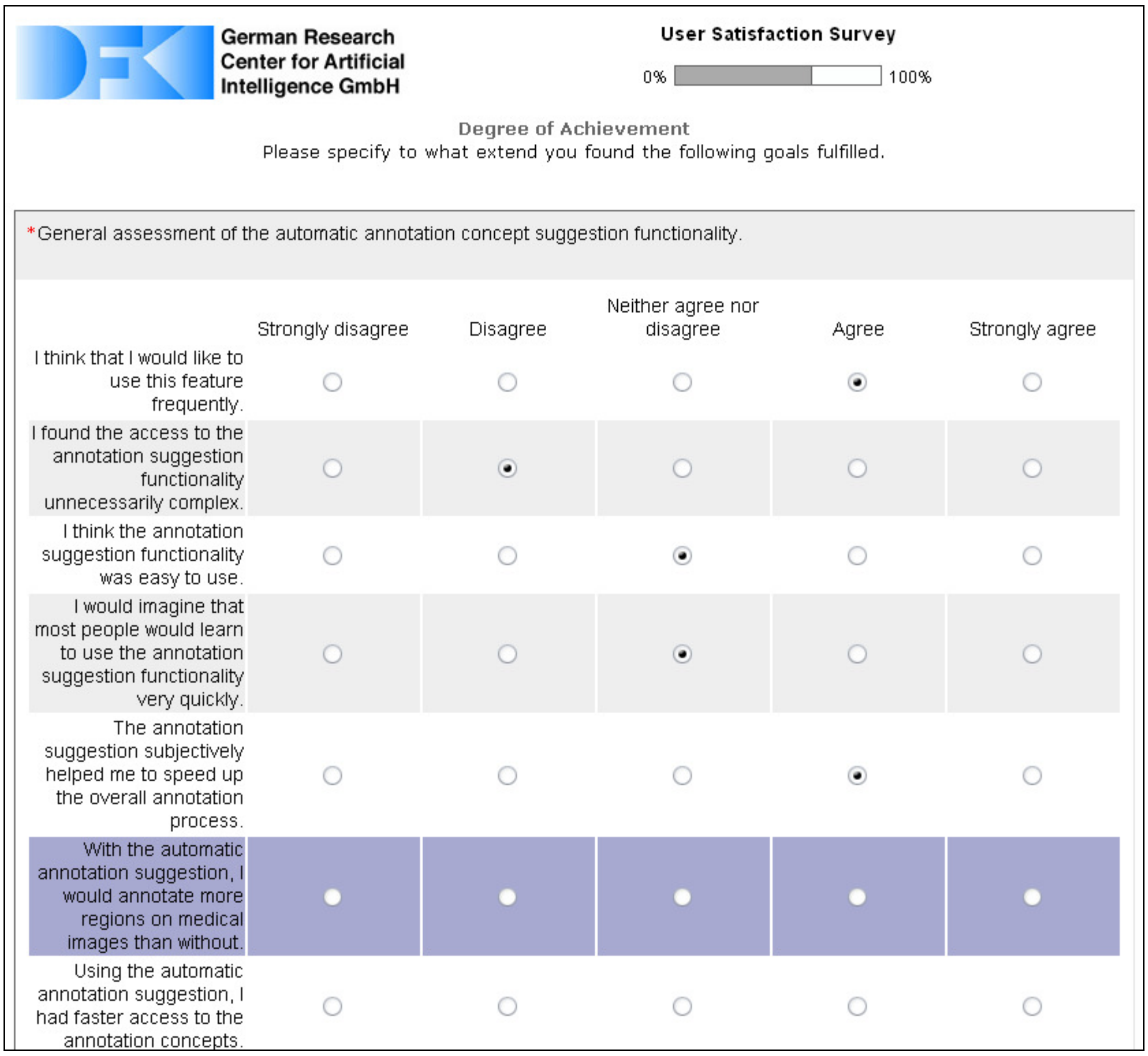
We finalised a prototype of a speech based client application that allows a medical user to search for medical images in the radiology department while using the MEDICO platform. In this user study, we will focus on the design and implementation of our multimodal user interface for patient/image search or annotation and its implementation while using a speech-based dialogue shell.

A note on privacy

This survey is anonymous.

The record kept of your survey responses does not contain any identifying information about you unless a specific question in the survey has asked for this. If you have responded to a survey that used an identifying token to allow you to access the survey, you can rest assured that the identifying token is not kept with your responses. It is managed in a separate database, and will only be updated to indicate that you have (or haven't) completed this survey. There is no way of matching identification tokens with survey responses in this survey.

Figure 31: Starting Screen


German Research Center for Artificial Intelligence GmbH

User Satisfaction Survey

Degree of Achievement

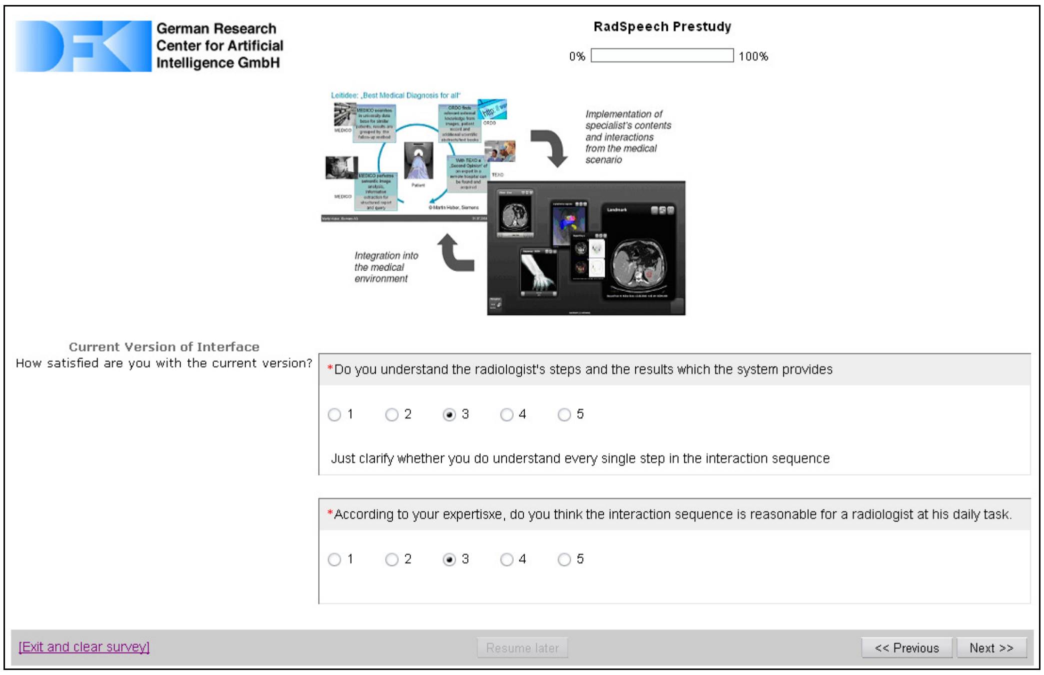
Please specify to what extend you found the following goals fulfilled.

*General assessment of the automatic annotation concept suggestion functionality.

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
I think that I would like to use this feature frequently.	○	○	○	●	○
I found the access to the annotation suggestion functionality unnecessarily complex.	○	●	○	○	○
I think the annotation suggestion functionality was easy to use.	○	○	●	○	○
I would imagine that most people would learn to use the annotation suggestion functionality very quickly.	○	○	●	○	○
The annotation suggestion subjectively helped me to speed up the overall annotation process.	○	○	○	●	○
With the automatic annotation suggestion, I would annotate more regions on medical images than without.	○	○	○	○	○
Using the automatic annotation suggestion, I had faster access to the annotation concepts.	○	○	○	○	○

Figure 32: Five Point Likert Scale Evaluation

LimeSurvey allows us to import images and screenshots from the MEDICO prototype’s user interface (Figure 33). In addition, we can provide complete action sequences and ask for improvements (Figure 34). The resulting surveys are used in the development process of the radiology prototype. An example can be downloaded from http://manuelm.org/survey/index.php?sid=63872&lang=en.


Leitidee: „Best Medical Diagnosis for all"
Implementation of specialist's contents and interactions from the medical scenario
Integration into the medical environment

German Research Center for Artificial Intelligence GmbH

RadSpeech Prestudy

0% 100%

Current Version of Interface
How satisfied are you with the current version?

*Do you understand the radiologist's steps and the results which the system provides

1 2 [✓ 3] 4 5

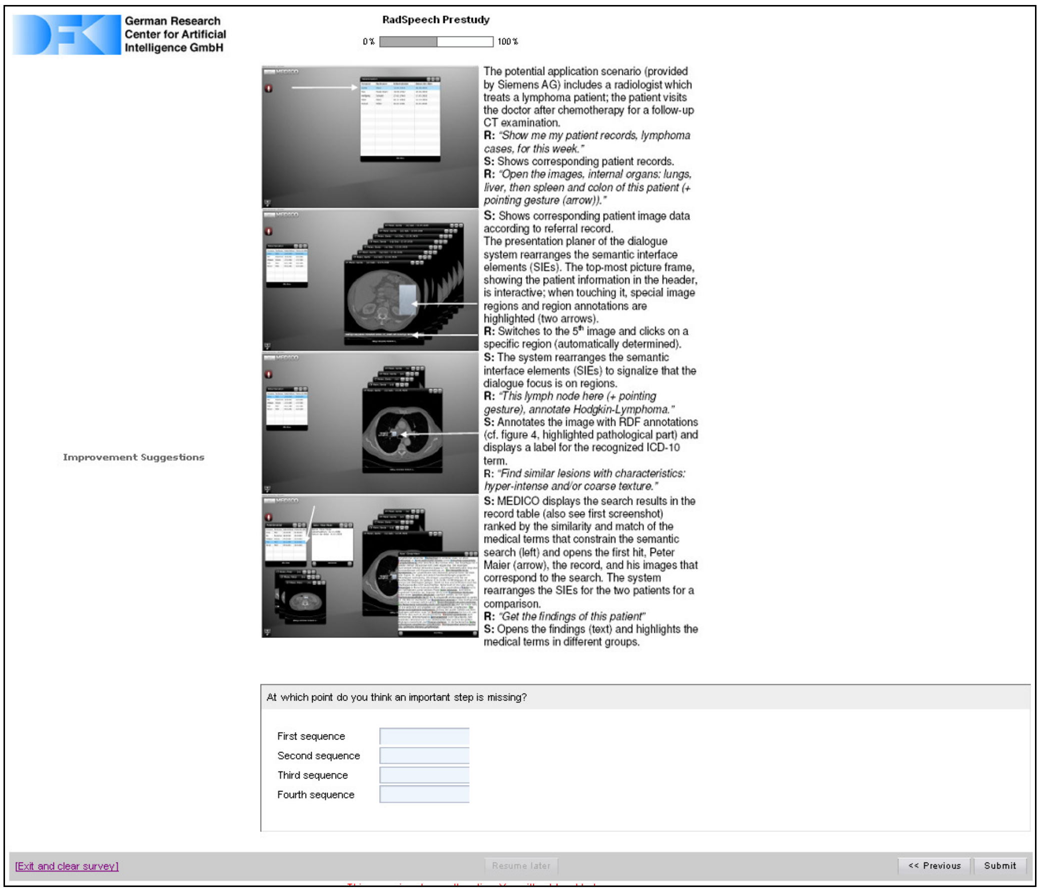
Just clarify whether you do understand every single step in the interaction sequence

*According to your expertisxe, do you think the interaction sequence is reasonable for a radiologist at his daily task.

1 2 [✓ 3] 4 5

[Exit and clear survey] Resume later << Previous Next >>

Figure 33: RadSpeech Evaluation Questionnaire



German Research Center for Artificial Intelligence GmbH

RadSpeech Prestudy

0% 100%

The potential application scenario (provided by Siemens AG) includes a radiologist which treats a lymphoma patient; the patient visits the doctor after chemotherapy for a follow-up CT examination.
R: *"Show me my patient records, lymphoma cases, for this week."*
S: Shows corresponding patient records.
R: *"Open the images, internal organs: lungs, liver, then spleen and colon of this patient (+ pointing gesture (arrow))."*
S: Shows corresponding patient image data according to referral record.
The presentation planer of the dialogue system rearranges the semantic interface elements (SIEs). The top-most picture frame, showing the patient information in the header, is interactive; when touching it, special image regions and region annotations are highlighted (two arrows).
R: Switches to the 5th image and clicks on a specific region (automatically determined).
S: The system rearranges the semantic interface elements (SIEs) to signalize that the dialogue focus is on regions.
R: *"This lymph node here (+ pointing gesture), annotate Hodgkin-Lymphoma."*
S: Annotates the image with RDF annotations (cf. figure 4, highlighted pathological part) and displays a label for the recognized ICD-10 term.
R: *"Find similar lesions with characteristics: hyper-intense and/or coarse texture."*
S: MEDICO displays the search results in the record table (also see first screenshot) ranked by the similarity and match of the medical terms that constrain the semantic search (left) and opens the first hit, Peter Maier (arrow), the record, and his images that correspond to the search. The system rearranges the SIEs for the two patients for a comparison.
R: *"Get the findings of this patient"*
S: Opens the findings (text) and highlights the medical terms in different groups.

Improvement Suggestions

At which point do you think an important step is missing?

First sequence	
Second sequence	
Third sequence	
Fourth sequence	

[Exit and clear survey] Resume later << Previous Submit

Figure 34: Display of Complete Action Sequences

9 New Research Topic: The Usability of Concrete Usability Guidelines

In the previous chapters we have studied usability guidelines and their application to the THESEUS use cases. In this final chapter, we will discuss meta guidelines for THESEUS Software and Interface Developments.

One of the most important usability engineering methods is the creation of guidelines. In this document, we will challenge the usability of traditional usability guidelines. Oftentimes, guideline descriptions and explanations are unsatisfactory, remaining vague and ambiguous in explanation.

By offering guideline descriptions that are more understandable, will hope practitioners will be able to choose relevant usability guidelines more easily and use them more carefully.

9.1 Background

Users of usability studies, such as THESEUS developers, tend to be dissatisfied with them. In this chapter, we will review existing guidelines, provide a literature overview, and inspect their internal structure. In general, guidelines with better descriptions are required.

9.1.1 Existing Guidelines

When creating software, a website, or other material, different information is communicated between numerous contributors and participants. These contributors are present at various stages of the process. Communication takes place between:

- Researchers and guideline constructors;
- Guideline constructors and designers;
- Designers and IT-systems.

The risk of communication failure or misunderstanding along the way is high. For almost all design situations the meaning of a guideline remains a matter of interpretation. Nevertheless, usability guidelines should be described in a way that they are perceived as usable and result in reducing the risk of communication failure. This can be achieved if good descriptions are provided.

Questions are "Is a guideline worth following?" and "How do I know that using a guideline will result in a more usable IT-system?" Spool (2009) notes that, in order to measure guidelines, they must first be expressed in numbers which can be compared. Spool's claim can be viewed as another motive for providing usable descriptions. Van Welie (1994) discusses guidelines and heuristics as a means to improve usability while designing. Furthermore, he points out that in practice the available checklists, tests, guidelines, etc., differ in terms of structure, content, and terminology. It is clear that guidelines need to have better descriptions.

Many guidelines or criteria for designing IT-systems presently exist (e.g., Nielsen, 1994; Häkkilä & Mäntyjärvi, 2006; Kärkkäinen, L. & Laarni, 2002; Muller, et al., 1998; Shneiderman, 1998). Guidelines are based on theories, empirical data, and good practical experiences. A guideline is defined as "information intended to advise people on how something should be done or what something should be" (Cambridge Dictionaries Online, 2009). Are the proposed guidelines being used and if so, why? Are popular usability guidelines even usable?

In this section, a set of principles for formulating guidelines will be proposed. These results are aimed at constructors of usability guidelines. We propose a set of meta guidelines (selected from Cronholm, 2009) to improve the usability of usability guidelines. These should give advice for the formulation of application specific usability guidelines. Good usability of human-computer interaction is the main goal of interface design (especially on the skeleton and surface plane).

At the top level (meta-meta-design level) a constructor is using theory, existing guidelines, and empirical experiences in order to propose principles for how to construct or design usable guidelines. At the next level (meta-design level), a constructor uses theories, principles, and empirical experiences in order to actually construct guidelines. At the design level, a designer applies usability guidelines in designing an IT-system. Finally, a user interacts with an IT-system at the so-called business level. The ultimate aim of the principles is to support the usability of IT-systems. We define usability as "a set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users" (AS/NZS_42169). Cronholm's suggestion

9.1.2 Literature on Guidelines

Literature offers several explanations of why usability guidelines are not used.

- Vredenburg et al. (2002) point out the major discrepancy between the commonly cited guidelines and the actually applied ones.
- The work by Tao (2008) has identified a significant gap between the knowledge of design and the application of web design guidelines.
- According to Chevalier & Ivory (2003), there is little support for the complexities involved in the design activity.
- Gould & Lewis (1985) claim that design guidelines are limited since their descriptions are not detailed enough.
- According to Burmester and Machate (2003), the designer has to understand the rationale behind the guidelines before applying them.

All of this criticism suggests that something vital is missing in current guideline descriptions and motivates a review of how to formulate guidelines. The part which is often missing is most often the significant part. A description must be meaningful but still concise. A reasonable condition for adopting commonly cited guidelines is that they are expressed in an understandable way. To determine how helpful a guideline is, ask some of the following questions in reference to it:

- What is this guideline about?
- Who is this guideline aimed at helping?

- What will I achieve by using this guideline?
- How should I use this guideline?
- Are there any examples provided?
- Does this guideline answer the 5 Ws (who, what, where, when, why)?

Granted, this brief list of questions is still vague, but it nevertheless gives us a starting point when it comes to determining whether a set of guidelines is useful or not.

9.1.3 Current Guideline Structures

Usability guidelines are commonly presented as flat lists (Nielsen, 1994; Häkkilä & Mäntyjärvi, 2006; Kärkkäinen, L. & Laarni, 2002; Muller, et al., 1998; Shneiderman, 1998).

A flat list is a list that is not categorized. An example is "Visibility of system status" (Nielsen, 1994) and "Match between the system and real world" (Nielsen, 1994). According to Cronholm & Bruno (2008, 2009), guidelines can reside on different abstraction levels. Clearly, the latter guideline resides on a higher abstraction level. By contrast, concrete guidelines give specific recommendations for, e.g., the organization of graphical interface elements. Usability guidelines are better understood if they are categorized into different abstraction levels. The advantage and role of a multilevel abstraction hierarchy is discussed in Rasmussen et al. (1994). Rasmussen et al. compare a multilevel abstraction hierarchy with a means-end hierarchy and claim that the former is often used in practical problem solving processes.

Gerlach & Kuo (1991) work towards enabling design practice to become more systematic and less intuitive than it is today. We, as well, recommend formulating guidelines in a more categorized and systematic way.

We have also been inspired by question batteries proposed by Strauss & Corbin (1998). Examples of questions borrowed from these batteries are:

- What is this sentence about?
- Why is this formulation hard to comprehend?
- What audiences am I writing for?
- Why and how is this document used?
- What is the meaning of this concept as part of the document?
- Why is this information represented in this document? For what purpose is this information used?
- What do you need in order to produce this information, and to whom do you communicate this information?

In THESEUS, we formulated some principles of design and their application in connection with the five usability planes illustrated in these *Guidelines* (see Figure

13 and Figure 35). In the implementation stage, the functional components were implemented while taking the design principles into account, in accordance with the prototype development stage.

1. The *broad accessibility* principle applies to the scope plane where the functional specifications and interface content requirements are met. Broad accessibility means that research and demonstrator systems should be accessible and usable by people of diverse abilities and backgrounds.

2. The *alignment* principle deals with the relative placement of visual affordances on a graphical screen; they should be aligned with related elements to create a sense of unity and cohesion. This principle often applies to the structure plane.

3. The *80/20 rule* suggests that 80% of a demonstrator's usage involves only 20% of the provided (dialogical) interaction competences or interface features. On the skeleton plane this principle essentially provides a recommendation for the implementation of the multimodal input possibilities.

4. *Aesthetic effect* describes the phenomenon that aesthetic designs are (subconsciously) more easily perceived and more effective at fostering a positive attitude towards the demonstrator system. This principle applies to the surface plane and the visual design of the graphical interface.

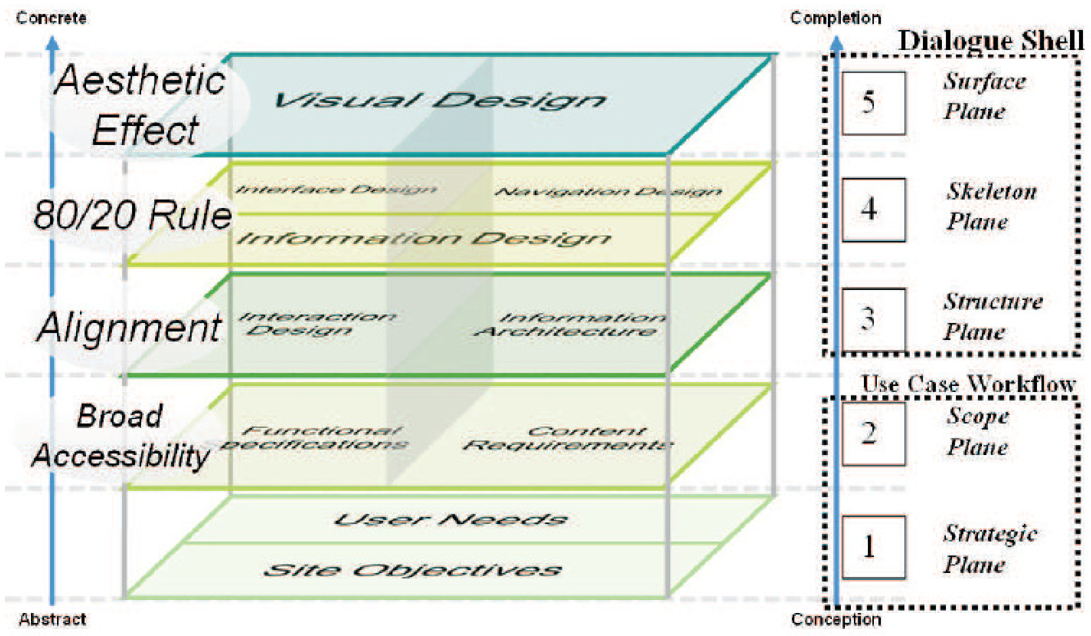


Figure 35: Usability Planes and Selected Design Principles

9.2 Meta Guidelines

In the following, we present four meta guidelines.

9.2.1 Meta Guideline 1 "Use paraphrases for rule descriptions."

According to Wittgenstein, one should, among other things, be careful about the use of nouns. Wittgenstein claims that many concepts are given the form of a noun instead of the original adjective or verb form. We normally name this *noun disease*. Non-reflected transferring of adjectives or verbs to nouns can lead us to wrongly think that there is an object or thing behind the noun when the noun in fact reflects an attribute of an object or a verb. This is relevant because it leads to imprecise guidelines. Also, the heterogeneity of different guidelines might lead to confusion. We believe a recommendation should contain both a verb and a noun, i.e., a recommendation to a designer to do something (verb/action) about something (noun/object). Using verbs/actions and nouns/objects together can be viewed as more active way of using the language. An example of a formulation of a guideline description following this form is presented by Shneiderman (1998): "Strive for consistency."

9.2.2 Meta Guideline 2: "Be suspicious of prominent rules."

The following two tables of guidelines by Nielsen (2009) and Shneiderman (1998) list numerous instances where recommendations are too vague to be particularly useful. In all of these examples, it quickly becomes clear just how much is left open to interpretation. For example, the description to Nielsen's "visibility of system status" mentions the need for "appropriate feedback within reasonable time." While this sounds correct, what exactly denotes "appropriate feedback" and how much time is "reasonable"? (We described response times and timing constraints in Chapter 2) Although very useful as an introduction, the guidelines (Tables 3 and 4) leave out concrete information which is vital for them to be useful in a specific THESEUS use case situation.

Visibility of system status	The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
Match between system and the real world	The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
User control and freedom	Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
Consistency and standards	Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
Error prevention	Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
Recognition rather than recall	Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

Flexibility and efficiency of use	Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
Aesthetic and minimalist design	Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
Help users recognize, diagnose, and recover from errors	Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
Help and documentation	Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Table 3: Guidelines as Established by Nielsen 2009

Strive for consistency	Consistent sequences of actions should be required in similar situations; identical terminology should be used in prompts, menus, and help screens; and consistent commands should be employed throughout.
Enable frequent users to use shortcuts	As the frequency of use increases, so do the user's desires to reduce the number of interactions and to increase the pace of interaction. Abbreviations, function keys, hidden commands, and macro facilities are very helpful to an expert user.
Offer informative feedback	For every operator action, there should be some system feedback. For frequent and minor actions, the response can be modest, while for infrequent and major actions, the response should be more substantial.
Design dialogue to yield closure	Sequences of actions should be organized into groups with a beginning, middle, and end. The informative feedback at the completion of a group of actions gives the operators the satisfaction of accomplishment, a sense of relief, the signal to drop contingency plans and options from their minds, and an indication that the way is clear to prepare for the next group of actions.
Offer simple error handling	As much as possible, design the system so the user cannot make a serious error. If an error is made, the system should be able to detect the error and offer simple, comprehensible mechanisms for handling the error.
Permit easy reversal of actions	This feature relieves anxiety, since the user knows that errors can be undone; it thus encourages exploration of unfamiliar options. The units of reversibility may be a single action, a data entry, or a complete group of actions.
Support internal locus of control	Experienced operators strongly desire the sense that they are in charge of the system and that the system responds to their actions. Design the system to make users the initiators of actions rather than the responders.
Reduce short-term memory load	The limitation of human information processing in short-term memory requires that displays be kept simple, multiple page displays be consolidated, window-motion frequency be reduced, and sufficient training time be allotted for codes, mnemonics, and sequences of actions.

Table 4: Guidelines as Established by Shneiderman 1998

9.2.3 Meta Guideline 3 "Provide examples for what to do and how."

Provide concrete examples of *what-to-do* and *how-to-do* since many usability experts can only implicitly use guidelines and prominent rules. The guidelines are often formulated as "Support undo and redo" and "As much as possible, design the system so the user cannot make a serious error". These descriptions do not inform the designer about what and how to do something on a more concrete level, though. Answering the following questions might help to define the "what" and "how" on a more concrete level:

- What are explicit guidelines for usage?
- I have used some of the general usability guidelines as inspiration, but how can this inspiration be formalized?

Additionally, there is often confusion between frequent users and expert users. A frequent user does not have to be an expert user (expert users often prefer accelerators.)

9.2.4 Meta Guideline 4 "Use text form principles."

Usability guideline descriptions can be improved. Based on this analysis, a set of text form **principles** have been generated (Cronholm, 2009). The generated principles are: ***relevance***, ***precision***, ***homogenous structure form***, ***language form***, and ***abstraction and granularity***.

The principle of ***relevance*** is a recommendation for why a specific guideline is important to consider while designing or evaluating an IT-system. The formulation of this guideline should also inform the designer/evaluator for whom this guideline is relevant, when it is relevant, and for which specific situation it is applicable. An IT-system supports a specific business and its specific business goal. It should be clear to a designer whether or not a guideline is applicable to the specific situation at hand.

The principle of ***precision*** A recommendation should be carefully chosen and precise enough to avoid confusion. All recommendations used, either in the heading or in the description, should be explained. By omitting explanations, there is a risk for confusion that may ultimately lead to the guidelines being abandoned.

Homogeneous structure form All guideline descriptions should, if possible, consistently be constructed according to the same underlying model. A good idea is to start with a description of why this guideline is important, explain the concepts used, and end with short and concise imperative (cf. language form) together with an illustrative example.

The sub-principle of ***language form*** refers to how the description of the guideline is formulated. It has been touched upon in meta guideline 1 already. A guideline is per se a recommendation. The description should therefore be expressed in an active form instead of a more passive, descriptive form. In an active form, the formulation is an imperative to a designer/evaluator. The imperative consists of a verb/action together with a noun/object. An imperative informs the designer/evaluator clearly about *what-to-do*. There should be no hesitation of what the designer/evaluator is to achieve. The language used should therefore be clear and simple. In order to achieve the recommendation of *what-to-do*, a guideline description should also include of a proposal of *how-to-do*. Some basic examples include:

- Maintain consistency by using only one font color, unless you are highlighting, categorizing, or differentiating between content.
- Increase the user's feeling of control by providing clearly marked exits, such as cancel or undo buttons.
- Include good error messages so that users can fix any problems themselves. These messages should be clear, precise, helpful, and polite.

The principle of ***abstraction and granularity***: Guideline descriptions are encouraged that consist of different abstraction levels since this supports comprehensibility. For example, instead of presenting them as a flat, unordered list, related guidelines should be categorized. Categorizing guidelines means that different levels of guidelines are identified. Categories can be divided into sub-categories and they can be abstracted to higher levels. According to Rasmussen et al., categorizing guidelines supports practical problem solving processes (Rasmussen et al., 1994). For example, a concrete recommendation with a very low level of abstraction could be, "To create this website, make the background white and the font black. Use the font size 32 for headings and the font size 18 for paragraphs. " By contrast, the following recommendation is much more abstract: "Make sure text is easy to read when creating this website."

9.3 Future Research on Meta Guidelines and THESEUS Use Cases

A proposal for future research and the integration of the meta guidelines and their application into THESEUS prototypes is 1) to describe guidelines according to the principles described here and 2) to test the guidelines in real development / evaluation projects.

This analysis has revealed some major deficiencies in how usability guidelines are described and explained. This work should not be perceived as finished; rather it is a start for further elaboration on recommendations for describing meta guidelines. These recommendations should have a direct impact on how specific usability guidelines for use case scenarios should be formulated according to the general guidelines contained in this document (we emphasized the importance of usability testing, particularly during the development of a given THESEUS prototype). We discussed the many advantages of testing prototypes and products in terms of costs, product quality, and customer satisfaction and concluded with a discussion of meta guidelines in order to give more concrete advice on usability guidelines and usability issues in general.

10 Further Readings

In this section we make useful recommendations for finding more detailed information about some of the relevant usability topics.

10.1 ISO Standards and Other Guidelines for Interface Usability

These are relevant ISO standards and guidelines concerning interface usability. The section contains international standards and special guidelines for major operation systems.

ISO 9126

Software engineering – product quality part 1: Quality model
ISO 9126 is an international standard for the evaluation of software quality. It is being superseded by the project SQuaRE (Software product Quality Requirements and Evaluation).
The ISO 9126-1 classifies software quality in a structured set of characteristics and sub-characteristics: functionality, reliability, usability, efficiency, maintainability, and portability.

ISO 25062

Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Common Industry Format (CIF) for usability test reports

ISO 9241

Ergonomic requirements for office work with visual display terminals
Part 11: Guidance on usability
Part 12: Presentation of information; Ergonomics of human-system interaction
Part 110: dialogue principals

ISO 13407

Human centered design process for interactive systems

ISO 16071

Ergonomics of human-system interaction - Guidance on accessibility for human-computer interfaces

ISO 16982

Ergonomics of human-system interaction – Usability methods supporting human-centered design

BSI 7179 (British Standards Institution)

Specifications for VDT Workstations: 5

ITU T P.851 (International Telecommunication Union)

Subjective quality evaluation of telephone services based on spoken dialogue systems

Information for developers using Microsoft® tools, products, technologies and services
http://msdn.microsoft.com/en-us/library/default.aspx

Open Source KDE Usability Guidelines Wiki
http://techbase.kde.org/Projects/Usability#

Usability.gov
http://www.usability.gov

10.2 Publications Concerning Usability

10.2.1Books

The Elements of User Experience (Jesse James Garret)
Jesse James Garret describes the different planes of user experience.

Don't Make Me Think! (Steve Krug)
Steve Krug gives a short introduction to usability testing.

The Design of Everyday Things (Donald A. Norman)
Donald A. Norman provides some thoughts on things we are annoyed about every day.

GUI Bloopers (Jeff Johnson)
Jeff Johnson writes about the dos and don'ts in interface design on the web.

The Human-Computer Interaction Handbook (Sears, Jacko)
This is a collection of interesting papers in the field of HCI.

The Adaptive Web (Brusilovski et al.)
This book deals with the topic of personalization and user adaptation.

Usability Engineering (Jakob Nielsen)
Jakob Nielsen's classic on this matter describes the basics of usability engineering.

Voice User Interface Design (Cohen et al.)
This book describes the basics of voice user interface design.

Web Usability (Nielson, Loranger)
The book presents results from several studies in the field of web usability.

Designing the User Interface: Strategies for Effective Human-Computer-Interaction (Shneiderman)
This book provides an introduction to user-interface design.

Creative Design of Interactive Products and Use of Usability Guidelines (Machate, Burmester)
This book proposes the strategic use and definition of guidelines associated with a user centred design process.

10.2.2Papers

SmartWeb Handheld Interaction: General Interactions and Result Display for User-System Multimodal Dialogue
http://www.dfki.de/web/forschung/publikationen?pubid=2919

Analysis and Simulation of User Interfaces
http://www.uclic.ucl.ac.uk/harold/srf/hci2000.pdf

Enabling Context-Sensitive Information Seeking
http://portal.acm.org/ft_gateway.cfm?id=1111479&type=pdf&coll=GUIDE&dl=GUIDE&CFID=47054797&CFTOKEN=81157679

Designing for Usability: Key Principles and What Designers Think
http://portal.acm.org/citation.cfm?id=3170

10.2.3Websites

Fraunhofer FIT
This is the homepage of the Fraunhofer Institute for Applied Information Technology. They provide an overview of usability methods in German.
http://www.fit.fraunhofer.de/services/usability/methoden.html

Usability Net
UsabilityNet is a project funded by the European Union to promote usability and user centered design. You can find a collection of methods for different phases of the design process.
http://www.usabilitynet.org/tools/r_international.htm

Craig Marion's HCI / Experience Design Page
This is a collection of useful links concerning usability engineering. Furthermore, you can find links to related topics.
http://mysite.verizon.net/resnx4g7/Academic/UseEng.html

Human Factors International
Human Factors International is company focused on software usability. They provide an overview of tools and standards.
http://www.humanfactors.com

Martijn van Welie's Homepage
On this site you can find a collection of practices in interaction design. You can find solutions for special problems. It is a library where a lot of patterns are listed and grouped into categories. Furthermore, you can find some tips and tricks for Visio.
http://www.welie.com/index.php

Don Norman's Homepage
Don Norman's website provides articles about design principles as well as links for further reading.
http://www.jnd.org

Bruce Tognazzini's Homepage
Bruce Tognazzini is an interaction design consultant who worked for companies like Apple and Microsoft. In the interaction design section you will find the basic principles and recommended readings.
http://www.asktog.com/index.html

Ben Shneiderman's Homepage
On Ben Shneiderman's homepage you find his papers on human-computer-interaction.
http://www.cs.umd.edu/~ben/

Bill Buxton's Homepage
Bill Buxton is a principal researcher at Microsoft Research. He is concerned with user experience.
http://www.billbuxton.com/

Jakob Nielsen's Homepage
Jakob Nielsen presents examples of bad design and current topics in usable information technology.
http://www.useit.com/

Jeff Johnson's Homepage
This website has an archive of web bloopers.
http://www.uiwizards.com/

STC Usability Website
This website is a forum for sharing information and experiences on issues related to usability and user-centered design. It is the home of the Usability and User Experience Community of the Society for Technical Communication.
http://www.stcsig.org/usability/

A Framework for Organizing Web Usability Guidelines
This website provides a framework for organizing web usability guidelines.
http://www.isys.ucl.ac.be/bchi/publications/2000/Scapin-HFWeb2000.htm

Perlman Software: Web-Based User Interface Evaluation with Questionnaires
http://hcibib.org/perlman/question.html

Questionnaires in Usability Engineering
http://www.ucc.ie/hfrg/resources/qfaq1.html

10.3 Tools for Conducting Tests or Simulations

10.3.1Software for Data Logging

Logging software is installed on the test computer and logs mouse movements, keyboard use, and system output. In some cases a second program is installed on the experimenter's computer for annotating users' actions.

Logging software may be used for all desktop scenarios without any problems. At this time, there is no logging software publicly available for mobile devices. In the context of CTC-WP4 dialogue shells, we implement additional logging facilities together with the mobile device application.

In this section you will find links to data logging software companies. Some companies provide commercial software and freeware tools as well.

Freeware

The following software is available at no cost.

uLog Lite
www.noldus.com/ulog

Ovo Logger
http://www.ovostudios.com/ovologger.asp

Commercial Software

Software listed in this section is commercial.

Techsmith Morae
http://www.techsmith.com/morae.asp

Noldus Observer
http://www.noldus.com/site/doc200401012

Interact
http://www.mangold-international.com/?id=11&L=1

Usabilityware
http://www.usabilitysystems.com/prod_usability_software.html

Biobserve Spectator2
http://www.usability.biobserve.com/index.html

Ovo Logger
http://www.ovostudios.com/ovologger.asp

uLog Pro
www.noldus.com/ulog

10.3.2 Software for Simulation

This software can be used to build prototypes and simulate user behavior:

CogTool

CogTool leverages the concept of a design storyboard to to create accurate models of skilled performance behavior.

http://www.cs.cmu.edu/~bej/cogtool/

10.4 Questionnaires

10.4.1 Online Questionnaires Toolkit

2ask

2ask is a website where you can build your own online questionnaire. 2ask will administer your questionnaire and analyze the results. This is a commercial service http://www.2ask.de/

LimeSurvey

LimeSurvey is one of the leading open source tool for online surveys and it contains many survey features you need for doing nearly every survey type.

http://www.limesurvey.org/

10.4.2 Evaluation Questionnaires

The following links present software for basic usability questionnaires. Most are not free of charge.

SUMI

The Software Usability Measurement Inventory is a rigorously tested and proven method of measuring software quality from the end user's point of view. SUMI is a consistent method for assessing the quality of use of a software product or prototype, and can assist with the detection of usability flaws before a product is shipped. It is backed by an extensive reference database embedded in an effective analysis and report generation tool.
http://sumi.ucc.ie/index.html

CSUQ

The Computer System Usability Questionnaire was developed by Lewis for IBM in 1995. It contains 19 questions.
http://oldwww.acm.org/perlman/question.cgi?form=CSUQ

SUS

John Brooke published the concept of a "System Usability Scale," a reliable, low-cost usability scale that can be used for global assessments of systems' usability. SUS is based on a Likert scale questionnaire with standardized content that results in an overall usability and user satisfaction index (ranging from 0 to 100).

http://meiert.com/en/blog/20070423/revitalizing-sus-the-system-usability-scale/

QUIS

The Questionnaire for User Interaction Satisfaction (QUIS) is a tool developed by a multi-disciplinary team of researchers in the Human-Computer Interaction Lab (HCIL) at the University of Maryland at College Park. The QUIS was designed to assess users' subjective satisfaction with specific aspects of the human-computer interface. The QUIS team successfully addressed the reliability and validity problems found in other satisfaction measures, creating a measure that is highly reliable across many types of interfaces.
http://www.lap.umd.edu/QUIS/index.html

Adaptation of Microsoft Product Reaction Card

This document contains all of the words used on the product reaction cards, Users can mark the words on this card according to their attitude towards the tested software. For example, a program can be described as clear, fast, and sophisticated. Another program can be associated with words like difficult, old, and slow. This is a method used to get a feeling of what users think about the software.
http://www.microsoft.com/usability/UEPostings/ProductReactionCards.doc

11 Bibliography

A. Chevalier and M.Y. Ivory (2003). Web site designs: Influences of designer's expertise and design constraints. International Journal of Human-Computer Studies, 58:1, p. 57-87.

S. Cronholm (2009). The Usability of Usability Guidelines – a Proposal for Meta-Guidelines. Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group: Design November 23-27, 2009, Melbourne, Australia.

S. Cronholm and V. Bruno (2009). Usability of IT-Systems is More than Interaction Quality – The Need for Communication and Business Process Criteria. In Proceedings of the 17th European Conference on Information Systems (ECIS). June 8-10, Verona, Italy.

S. Cronholm and V. Bruno (2008). Do you Need General Principles or Concrete Heuristics? - a Model for Categorizing Usability Criteria. Proceedings of the Australasian Computer-Human Interaction Conference (OZCHI). Cairns.

J. J. Garrett (2002). The Elements of User Experience. New York, New York, USA: American Institute of Graphic Arts.

J.H. Gerlach and F.Y. Kuo (1991). Understanding Human- Computer Interaction for Information Systems Design. MIS Quarterly, 15,4 , p. 527-549.

J.D. Gould and C. Lewis (1985). Designing for usability: key principles and what designers think. Commun. ACM, 28,3. p. 300-311.

J. Häkkilä and J. Mäntyjärvi (2006). Developing design guidelines for context-aware mobile applications, ACM International Conference Proceeding Series; Proceedings of the 3rd international conference on Mobile technology, applications & systems. Bangkok, Thailand, ACM.

L. Kärkkäinen and J. Laarni (2002). Designing for small display screens, ACM International Conference Proceeding Series; Proceedings of the second Nordic conference on Human-computer interaction. Aarhus, Denmark, ACM. p. 227-230.

Lewis, J. R. (1995). IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use. International Journal of Human-Computer Interaction, 7:1, 57-78.

D. J. Mayhew (1999). The Usability Engineering Lifecycle. San Francisco, California, USA: Morgan Kaufman.

M.J. Muller, et al. (1998). Methods & Tools: participatory heuristic evaluation, in interactions. Volume 5:5, p. 13-18.

J. Nielsen (1993). Usability Engineering. Mountain View, California, USA: Morgan Kaufman.

J. Nielsen. Heuristic Evaluation (1994). In: J. Nielsen and R.L. Mack (eds), Usability Inspection Methods, John Wiley & Sons, New York, 1994.

J. Nielsen. Paper Prototyping: Get User Data *Before* Coding (2003). http://www.useit.com/alertbox/20030414.html. Accessed April 02, 2010.

Norbert Pfleger and Jan Schehl (2006). Development of Advanced Dialog Systems with PATE. Processing of the International Conference on Spoken Language Processing (Interspeech 2006 - ICSLP), Pittsburgh, PA.

J. Rasmussen, A.M. Pejtersen, and L.P. Goodstein (1994). Cognitive Systems Engineering, Wiley & Sons Inc.

Norbert Reithinger and Gerd Herzog (2006). An Exemplary Interaction with SmartKom. In: Wolfgang Wahlster (ed.): SmartKom - Foundations of Multimodal Dialogue Systems, Springer, 2006.

B. Shneiderman (1998). Designing the user interface: Strategies for effective human-computer-interaction. 3rd ed, Addison Wesley Longman, Reading, MA.

Daniel Sonntag (2007). Interaction Design and Implementation for Multimodal Mobile Semantic Web Interface. Proceedings of HCI International 2007, HCI (9), Volume 4558, Lecture Notes in Computer Science, Springer.

Daniel Sonntag, Matthieu Deru and Simon Bergweiler (2009). Design and Implementation of Combined Mobile and Touchscreen-Based Multimodal Web 3.0 Interfaces . Proceedings of the 2009 International Conference on Artificial Intelligence (ICAI).

Daniel Sonntag (2010). Ontologies and Adaptivity in Dialogue for Question Answering. Monograph in Studies on the Semantic Web, ISBN: 978-3-89838-623-4 (AKA), 978-1-60750-054-4 (IOS Press), Hard Cover.

Spool, J.M. (2002)'Evolution trumps usability guidelines', http://www.uie.com/articles/evolution_trumps_usability/. Accessed April 12, 2010.

A. Strauss and J. Corbin (1998). Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. Sage Publications, Thousands Oaks, CA.

Y.-H. Tao (2008). Information system professionals' knowledge and application gaps toward Web design guidelines. Computers in Human Behavior, 24:3, p. 956-968.

K. Vredenburg, J.Y. Mao, P.W. Smith and T. Carey (2002). A survey of user-centered design practice. Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves. Minneapolis, USA: ACM Press.

Wolfgang Wahlster (2003). Towards Symmetric Multimodality: Fusion and Fission of Speech, Gesture, and Facial Expression. KI 2003: Advances in Artificial Intelligence, Lecture Notes in Computer Science, Volume 2821, Springer, p. 1-18.