A Teleradiology System Design Case

Erik Borälv
Uppsala University
Center for Human-Computer Studies
Lägerhyddvägen 18
S-752 37 Uppsala, Sweden
Erik.Boralv@it.uu.se

Bengt Göransson
Uppsala University
Center for Human-Computer Studies
Lägerhyddvägen 18
S-752 37 Uppsala, Sweden
Bengt.Goransson@it.uu.se

ABSTRACT
This paper describes the teleradiology application CHILI from the graphical user interface point of view. We present the most important design decisions taken during the construction of the system and discuss different methods and techniques that affected the design process.

Some non-standard design principles are presented, and the reasons behind them. Several of the basic GUI constructions used in the CHILI application are somewhat similar to those seen in Sun's HotJava Views™ [3]; the application lacks the traditional connection to the desktop metaphor and has instead a work task oriented approach.

Keywords
Design criteria, GUI, teleradiology, work task, patterns.

INTRODUCTION
The chili application is intended for teleradiology using ISDN communication. Its main characteristics is (i) the ability to transfer images between users of the chili application, (ii) the on-line viewing and processing of images, and (iii) the retrieval of images from external image capturing modalities. One of the most important purposes of the application is to (iv) enable the sharing of expensive resources, such as high quality film printers, rapid image transportation and image analysis specialists.

The two authors of the paper were given the task to design the chili application by means of constructing a working prototype and to emphasize the power usability aspects (see quote on next page). The final system would be based on new user interface design techniques to ensure a longer lifetime of the application. [1]

ORGANISATIONAL CONTEXT
The development team, including the designers, have extensive experience developing computer applications for the medical field. Application developers in the team are all medical image researchers by profession and have a thorough understanding of the medical staff's needs in terms of functionality and working conditions.

OUR FOCUS
The authors have recently experienced difficulties in the transfer of design knowledge from designer to developer.

The CHILI application in particular is the successor of an another teleradiology application where the initial design and specification phase did not guide the implementors enough, resulting in a system which was different from the expected one. [2]

The focus of this paper is to informally discuss the design process and the key design decisions made for the CHILI application.

DESIGN PROBLEMS

Image Size
It is well known in the image processing community that there is too little space vertically on the screen. This is caused by the fact that most screens have a higher resolution horizontally which usually makes the resizing of images limited by vertical space. The images in radiology are normally 512 x 512 pixels, so the optimal area for images has the shape of a square, which means that one can resize images more naturally simply by a scaling factor.

The user's general wish is to have the image area as large as possible.

Constraint
The obvious restricting factor is that we may not simply enlarge the images as much as the screen size allows, because we still need a system which is possible to use in an efficient way. E.g. we still need to be able to view patient data efficiently, search and navigate the patient database, etc.

Usability Issues
As the case is for most computer applications, the end-user community is heterogeneous. This is not only caused by different levels of computer experience, but also by the fact that there will be different categories of users targeted by the system. These categories range from general practitioners to expert radiologists to technical staff.

Constraint
The mixed user community is of more concern than might first be thought at; designers are limited in the work process by having to make a system that novice users are able to use. It is difficult to support efficiency of work for expert users at the same time.

"Usability concerns of power users (those with significant experience, training, or a professional orientation to the product) are often neglected in favor of an emphasis on success during initial interactions with a product. This is partly due to the compressed time scale (hours) of the typical usability test compared with a user's eventual experience with a product (often years). There is also a tension between initial usability and efficiency of skilled performance. A focus on initial usability elevates learnability above efficiency once up a learning curve. While this approach is appropriate for some products designed for casual users, it neglects usability issues of power users and may inhibit innovation in user interface design". [4]

DESIGN PRINCIPLES

When the specification phase started, the need to categorize different design criteria according to an expected level of importance was apparent. At this initial stage one could easily find more opinions and expectations on the future system than would be possible to take into consideration. We had to know what needs were common among both end-users and developers, but at the same time also to know how important the needs were.

We knew that **image size** alone was the single most important factor for a successful application. The target audience have high demands on their working conditions, and the importance of image size was emphasized at all user levels. Once this demand was established, everything else was left second place — a decision which solved many design obstacles.

Figure 1. The list of criteria (or principles) in order of importance.

c1. Image size
c2. Minimal user load
c3. Work oriented
c4. Support multiple user levels

List of Criteria

It was later on considered an important step, to list the main characteristics of the system according to importance. Making a "top list" will definitely keep the development on track and avoid many future inconsistencies. The design principles are not derived from a formal survey, nor are they intended to be so. Instead they reflect the wishes of the development team — or maybe even marketing reasons.

Maintaining the same focus and ideas throughout the whole development cycle can be troublesome, especially in large groups of developers and designers. We found the process of *agreeing* on a criteria list in the development group being fruitful and rewarding. It helped to bring a clearer image of the future system, and served as a common goal during the development.

We made a listing, documented in a form of Design Patterns, to further elaborate on the list of criteria. The patterns would be reusable and at the same time serve as documentation for the development process.

Principle (c1): Image size.
Intent: Make the area for images as large as possible.

Principle (c2): Minimal user load.
Intent: Make the handling of the application as easy as possible. No extra load should be put on the user. This includes short access paths to functions and minimized navigation paths within the application.

Principle (c3): Work oriented.
Intent: The application should map (support) the actual work situation (work flow) and domain to further minimize the mental load.

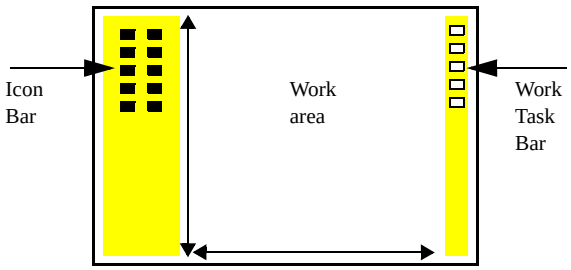
Principle (c4): Support multiple user levels.
Intent: The same application should allow both novice and expert users to gain maximal usability. There should be no expert mode needed.

DESIGN DECISIONS

Layout

Because we aimed at a square shaped area for image viewing, this automatically lead to the left and right side placement of the application related functions (the Icon Box and the Work Task Bar).

Figure 2. The general framework of the CHILI application.



The remaining area (the Work Area) then becomes more like a square, which means that we can maximize images more naturally by a scaling factor. By having this kind of layout we fulfilled our most important criteria (c1).

Single Window

Few projects have the power, knowledge and financial capability to complete an analysis of the full functionality of the developed application. Therefore, projects are often, at the end of the implementation stage, faced with a surprisingly large set of functions that have not yet been fitted into the application. These "latecomers" often end up in separate child windows due to lack of space or planning, and result in a bad design.

The traditional window handling is a resource consuming task which practically in all cases take too much attention from the real work [5]. There are many programs which simply become impossible to use due to the difficult handling of the window system itself.

Having seen too many systems not given a proper design phase and developers leaving the insertion of functionality as an excuse for popping up a new window, we decided to

remove all traditional window handling! The single window decision was supported by principle c2.

No Desktop

In reality, a single window means we don't use the desktop metaphor. A running application will occupy the complete screen and require to be on top of all other windows.

There are many reasons for not depending on a desktop metaphor; the classical desktop metaphor results in a separation of application and application data. This separated view is of little use for anyone but a systems developer. A typical user would, at all times, like to see, access and process the data through the application, never by itself or separated. [3]

Work Tasks

Without a multiple windows solution available, we split the most significant functionality into **work tasks**. This classification of basic functions is based on an analysis of the user needs and a study of the application domain. The resulting work tasks in our case are: *image viewing*, *image sending*, *image database browsing*, *image printing* and *adjusting system settings*. The notion and focus on work tasks is supported by principle c3. [8]

There is a general need to bring the syntax and semantics of the user's domain into the application context. Just by "speaking the language" of the domain there will be less misunderstandings and fewer errors made by users. Sadly, it is a non-trivial task to identify the right concepts. [7] [9]

Figure 3. A sample Work Task Bar on the left. The View Task has been selected and is shown in reverse video with a different background colour.

States

Introducing work tasks means there will be a small, fixed number of states of the application. These states could be presented as "virtual rooms", "modes" or something similar. We decided to introduce a static **work task bar** where the user would navigate between the different work tasks and to combine this with switching screens.

Switched Screens

A switched screen solution means that the complete screen will be dedicated to one single task. If the user changes task, then the complete contents of the screen changes!

A single screen per work task forces the design to be more complete from the users point of view; everything needed to complete a certain work task has to be present in one screen, and there will (can) be little or no navigation while performing a specific task.

A simplified set of functions in each task promotes usability aspects in some sense as it focuses attention on the main functionality of the application.

We are convinced that this decision will lead to a more coherent design of user interfaces. There will very likely be an improved speed of use because of the same reasons.

No Menu Bar

Users will navigate the user interface by selecting work tasks by mouse clicks or short keys. Having reduced the level of functionality and having the whole screen available for a single task allowed us to avoid using the traditional menu bar.

Our belief is that it is possible to make such a compact design of each work task, where everything needed within that specific task would fit directly onto one screen layout.

Folded Areas

We didn't want to split the application into an expert part and a novice part because of practical and economical reasons, so we had to find a solution to the conflict between initial usability and power usability.

The solution was the concept of **folded areas**, a technique that allows an expert user to hide away (to fold) some of the functions he or she does not need to find direct support for in the GUI. These hidden functions are then accessible by some other technique, for example keyboard short-cuts or pop-up menus. The folded area may later be unfolded to its original size.

This decision is a consequence of applying principle c4, and c1 to some extent.

Figure 4. The CHILI application in its unfolded state.

LESSONS LEARNED

Folded Areas

Using folded areas we managed to achieve an image area of about 85% of the screen space available. The resulting 15% is used for application navigation and control. The image area contains both the image and all the image related functions.

The corresponding ratio for the unfolded state is 70/30.

Figure 5. The CHILI application in its folded state.

Work Tasks

Focusing on the users work tasks and avoiding the traditional window handling results in an easier to use system without compromising the expressiveness for power users.

Documentation of Design Efforts

We found that discussing with developers and users while simultaneously running a prototype isn't sufficient for the transfer of design knowledge. Only in a few cases were external people (on their own) able to see which details in the prototype where carefully designed, and which ones were not. This applied even in those cases when a specific detail earlier had been discussed and thoroughly examined. It seems the importance of certain details in the prototype were difficult to comprehend.

Written Documentation

Compiling a proper documentation of a complete prototype requires enormous efforts. We didn't find the traditional way of documenting rewarding enough. Especially the documenting of decisions that were rejected, so that these alternatives do not (re-)surface later on, was exhausting and tedious work. This was made extremely apparent when we anyway had to override such a documented rejected decision; this override made some of the other prerequisites in the documentation obsolete, and absolutely impossible to trace.

We found no obvious way of documenting explanations of rejected solutions. When a design solution was "back tracked" and remodelled, this further complicated the documentation efforts. The Design Rationale technique would work fine for both the documentation and education aspects, but only when combined with a running prototype. [6]

Prototype

As mentioned earlier, not even a prototype combined with verbal explanations were always enough to assert design intentions. It is in addition only possible to test certain aspects of a user interface with a prototype. It is straightforward to test the speed of learning, but impossible to test power usability aspects.

However, the main advantage of the prototype is that once a detail is fully built, there is very little left to explain.

We found a true prototyping environment the most fitting, meaning implementing the prototype using the target platform libraries. Using a mock-up or a fake prototype is not enough to test some aspects of the GUI design, and therefore we need to implement design ideas within the target platform. It is also necessary to design ideas all the way, because the intentions of the designer are difficult to transfer, no matter what transferring or documentation method is used.

ACKNOWLEDGMENTS

We thank the CHILI development team at the Tranferzentrum Mediziniche Informatik (TZMI) in Heidelberg, Germany, and especially project manager Dr. Uwe Engelmann.

REFERENCES

1. Borälv E, Göransson B, Olssson E, Sandblad B, Usability and Efficiency - the Helios approach to development of user interfaces, *Computer methods and programs in biomedicine, supplement volume 45*, December 1994, pp. 47-64.
2. Engelmann U et al, Teleradiology System Medicus, In: Lemke (Ed). *CAR `96: Computer Assisted Radiology*, 10th International Symposium and Exhibition, Paris. Amsterdam: Elsevier (1996) 537-542.
3. Gentner D, Ludolph F, Ryan C, Designing the HotJava Views™ user environment for a network computer, Sun Microsystems, Inc. <URL: http://www.javasoft.com/products/hotjavaviews/hjv.white.html>
4. Karn K, Testing for power usability, *CHI '97 Workshop*, March 23, 1997, Atlanta, GA, http://www.acm.org/sigchi/chi97. Usenet News post on comp.human-factors on 8 Jan. 1997.
5. Lind, M, Effects of sequential and simultaneous presentations of information, *Report no. 19*, CMD, Uppsala University, 1991.
6. MacLean A, Young R.M, Bellotti V.M.E, Moran T.P, Questions, options, and criteria: elements of design space analysis, *Human-Computer Interaction* 6, 1991.
7. Norman, DA, The psychology of everyday things, *Basic Books*, 1988.
8. Olsson E, Göransson B, Borälv E, Sandblad B, Domain Specific Style Guides - Design and Implementation, *Proceedings of the Motif & COSE International User Conference*, Washington D.C. 1993, pp. 133-139.
9. Patel L.V, Kushniruk W.A, Understanding, navigating and communicating knowledge: issues and challenges, *Conference on Natural Language and Medical Concept Representation*, IMIA WG6, January 19-22 1997.