**Global Co-Operation in the New Millennium**
The 9th European Conference on Information Systems
Bled, Slovenia, June 27-29, 2001

# APPLICABILITY OF HCI RESEARCH TO E-GOVERNMENT APPLICATIONS [CASE STUDY]

**Felix Kossak**

Software Competence Center Hagenberg (SCCH), 4232 Hagenberg, Austria
Tel.: ++43 7236 3343 811, Fax: ++43 7236 3343 888
Felix.Kossak@scch.at

**Wolfgang Essmayr**

Software Competence Center Hagenberg (SCCH), 4232 Hagenberg, Austria
Tel.: ++43 7236 3343 842, Fax: ++43 7236 3343 888
Wolfgang.Essmayr@scch.at

**Werner Winiwarter**

Software Competence Center Hagenberg (SCCH), 4232 Hagenberg, Austria
Tel.: ++43 7236 3343 881, Fax: ++43 7236 3343 888
Werner.Winiwarter@scch.at

## ABSTRACT

*This paper gives an overview of the state of the art in the field of HCI (Human-Computer Interaction) and investigates the applicability of HCI research to the development of large, data intensive software systems.[1] We employ our experiences in a project on administration software for the health insurance sector, for which we designed and implemented an HCI prototype. We describe problems we encountered, focussing on the design process, user involvement, and GUI design. We propose possible solutions for these problems and conclude that for e-government applications the problems we pointed out are also typical but even more critical to solve due to issues like web-based user interface constraints, for instance, as there are neither any usability guidelines for web pages established, nor is there any usability awareness.*

## 1. INTRODUCTION

The Austrian health insurance system is based on obligatory insurance for all working people, employees as well as entrepreneurs. The administration of registrations and premiums is delegated to several health

1 This work has been funded by the K-plus programme of the Austrian government, the province of Upper Austria, and the Chamber of Commerce of Upper Austria

insurance bodies that served a total of 5.5 million customers in 1999 (Social-Net, 2000). Although these insurance bodies are no national authorities in a narrow sense, they serve important tasks on behalf of the federal government.

As "e-government" is coming onstream in the wake of e-commerce, the Austrian health insurance services are redesigning their administration systems accordingly. Facing a large and diverse community of users, usability is becoming a key factor for e-government applications.

Research concerning the usability of hardware and software for common, non-expert users started as early as the beginning of the nineteen-sixties, where e.g. the roots of graphical user interfaces (GUIs) were laid (Myers, 1998). Meanwhile, this has developed into an own, broad field of computer science, now usually termed Human-Computer Interaction (HCI).

But whereas HCI research has had a considerable influence on standard software which we all use every day, many developers of custom software are hardly aware of the ongoing research in this field, using its benefits only indirectly (e.g. by using GUI tools, which push the interface design into a certain direction, and roughly adhering to platform style guides, so far as they are known).

The most blatant indication of this is the complete lack of any consistency throughout web pages. Designers often seem to think that a page is good if it is as technically sophisticated and extravagant as possible – featuring painful colour mixes, loss of orientation, and loading times that make most of the visitors click back before even knowing what the site is about.

Many technologies featured by front-line HCI research are in the headlines now: virtual reality, speech recognition, or biofeedback, just to give a few examples. However, such technologies are hardly taken into account for most commercial projects.

But HCI research covers much more than that. Multi-disciplinary teams look into the design process and its influence on usability, investigate into how to elicit from users what they need and what they want, and assemble design principles and tips which pay respect to knowledge from cognitive psychology and design experience from various long-established fields of engineering.

In this paper, we try to assess the relevance and applicability of HCI state of the art to large e-government projects, drawing experience from a project with the Austrian health insurance services.

Section 2 gives an overview of the state of the art and current HCI research, in so far as it was relevant for our project. The focus is put on general design principles, and GUI design in particular, as well as on the design process from the usability point of view.

In Section 3, we give a description of our project, its context and its constraints. We developed an exemplary GUI prototype for a large, administrative software system for the Austrian health insurance services.

This gives the background for Section 4, where we investigate the applicability of HCI research to commercial, comprehensive administration systems under such constraints. We use examples and experience from our project in three exemplary areas – user involvement, consistency across the application, and single GUI aspects – to demonstrate the possible benefits of applying scientific work in this field to fairly conventional software projects.

## 2. HCI STATE OF THE ART AND RELATED WORK

The Special Interest Group on Computer-Human Interaction of the Association of Computing Machinery (ACM SIGCHI) defines Human-Computer Interaction (HCI) as "a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them" (ACM SIGCHI, 1992).

The stress lies on that part of the system, which is actually supporting the interaction; as IBM (2000) formulate it, HCI is "the study, planning, and design of what happens when you and a computer work together".

The notion of HCI could also be used for a pronouncedly user-centred approach in designing such systems as contrasted with the functionality-centred "traditional" approach, and this is where actually a great part of HCI research is focussed upon.

The current focus in HCI research lies on input / output techniques on the one hand and user-centred design methods as well as usability evaluation on the other hand. Many researchers also investigate the impact and applicability of on-stream software technologies to HCI, such as component software, agents, and revived AI.

In another development, Myers et al. (2000) see an increasing need for end-user customisation, programming, and scripting, which will have to be supported accordingly by modern user interfaces.

In the context of the specific project we were working on, we will concentrate on software design issues: standards and guidelines, GUI design principles, and user involvement in the design process.

### 2.1. Design Standards and Guidelines

As will be outlined below, an important design principle is consistency, not only within one product, but also across different applications, operating systems and hardware. This can only be achieved using standards. Standards and guidelines also carry a vast amount of experience and knowledge for designers to exploit.

However, as is pointed out throughout the literature, "Standards must be continually reviewed and updated. If not, they tend to freeze technology and stifle innovation" (Mandel, 1997). The development of an ISO standard, for instance, may take more than a decade (as described in Stewart, 2000), rendering at least parts of it outdated.

Moreover, as shown in a study of Thovtrup and Nielsen (1991), even usability standards can be fairly unusable themselves, due to size, bad structure and inappropriate style.

Consequently, it will always be necessary to filter the relevant information from different standards and guidelines into custom style guides, adapt and supplement them for one's own organisation and for every new project, and constantly update them.

The most important international standard for HCI is ISO 9241, Ergonomics Requirements for Office Work with Visual Display Terminals. The 17 parts of the standard deal not only with (conventional) hardware, but also with general usability issues, design, general GUI features, and GUI details like specific forms of dialogues.

ISO 13407, Human-centred Design Processes for Interactive Systems, deals with design processes as well as key design principles.

ISO 14915, Multimedia User Interfaces, is still in draft status (systemconcepts, 2000).

National examples are ANSI/HFES 200 (USA), which includes parts of ISO 9241, or Germany's DIN 66234 on ergonomics and HCI.

There are many proprietary guidelines, which aim to guarantee consistency throughout specific platforms, such as the MOTIF Style Guide, the Microsoft Style Guide, IBM's Common User Access guidelines (CUA), or Apple's Macintosh Human Interface Guidelines.

### 2.2. UI Design Principles and Techniques

The most striking and colourful user interface is not necessarily the best one. Animated objects and inappropriate sounds and colours are usually distracting rather than motivating, especially in the long run (see "Las Vegas effect" and "habituation" in Mandel, 1997). The best interface is rather the one, which you don't have to pay much attention to (IBM, 2000). Mandel (1997) formulates the key principle of user-centred design: "Let users do what they want to do, when they want to do it, and how they want to do it." It is

obvious from this that user-centred design does not only concern the actual interface, but the product as a whole, so that the underlying logic or implementation details do not force the user to do things in a certain way.

On the other hand, user interface design must take into account various psychological and physical properties and constraints of humans: the way that humans perceive, how short-term and long-term memories work, what tasks humans are generally good at and for what tasks they need support from the machine.

Mandel (1997) identifies three areas of user interface design principles:

- Place users in control
- Reduce users' memory load
- Make the interface consistent

An issue concerning both the control over the interface and memory load is that of modal dialogues. Again and again, we are asked for information that can be found on another window, forcing us to close the dialogue – because it is modal (controls the interface) – to look the information up and restart the dialog equipped with some handwritten notes (memory load) (see e.g. Mandel (1997), IBM (2000)).

User control is concerning various aspects: the sequence of tasks and steps therein, the possibility to interrupt tasks, the media and form of input and output, or the arrangement and appearance of objects, background, etc. Some see customisation more and more extended towards scripting, while others point out that also scripting requires programming skills, which most users will not have.

When the user is in control, the computer's role is to provide "proactive assistance" (IBM, 2000). This includes the indication of the current state of the system and the possible actions as well as immediate reactions to user input.

Another key element to facilitate user control is "forgivingness" (reversibility of user actions; see e.g. Chrusch, 2000). Possible exceptions and interruptions should be identified early in the design process, preferably together with user scenarios (Monk, 1998).

Humans are rather bad at recalling, but much better at recognition: thus, for instance, they rather find the right item in a list than remembering the correct name (of a command, etc.). Metaphors are very helpful; however, many authors complain that many metaphors are inappropriate (see e.g. Mandel (1997), IBM (2000)).

Metaphors must be consistent, just as all other elements of the interface. Inconsistent behaviour within a product or across products often leads to superstition. In consequence, users are discouraged from "exploring" a new system (Mandel, 1997).

Other issues to be considered are simplicity, versatility, intuitiveness and familiarity, but also safety and encouragement (compare: IBM, 2000).

### 2.3. The Design Process

Visual design should not be the "icing on the cake" but an integral part of the design process (IBM, 2000). The technically useful separation between business logic and presentation should not lead to inconsistent design or technical details restricting user interaction.

The goal of the design process is to transform the user's implicit conceptual model via the designer's model of the intended overall user experience to the programmer's model. Presentation makes up for only about 10% of the designer's model, interaction another 30%, but the main focus lies on the user's objects, their relationships and behaviour (the "look-and-feel iceberg", Mandel, 1997).

ISO 13407 describes four basic design principles:

- Active involvement of users in the design process

- Appropriate use of human skill by function allocation
- Iteration
- Small, multi-disciplinary teams

Furthermore, four key design activities are mentioned:

- Understand and specify the context of use
- Specify user and organisational requirements
- Produce several design solutions
- Evaluate the solutions against the requirements by real user testing

In the industry, the trend moves actually in the same direction, in particular with respect to user involvement, small and multi-disciplinary design teams, and iterative design (see e.g. Atyeo et al. (2000) (Nortel), MacLean (2000) (Xerox), IBM (2000)).

While there are various methods known for gathering and representing user information on the one hand and designing on the other hand, Wood et al. claim that most designers still see the transformation from the specification to the design as a piece of art, or "a little magic" (Wood, 1998). They describe various techniques for "bridging this gap".

The two major factors for improving the design process are user involvement and multi-disciplinary design teams. The classical way of involving users in the development process is to build prototypes and let (prospective) users evaluate them. In the past, this was done only at the stage of alpha- and beta-versions (usability testing) where design was already completed; nowadays, GUI builder tools enable rapid prototyping, which makes it easier for the developer to involve the user already in the design of the user interface.

While Mandel (1997) points out the importance of asking the actual users what they want instead of (only) their managers, it is also important to involve all other interest groups ("stakeholders") as well, such as administrators and the management (see e.g. Microsoft (1999), Monk (1998), Graefe (1998)).

Beyond interviews and usability testing, user input can be collected in more direct ways:

- Observation: Users are observed in their usual environments or in special studios that simulate that environment. Gilmore and Velázquez (2000) stress the need not only to understand what the users do, but also why they do it.
- "Creativity sessions": Many techniques are known for analysis and basic design sessions that can involve user representatives, usually building on brainstorming and association techniques.
- Walkthroughs: To evaluate initial design alternatives, these are visualised as low-fidelity prototypes, usually sketches on paper.
- High-fidelity Prototyping: Different alternatives for the overall design are evaluated with prototypes that have no functionality except basic navigation; later, when the design of the different parts is refined, more functionality can be required. Prototyping is done in iterative steps. Through interaction with the user interface, users' conceptual models may be expanded, which in turn may cause them to realise new requirements that they had not thought of before (IBM, 2000).

User-centred design requires multi-disciplinary teams. HCI involves technology, arts, psychology, and marketing. IBM (2000) project teams e.g. include visual or industrial designers, specially trained HCI designers, marketing specialists, as well as service and support specialists. Microsoft (1999) define, amongst others, special roles for logistics management and user education. Mandel (1997) advises to engage people with writing skills for all textual aspects (see also Johnson (2000)).

However, there is a trend towards smaller teams. For instance, Dayton et al. (1998) use five-member design teams (expert user, novice user, usability engineer, developer, system engineer).

### 2.4. GUI Tools and Class Libraries

Graphical user interfaces (GUIs) on all common platforms have hardly changed since the Macintosh interface of 1984. This has the advantages that users find it easy to deal with a new system once they are acquainted with one, and that tool builders had plenty of time for refining the concepts (Myers et al., 2000).

The major impacts of GUI tools on the user are

- The achievement of a universal and consistent look-and-feel,
- The influence of a specific tool on the developer towards certain features and
- The support of rapid prototyping, by which the user can be better involved into the design.

State of the art are (still) so-called interface builders, such as 4GL (fourth-generation language) tools, whereas tools based on constraints, formal languages, or automatic and model-based techniques (generation from input / output specification) remained unsuccessful so far. This is predicted to change in the near future, when the user interfaces for one application must be specified hardware- and system-independent and then automatically generated for each sort of interface, which can range from a mobile phone display to “wall size” displays and various input devices.

While current GUI tools support design and implementation, Myers et al. (2000) also demand an integrated support for rapid evaluation.

## 3. PROJECT DESCRIPTION AND CONSTRAINTS

We participated in the design of a large e-government like system for administering health insurance service cases. This system is developed at three different health insurance bodies and will be used throughout Austria. This means that project documents and prototypes are evaluated by representatives of most of the Austrian health insurance bodies through the umbrella organisation HVB (“Hauptverband”). These representatives convene in special evaluation meetings.

The goal of this system is to enable customer-oriented acceptance and proceeding of claims. Special interfaces shall be provided for external partners like doctors, hospitals and ambulance services. The system also interacts with other health insurance IT systems, for instance for the administration of customers and other partners. Presently, a legacy system with a text-based interface covers much of the functionality, but the new system was designed completely from scratch.

The software system has separate modules for different kinds of services, such as (transient) inability to work, stationary treatment, or travel and transport costs. A central module performs the assessment of whether and how a customer was insured at a certain time. Other modules manage customer accounts or provide general functionality such as the administration of dates connected with particular insurance cases.

The goal of our part of the project was to build an HCI prototype to serve as a paragon for this system. We built a GUI prototype with navigation and limited functionality for usability evaluation. We started our work after the analysis and system design phases, at the beginning of the prototyping phase. While a rough task description was already supplied, we were still able to refine it and to develop user scenarios together with a user representative and an analyst of the IT Development Department of our partners, and to exert some influence on the user object model, which was otherwise in an advanced stage.

For documenting user requirements, the process model used by the Austrian health insurance bodies prescribes formal task descriptions and scenarios, on which prototyping builds.

The technical constraints were to build a front end using the PowerBuilder GUI builder tool and a special class library provided by the umbrella organisation of Austrian health insurance bodies. The class library, together with specific GUI guidelines (basically based on the MS Windows Style Guide), enables a common look-and-feel throughout custom-made applications in the Austrian health insurance sector, as well as a high level of reuse. The architecture we built upon is a client-server system with an intermediate transaction system that allows for the use of distributed, heterogeneous databases. Although this transaction service provides functions for easier database access, some of the business logic is to be realised on the PowerBuilder client, which makes it difficult to achieve the goal of having real "thin" clients.

Besides these constraints, we feel that the applicability issues we gained from our experience in that project (see Section 4) are even more important when developing large administration systems in a Web-based, thin-client, and multi-tier e-government environment. A user-centred design, i.e. defining the group of users and taking their needs, tasks, experience, etc. into account for designing the navigational system, the applications and content to be provided, is especially important in e-government environments, independent of the technology deployed.

## 4. THE APPLICATION OF HCI RESEARCH IN THE AREA OF LARGE ADMINISTRATION SYSTEMS

The application of "state-of-the-art" techniques and research results to urgently needed (as always), large systems in the industry, the civil services and related areas is limited by the fear of additional sources for glitches through insufficiently tested technologies and lack of experience. Managers as well as engineers are often hesitant to employ software with version number 1.0.

This becomes even more difficult when the introduction of new computer systems and software is required to be approved through a top-level, formal process in big organisations, or – even worse – in big associations of organisations. For instance, the introduction of new developer software or new techniques may render an elaborate, corporate class library worthless. The same holds true for changes in the development processes of custom-made systems.

In the course of our project, we had to learn that large organisations cannot easily replace the equipment for hundreds or thousands of staff. Being used to modern, large and high-resolution screens, or fast PCs and LANs, we were forced to build the prototype for considerably smaller screens and ran into performance problems very easily.

However, in practice, there is always a considerable scope for designers and developers within the given constraints, where scientific research can contribute to the usability of the output. Most developers, managers and users are not aware of usability engineering and HCI as an existing discipline and that extensive knowledge is available in this field (Giller and Tscheligi, 2000). Our goal was to add HCI expertise to the project of the health insurance services and, in turn, to draw specific experience out of the practical work within the project.

### 4.1. User Involvement

Typically, a developer gets the scenarios for one task, which include all the data that the user is required to enter as well as the required output data, and often supplemented by a rough pseudocode draft. Out of this, he or she creates a few windows (or, rather, some tab-pages for one window), which are then discussed with the user representative assigned for the respective module.

Developers are urged to create evolutionary prototypes, which can be further developed to a release version, and time is fairly scarce. Consequently, the first drafts presented to the user representatives are already full-blown software prototypes with all the layers that the class library prescribes: window, tab-pages, data window controls, and data window objects with data items.

It takes quite some time and effort to get to this stage. It is hardly surprising, then, that developers become frustrated when user representatives reject their drafts, realising that it will take several iterations until they are satisfied.

We have therefore created the first drafts for our exemplary work package by simply dropping standard controls supplied by the GUI builder tool onto a plain window, ignoring the class library for the time being. This can be done very quickly, and changes can be performed very easily.

Wherever applicable, two or more alternatives should be created for a screen – which is also facilitated by the ease of creating low-fidelity prototypes using a standard GUI tool. People always find it easier to select between alternatives, or combine features of different alternatives, rather than to state clearly how they would like to alter the interface of a single draft given.

One should also include alternatives even if one does not particularly like them, as users might see them in a completely different light, or spot some detail, which they would like to be added to the other version.

Only after one draft was finally approved, we built a full prototype, scrapping the low-fidelity prototypes. We think that this process saved considerable time in the end, and at the same time enhanced quality by encouraging input from the user representative rather than leaving a bad conscience for causing troubles.

A still better way would have probably been to make at least one brainstorming session at the very start together with more than just one user representative as well as the analyst responsible for the respective module. This should have produced a low-fidelity prototype (or several prototypes) in the form of rough, hand-drawn paper sketches. The required infrastructure such as a special design room (see Simpson (1998): “UI War Room”, Rohlfs (1998)) would not be very expensive.

Unfortunately, we ran into troubles after the first high-fidelity prototype was reviewed by more user representatives, who detected that we had used a wrong model of the hierarchy of organisational units (such as branches, departments and working groups), based on a preliminary data model which had been altered later on.

This seemed to be only a detail at first sight, but it had a major impact on the logic involved (such as enabling and disabling a considerable amount of functionality under certain conditions), and it caused a lot of work to change the prototype accordingly. We think that this could have been possibly avoided if more user representatives had been available at an earlier stage and had been involved more intensely (i.e. when experimenting with low-fidelity prototypes, or even before, when developing the user scenarios).

### 4.2. Towards Consistency Across the Application

To achieve consistency and a common look-and-feel across a large application, it does not suffice to rely on standards and platform-specific guidelines, or to apply corporate style guides (although the umbrella organisation of Austrian health insurance bodies have a fairly detailed style guide).

The development of an application-specific style guide actually proved necessary in the course of the project. It was done implicitly in the form of consolidated minutes of the weekly prototyper meetings, supplemented by tables for common abbreviations, field sizes, etc.

Apart from the implicit and rather unstructured form of the project “style guide”, we also think that the forum in which it evolved was not ideal: the developers stipulated the details in weekly meetings and implemented them, only to find that user representatives, who did not participate in these meetings, were not happy about some of them. Also the “analysts” (designers) should have been explicitly involved.

Work on the style guide could have started even before the prototyping phase. However, the involvement of the developers meant that they knew the rules much better than they would have if they had just read them.

The problems described above are well known. The importance of project-specific style guides and their development in the design phase is discussed by Rohlfs (1998). Johnson (2000) brings examples that

illustrate the importance of a consistent terminology. Thovtrup and Nielsen (1991) point out problems concerning the applicability of standards (and guidelines) by developers (see also Subsection 2.1).

An important instrument for achieving application-wide consistency is a sample prototype (Rohlfs, 1998). This should be accompanied by a "how-to" recipe and by contributions to the project style guide. This was actually the task we set out to accomplish for this project; unfortunately, due to lack of staff, we started more than two months late – shortly after the prototyping team had commenced their work.

The negative effects we encountered underline the importance of such a sample prototype. Especially developers who were not yet familiar with the class library and its underlying ideas found it hard to get started. Prototypes from another project that employed the same class library and the private "recipe" of one developer proved to be helpful, yet insufficient.

Due to our experiences in that project, we recommend to start with a sample prototype considerably earlier than two months before general, high fidelity prototyping, if one is not familiar with the class library to be used (as was the case here). If such a class library does not exist, even more time must be planned – at least half a year (though this will not suffice to develop a new class library; this could then be done, at least partially and after sufficient preparation, together with the other prototypers).

### 4.3. The GUI Appearance

Throughout the literature, it is recommended to build a user interface around a central metaphor. Although we do not know whether this was chosen consciously, the pervading metaphor of the overall project is a (paper) form (or a set of forms). This is a very common metaphor for this kind of applications, and is virtually forced upon by the GUI builder used (actually, by most GUI builder tools).

It is probably an adequate metaphor from "real" life (see Rohlfs, 1998): before the use of computers, clerks would have completed forms or, rather, have patients fill them in. Many of the tasks still involve entering data from a paper form handed in by a patient into the computer. Moreover, the step from the old, character-based system is not too big so that clerks already used to the old system should find it relatively easy to switch.

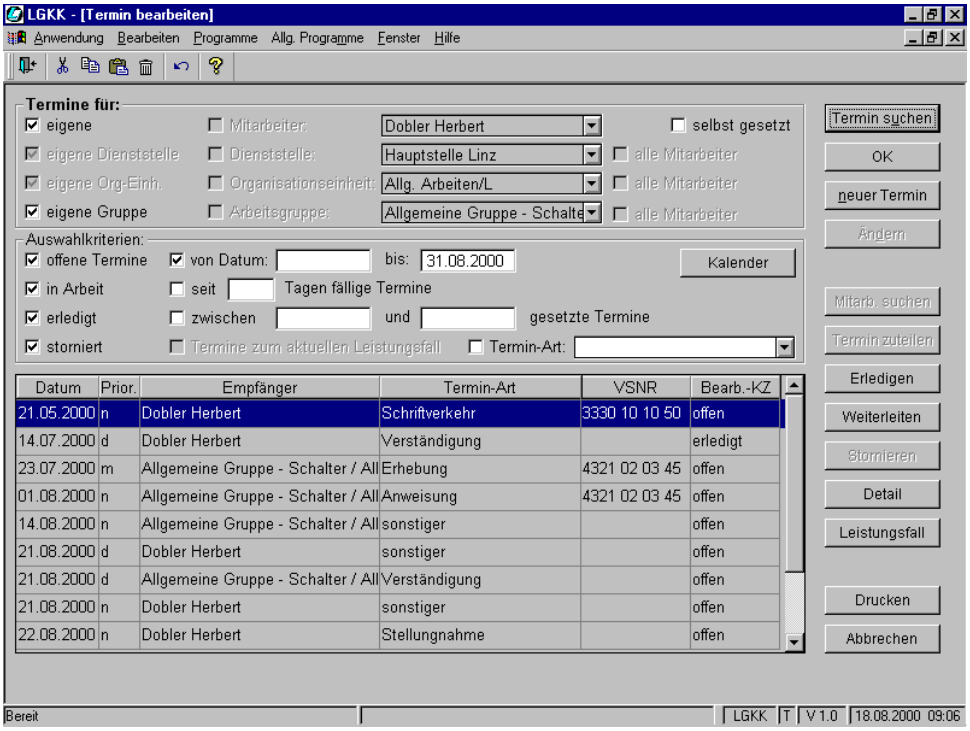


***Figure 1:*** *A typical screen (data are fictional).*

Despite the object-oriented implementation, the interface itself is task-oriented. Although this is rather against the trend, we can argue with Rohlfs (1998) that this is adequate for a transaction-centred type of software like this, where a particular sequence of steps is usually important. Also the present way in which things are done is basically task-oriented.

It could, however, be questioned whether the overall appearance could not have been made a little less "dull" and packed. Figure 1 shows a typical window of the prototype for searching and editing dates ("Termin bearbeiten"; sample data are fictional). We expect that the constant use of screens like that is likely to cause fatigue. Push buttons as seen on the right-hand side of the screen are not commonly used any more like this, although they are still supported and partially even promoted by relevant tools. Nowadays, people are rather used to invoke actions via the main menu, pop-up menus (right mouse click), and very frequently, the tool bar, where buttons are usually marked by graphical symbols, which can be identified more quickly than text after a relatively short learning period – especially if the symbols are intuitive enough.

By moving the buttons from the right-hand side to the tool bar (in an appropriate form), the screen would gain a little more "colour", and – more importantly – the space gained thereby could be used to relax the window and structure it better. Single buttons, such as the "Kalender" (calendar) button in the picture, could possibly be retained (for instance, to indicate the context with specific entry fields), but rather as a picture button with a square form and a graphical symbol, adding even more colour.

The style guide of our partner's umbrella organisation explicitly forbids the use of graphical symbols on buttons except for four particular symbols for navigation in data windows. While this measure guarantees a high level of consistency, there are many more common and even standardised graphical symbols that are familiar not only to MS Windows users, but across different platforms. Additional symbols could be added to the corporate style guide (though this should be handled with care not to strain the users' imaginations).

It is often only for small details which sum up to a more user-friendly interface: for instance, the visual distinction between compulsory and optional entry fields (we solved this by giving compulsory fields a bold caption), or simply the clear structuring of GUI controls into groups which can be easily surveyed. Such details are very often not considered by developers, which stresses the importance of HCI training for the whole development team.

## 5. CONCLUSION

Whereas popular HCI research issues are mostly centred upon revolutionary input and output technologies, most of the current commercial software projects are too limited in resources and time to experiment with novel technology. Still, there is considerable scope and need to improve the usability of conventional software.

We have given some examples of problems from three different areas concerning usability in a commercial administration software project, namely, user involvement, consistency, and GUI appearance. We have also put forward some exemplary proposals for tackling them.

However, our project has also shown that the typical constraints of such a project make it hard to introduce state-of-the-art technology, although more would certainly be feasible. We think that unnecessary limitations are due to high-level decisions in the first stages of the project. Adequate counselling in those stages by HCI experts from the scientific community could probably lead to a considerably higher usability level of the resulting system. On the other hand, due to the intensive dialog with our project partner during the project we jointly developed several incitements that considerably enhanced the project results paving the way for the forthcoming implementation phase.

The problems we described for a large, administrative software system become even more critical with web-based e-government interfaces – a possible supplement for the system that we worked on. There is hardly any awareness for usability issues of web interfaces so far, which is reflected in the lack of any respective style guide. Usability experts such as Jakob Nielsen (see Nielsen, 2000) are trying to convince the web community to consider usability aspects in page and site design (e.g., navigation, linking, style sheets, frames

versus no frames) but also in content design (how to present different content legibly with acceptable response times, how to use animation, videos and other multimedia sources). Even though studies are being conducted on usability aspects, the enormous speed in the technological development of the web and its use make the development of generally accepted standards hard, especially as they would have to be adapted constantly.

## REFERENCES

ACM SIGCHI – Association For Computing Machinery / Special Interest Group On Computer-Human Interaction (1992). *Curricula for Human-Computer Interaction*. Web page: http://www.acm.org/sigchi/cdg/cdg1.html (accessed on 09.10.2000).

Atyeo, M., J. Ramsay, and J. Rattle (2000). From Behaviour to Innovation at Nortel Networks. In *CHI 2000 Extended Abstracts* (Szwillus, G. and T. Turner Eds.), p. 233-234, ACM Press.

Chrusch, M. (2000). Seven Great Myths of Usability. *ACM Interactions*, 7 (2), 13-16.

Dayton, T., A. Mcfarland, and J. Kramer (1998). Bridging User Needs to Object Oriented GUI Prototype via Task Object Design. In *User Interface Design: Bridging the Gap from User Requirements to Design* (Wood, L.E. Ed.), p.15-56, CRC Press.

Giller, V. and M. Tscheligi (2000). Making Usability Engineering Happen: Centre for Usability Research & Engineering (CURE). In *CHI 2000 Extended Abstracts* (Szwillus, G. and T. Turner Eds.), p. 223-224, ACM Press.

Gilmore, D.J. and V.L. Velázquez (2000). Design in Harmony with Human Life. In *CHI 2000 Extended Abstracts* (Szwillus, G. and T. Turner Eds.), p. 235-236, ACM Press.

Graefe, T.M. (1998). Transforming Representations in User-Centered Design. In *User Interface Design: Bridging the Gap from User Requirements to Design* (Wood L.E. Ed.), p.57-79, CRC Press.

IBM (2000). *What Is HCI?* Web page: http://www-3.ibm.com/ibm/easy/eou_ext.nsf/Publish/10, and linked pages (accessed on 09.10.2000).

Johnson, J. (2000). Textual Bloopers: An Excerpt from GUI Bloopers. *ACM Interactions*, 7 (5), 28-48.

Ludolph, F. (1998). Model-based User Interface Design: Successive Transformations of a Task/Object Model. In *User Interface Design: Bridging the Gap from User Requirements to Design* (Wood, L.E. Ed.), p.81-107, CRC Press.

Maclean, A. (2000). Xerox Research Centre Europe (XRCE). In *CHI 2000 Extended Abstracts* (Szwillus, G. and T. Turner Eds.), p. 217-218, ACM Press.

Mandel, T. (1997). *The Elements of User Interface Design*. John Wiley & Sons, Inc, New York.

Microsoft (1999). *Microsoft Official Curriculum: Principles of Application Development*.

Monk, A. (1998). Lightweight Techniques to Encourage Innovative User Interface Design. In *User Interface Design: Bridging the Gap from User Requirements to Design* (WOOD, L.E. Ed), p.109-129, CRC Press.

Myers, B.A. (1998). A Brief History of Human Computer Interaction Technology. *ACM Interactions*, 5 (2), 44-54.

Myers, B., S.E. Hudson and R. Pausch (2000). Past, Present, and Future of User Interface Software Tools. *ACM Transactions on Computer-Human Interaction*, 7 (1), 3-28.

Nielsen, J. (2000). *Designing Web Usability*, New Riders Publishing.

Rohlfs, S. (1998). Transforming User-Centered Analysis into User Interface: The Redesign of Complex Legacy Systems. In *User Interface Design: Bridging the Gap from User Requirements to Design* (WOOD, L.E. Ed.), p. 185-214, CRC Press.

Simpson, K.T. (1998). The UI War Room and Design Prism: A User Interface Design Approach from Multiple Perspectives. In *User Interface Design: Bridging the Gap from User Requirements to Design* (WOOD, L.E. Ed.), p. 245-274, CRC Press.

Social-Net (2000). *Beitragsleistende Versicherte im Jahresdurchschnitt* (Annual Average of Paying Insured; in German). Web site of the umbrella organisation of Austrian health insurance bodies (HVB): http://www.sozvers.at/hvb/statistik/index.htm (accessed on 07.11.2000).

Stewart, T. (2000). *Ergonomics User Interface Standards: Are they More Trouble than they Are Worth?* Web page: www.system-concepts.com/stds/standards.pdf (accessed on 13.10.2000).

Systemconcepts (2000). *ISO 14915 Status*. Web page: http://www.system-concepts.com/stds/ISO14915.html (accessed on 13.10.2000).

Thovtrup, H. and J. Nielsen (1991). *Assessing the Usability of a User Interface Standard*. Web page: http://www.useit.com/papers/standards.html (accessed on 20.10.2000).

Wood, L.E. (1998). Introduction: Bridging the Design Gap. In *User Interface Design: Bridging the Gap from User Requirements to Design* (WOOD, L.E. Ed.), p. 1-14, CRC Press.